A derivative-free approach to optimal control problems with a piecewise constant Mayer cost function

C. Audet, P.-Y. Bouchet, L. Bourdin

A derivative-free approach to optimal control problems with a piecewise constant Mayer cost function

Charles Audet [a, b]

Pierre-Yves Bouchet [a, b]

Loïc Bourdin [c]

[a] *GERAD, Montréal (Qc), Canada, H3T 1J4*

[b] *Polytechnique Montréal, Montréal (Qc), Canada, H3T 1J4*

[c] *XLIM Research Institute, UMR CNRS 7252, University of Limoges, France*

charles.audet@gerad.ca
pierre-yves.bouchet@polymtl.ca
loic.bourdin@unilim.fr

Abstract :   A piecewise constant Mayer cost function is used to model optimal control problems in which the state space is partitioned into several regions, each having its own Mayer cost value. In such a context, the standard numerical methods used in optimal control theory naturally fail, due to the discontinuities and the null gradients associated with the Mayer cost function. In this paper an hybrid numerical method, based on both derivative-free optimization and smooth optimization techniques, is proposed to solve this class of problems. Numerical simulations are performed on some standard control systems to show the efficiency of the hybrid method, where NOMAD and IPOPT are used as, respectively, derivative-free optimization and smooth optimization solvers.

Keywords: Optimal control, piecewise constant Mayer cost function, derivative-free and blackbox optimization

Résumé :   Une fonction de coût de Mayer constante par morceaux est requise pour correctement modéliser des problèmes de contrôle optimal dans lesquels l'espace des états est partitonné en différentes régions ayant chacune une valeur de coût de Mayer. Dans un tel contexte, les méthodes numériques communes dans la théorie du contrôle optimal échouent naturellement, à cause des discontinuités et des gradients nuls associés à la fonction de coût de Mayer. Dans cet article, nous proposons une méthode numérique hybride, exploitant des techniques d'optimisation lisse et d'optimisation sans dérivées, pour traiter cette classe de problèmes. Nous présentons des simulations numériques sur plusieurs systèmes de contrôle classiques pour montrer l'efficacité de notre méthode hybride, en exploitant les solveur IPOPT pour l'optimisation lisse et NOMAD pour l'optimisation sans dérivées.

Mots clés :   Contrôle optimal, fonction de coût de Mayer constante par morceaux, optimisation sans dérivées et de boîte noire

1 Introduction

1.1 Motivation

The present work was initially motivated by numerically solving Bolza optimal control problems of the form

$$
\begin{aligned}
&\underset{\substack{x \in \mathcal{AC}([0,T],\mathbb{R}^n) \\ u \in \mathcal{L}^\infty([0,T],\mathbb{R}^m)}}{\text{minimize}} \quad \lfloor \|x(T) - x_{\text{end}}\| \rfloor + \int_0^T \ell(t, x(t), u(t))\, dt \\
&\text{subject to} \quad \dot{x}(t) = f(t, x(t), u(t)), \quad \text{a.e. } t \in [0,T], \\
&\qquad\qquad\quad x(0) = x_0, \\
&\qquad\qquad\quad c(t, x(t), u(t)) \leq 0, \qquad \text{a.e. } t \in [0,T], \\
&\qquad\qquad\quad c_T(x(T)) \leq 0
\end{aligned}
\tag{P_{Motiv}}
$$

where standard notations are used (recalled in Sections 3 and 4) and in which the dynamics, the constraint functions and the Lagrange cost function are smooth, but where the Mayer cost is discontinuous, being defined as the truncation (involving the floor function $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$) of the distance between the final state $x(T)$ and a reference target x_{end}. Note that the corresponding Mayer cost function is piecewise constant with integer values. In this paper we call *plateaus* the subsets of $\mathbb{R}^n$ on which a piecewise constant function from $\mathbb{R}^n$ to $\mathbb{R}$ has a given value. As illustrated in Figure 1(a), in the two-dimensional case $n = 2$ the plateaus of the Mayer cost function of Problem (P_{Motiv}) are concentric rings centered at x_{end}. Such a Mayer cost function is used to model practical problems in which a kind of *discrete distance* to a target is evaluated, for example in precision sports (shooting, darts, archery, precision landing of a parachute, etc.). Figure 1(b) illustrates several properties of these problems. For example, two trajectories ending on the same plateau are only discriminated by the Lagrange cost. Also, a trajectory with a high Lagrange cost ending on a given plateau may have a smaller Bolza cost than another with a low Lagrange cost but ending on a higher-value plateau. Finally, the two trajectories ending on the lowest-cost plateau show that reaching x_{end} as close as possible may not be the optimal solution. These examples show that one cannot remove the truncation in the Mayer cost function without altering (possibly significantly) the optimal solutions.
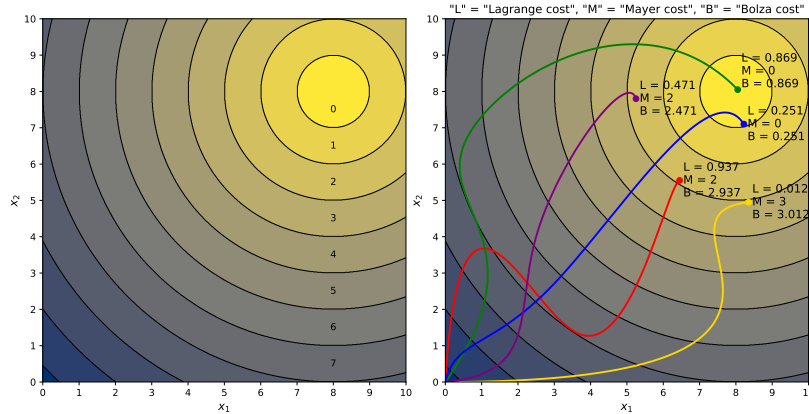
Figure 1: **(a)** Representation of the Mayer cost function of Problem (P_{Motiv}) **in the two-dimensional case** $n = 2$ **with** $x_{\text{end}} = (8, 8)$. **(b)** Artificial trajectories and their associated costs.

In optimal control theory, the numerical methods are usually classified as either direct or indirect methods [1, 2]. An indirect method relies on the first-order necessary optimality conditions provided by the *Pontryagin maximum principle* (PMP). Most of the literature on PMP concerns smooth problems (see, e.g., [3, 4, 5]), but extensions exist for problems satisfying (only) Lipschitz-continuity assumptions, by using generalized notions of gradient (see, e.g., [6, 7, 8]). Some papers derive a PMP for problems with discontinuities, such as [9] in the context of hybrid dynamics, or [10] in the context of time

crisis problems where discontinuities occur as an indicator function in the Lagrange cost function. However, deriving a relevant version of the PMP in our context is unlikely because optimal trajectories are inclined to end on the boundaries of the plateaus of the Mayer cost function, where gradients and generalized gradients are not defined. Our numerical simulations provided in Section 4 confirm this phenomenon. Therefore, in this paper, we rather focus on direct methods which rely on a full discretization of the optimal control problem, via a Runge-Kutta method [11] for example, in order to recover a finite-dimensional optimization problem that can be numerically solved using a relevant optimization algorithm.

However, a full discretization of Problem (P_{Motiv}) produces a finite-dimensional optimization problem which may preserve the discontinuity of the Mayer cost function, while most optimization algorithms in the literature are dedicated to smooth problems (see, e.g., [12]). Nevertheless, one can easily design numerical approaches attempting to overcome this obstacle. A first approach consists in considering a smooth approximation of the discretized problem that can be numerically solved using a *smooth optimization* (SO) solver such as IPOPT [13]. However smoothing the piecewise constant Mayer cost function leads to a smooth function with gradients which are null (or almost null) on most of the domain. Hence they do not provide useful information to the SO algorithm which thus focuses only on the optimization of the Lagrange cost, while the latter may be less rewarding than optimizing the Mayer cost, as illustrated in Figure 1(b). Our numerical results provided in Section 4 confirm that this first approach is inefficient. A second approach relies in solving directly the full discretization of Problem (P_{Motiv}), but with a *derivative-free and blackbox optimization* (DFBBO) algorithm. Indeed these algorithms are designed to solve singular finite-dimensional optimization problems. DFBBO relies only on a proximal analysis of candidate solutions close to the current incumbent, and requires neither existence nor values of gradients [14, 15]. In this paper we focus on the MADS algorithm [16] implemented in the NOMAD solver [17], but other techniques exist (see, e.g., [18, 19, 20, 21]). MADS can be applied to a full discretization of Problem (P_{Motiv}), even if the discontinuities of the original problem are preserved. However MADS is usually efficient for problems with a few dozens of variables at most [14, Section 1.4], while a discretized optimal control problem usually has a much larger scale. Our numerical results provided in Section 4 confirm that this second approach is also inefficient.

1.2   Contributions

The difficulties mentioned above are actually present in a whole class of optimal control problems. For example, any piecewise constant Mayer cost function leads to a lack of information provided to SO algorithms, because its gradients are zero (and remain almost-zero after smoothing). As discussed above, these difficulties cannot be addressed easily with common techniques from the literature. Therefore, the objective of the present paper is to propose a numerical method to solve efficiently optimal control problems with a piecewise constant Mayer cost function, in view of handling various situations of possible plateaus (as illustrated in Figure 2 in the two-dimensional case $n = 2$). To illustrate polygonal plateaus as in Figure 2(c), consider parking a car downtown where we look for a cheap parking lot, while the position of the exact parking spot within this parking lot is less relevant. In such a problem, the parking lots are the plateaus, each having its own Mayer cost (for example the price to park in the lot), while all parking spots in the same parking lot have the same quality. However note that reaching a parking spot far from the entrance requires more displacement time. Hence some local considerations (taken into account by the integral Lagrange cost) may discriminate two parking spots, but this optimization has a noticeably smaller impact than the choice of the parking.

To propose a numerical method to solve efficiently optimal control problems with a piecewise constant Mayer cost function, we exploit the fact that the inner difficulty depends entirely on the final state variable only. The main idea is thus to optimize the final state alone via a DFBBO algorithm, while a smooth subproblem, which consists in solving the initial problem with the final state being fixed and with a minimal Lagrange cost, is solved via a SO algorithm. The resulting method is called *hybrid* as it combines tools from DFBBO and SO.
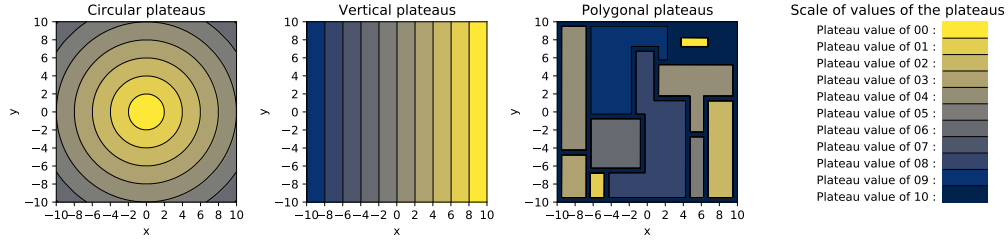
Figure 2: **Examples of possible plateaus in the two-dimensional case** $n = 2$. **In all figures of the present paper, the light-colored plateaus represent the lowest-cost plateaus to reach, while the dark ones represent the highest-cost plateaus to avoid.**

The idea to separate variables can be extended to handle cases where the final state is not the only variable affecting a singularity. Consider for example a variant of Problem (P_{Motiv}) where the Mayer cost is defined as $\lfloor \|x(T) - x_{\text{end}}\| \rfloor + \lfloor \|x(T/2) - x_{\text{mid}}\| \rfloor$, requiring for the trajectory to approach a target x_{mid} at the midtime and a target x_{end} at the final time. Here the idea is to optimize the states $x(T/2)$ and $x(T)$ via a DFBBO algorithm and solve via a SO algorithm two smooth subproblems recovering a trajectory joining respectively $x(0)$ to $x(T/2)$ and $x(T/2)$ to $x(T)$, with a minimal Lagrange cost.

Hence the main generic problem that we address in this work is given by

$$
\begin{aligned}
&\underset{\substack{x \in \mathcal{AC}([0,T],\mathbb{R}^n) \\ u \in \mathcal{L}^\infty([0,T],\mathbb{R}^m)}}{\text{minimize}} && g(\{x(t)\}_{t\in\Omega}) + \int_0^T \ell(t, x(t), u(t))\, dt \\
&\text{subject to} && \dot{x}(t) = f(t, x(t), u(t)), \quad \text{a.e. } t \in [0, T], \\
& && x(0) = x_0, \\
& && c(t, x(t), u(t)) \le 0, \qquad \text{a.e. } t \in [0, T], \\
& && c_T(x(T)) \le 0
\end{aligned}
\tag{P_{Generic}}
$$

where the Mayer cost depends on the state of the control system at one or several times (indexed by the set $\Omega \subset [0, T]$ with finite cardinality) and the corresponding Mayer cost function is piecewise constant in a generic sense, that is, without being restricted to a discrete distance to some reference targets. Actually, even if we focus on piecewise constant Mayer cost functions in this paper, our work can be considered for any Mayer cost function presenting singularities such as discontinuities (when involving the floor function $\lfloor \cdot \rfloor$ for example).

The hybrid method proposed in this paper relies on the knowledge of the state variables which may affect the singularities. Precisely, once Problem (P_{Generic}) is fully discretized and seen as a finite-dimensional optimization problem, we reformulate it as a main problem handling the state variables which may affect the singularities, and involving subproblems dealing with all remaining state-control variables. The main problem is solved using a DFBBO algorithm. Each subproblem's goal is to recover a partial feasible trajectory-control pair joining with a minimal partial Lagrange cost two consecutive states proposed by the main DFBBO algorithm. Since the variables affecting the singularities are fixed, the subproblems are smooth discrete optimal control problems with fixed terminal states, solved efficiently via a SO algorithm. Hence the hybrid method applies both DFBBO and SO algorithms,

The goal of this paper is to formally introduce the hybrid method. Section 2 develops a mathematical framework to express precisely the reformulation discussed above, but for a generic singular finite-dimensional optimization problem, allowing to discuss properties of the reformulation in a simple and general context. The specification to discretized optimal control problems is done in Section 3, with details on how the hybrid method works in that context. Section 4 proposes numerical experiments showing that the hybrid method is efficient for optimal control problems with a piecewise

constant Mayer cost function. Finally Section 5 provides an opening discussion on future perspectives and possible improvements of the hybrid method.

Let us observe that the hybrid method has connections with trajectory optimization [22, 23] in a context of driving an aircraft. However, in contrary to the present work, these papers require existence and explicit knowledge of first- and second-order information about the system and the costs. The hybrid method also has connections with bilevel optimization [24, 25] in which the common framework defines an upper level trying to minimize the optimal value of a lower level parameterized by the upper level. However the bilevel paradigm involves a conflictual leader-follower structure, thus it differs from this work which simply partitions the variables.

2 Reformulation of a generic singular optimization problem

This paper proposes a numerical approach dedicated to solve a class of discretized optimal control problems. These problems can be written as finite-dimensional constrained optimization problems of the form

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & \varphi(x) \\ \text{subject to} & c(x) \le 0 \end{array} \tag{$\mathcal{P}$}$$

with $n \ge 1$ real variables $x \in \mathbb{R}^n$, where $\varphi : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, and with $d \ge 1$ inequality constraints expressed as a component-wise vectorial inequality $c(x) \le 0$ via the *constraint function* $c : \mathbb{R}^n \to \mathbb{R}^d$. In some cases, the functions φ or c may be singular (e.g., discontinuous) with only a known subset of variables affecting the singularities. Exploiting this knowledge, this section focuses on the *reformulation* of ($\mathcal{P}$) given by

$$\underset{y \in \mathbb{R}^p}{\text{minimize}} \quad \begin{pmatrix} \underset{z \in \mathbb{R}^q}{\inf} & \varphi(y,z) \\ \text{subject to} & c(y,z) \le 0 \end{pmatrix} \tag{$\mathcal{R}$}$$

where the n real variables $x \in \mathbb{R}^n$ are partitioned into two disjoint groups, being expressed as $x = (y,z)$ and ordered so that $y \in \mathbb{R}^p$ stands for the $p \in [\![0,n]\!]$ variables which may affect the singularities of ($\mathcal{P}$) (called *singular variables*) and $z \in \mathbb{R}^q$ stands for the remaining $q = n - p$ variables (called *smooth variables*). The content in the large parenthesis of ($\mathcal{R}$), depending on $y \in \mathbb{R}^p$, represents the (possibly extended-) real value $\inf\{\varphi(y,z) : z \in \mathbb{R}^q, \ c(y,z) \le 0\}$. The format includes situations where $p = 0$ or $p = n$, but the case of interest is $p \in [\![1, n-1]\!]$ with $n \ge 2$. In view of analysing Reformulation ($\mathcal{R}$), we introduce for any $y \in \mathbb{R}^p$ the *subproblem* ($\mathcal{S}(y)$) given by

$$\begin{array}{ll} \underset{z \in \mathbb{R}^q}{\text{minimize}} & \varphi(y,z) \\ \text{subject to} & c(y,z) \le 0. \end{array} \tag{$\mathcal{S}(y)$}$$

Reformulation ($\mathcal{R}$) is introduced to numerically solve Problem ($\mathcal{P}$) when the latter has singularities (e.g., discontinuities) affected only by a known subset of variables. In that context, solving ($\mathcal{P}$) numerically may be hard since it is a singular optimization problem in which gradients are not available. This falls into the DFBBO framework in which neither explicit knowledge nor existence of derivatives is required. However a DFBBO algorithm is usually less efficient than a SO algorithm using gradients when those are available [14, Section 1.4]. Therefore Reformulation ($\mathcal{R}$) is relevant since it separates the variables affecting the singularities of ($\mathcal{P}$) from the others. Hence, the main idea of the hybrid method introduced in this paper is to numerically solve ($\mathcal{R}$) instead of ($\mathcal{P}$), by optimizing the variables y at the singular unconstrained upper level via a DFBBO algorithm, while the remaining variables z are optimized by solving ($\mathcal{S}(y)$) via a SO algorithm. This approach gives as few variables as possible to a DFBBO algorithm and leaves all the others to a smooth subproblem solved with a more efficient SO algorithm.

The section is organized as follows. First Section 2.1 recalls standard notions, such as the definitions of a *feasible* and a *well-defined* optimization problem. Section 2.2 analyzes relationships existing between ($\mathcal{P}$) and ($\mathcal{R}$). Section 2.3 discusses issues arising in the numerical resolution of ($\mathcal{R}$) via the hybrid method.

2.1 Feasible and well-defined optimization problems

Consider the optimization problem ($\mathcal{P}$). The points $x \in \mathbb{R}^n$ such that $c(x) \leq 0$ form the *feasible set*. A feasible point x^* is said to be an *optimal solution* of ($\mathcal{P}$) if it satisfies $\varphi(x^*) \leq \varphi(x)$ for all feasible points x. In that case $\varphi(x^*)$ is called the *optimal value* of ($\mathcal{P}$).

A *constraint violation function* for ($\mathcal{P}$) quantifies the feasibility or infeasibility of any point $x \in \mathbb{R}^n$. Writing $c(x) = (c_1(x), \ldots, c_d(x))$ for all $x \in \mathbb{R}^n$, the constraint violation function h_c considered in this work, as in [26], is given by

$$h_c : \begin{cases} \mathbb{R}^n & \to & \mathbb{R} \\ x & \mapsto & \sum_{k=1}^{d} \max\{c_k(x), 0\}^2 \end{cases}$$

in which the square allows to avoid introducing additional singularities.

Definition 2.1. For any $0 \leq \varepsilon \leq +\infty$, the *$\varepsilon$-feasible set* of ($\mathcal{P}$) is defined by

$$\Theta_\varepsilon = \{x \in \mathbb{R}^n : h_c(x) \leq \varepsilon\}$$

and Problem ($\mathcal{P}$) is said to be *ε-feasible* (denoted by ε-F) if $\Theta_\varepsilon \neq \emptyset$, while it is said to be *ε-infeasible* otherwise. When ($\mathcal{P}$) is ε-F for some $0 \leq \varepsilon \leq +\infty$, it is said to be *$\varepsilon$-feasible-undefined* if $\varphi(\Theta_\varepsilon)$ is not bounded below, and *ε-feasible-defined* (ε-FD) if $\varphi(\Theta_\varepsilon)$ is bounded below, and *ε-feasible-well-defined* (ε-FWD) if $\varphi(\Theta_\varepsilon)$ is bounded below and attains its lower bound.

In the sequel the ε will be omitted in the acronyms of Definition 2.1 when it equals 0 (e.g., a FWD problem is a 0-FWD problem). Note that ($\mathcal{P}$) has an optimal solution if and only if ($\mathcal{P}$) is FWD.

Definition 2.2. When it exists, the *minimal infeasibility* of ($\mathcal{P}$) is defined as the smallest value $0 \leq \varepsilon \leq +\infty$ for which the set Θ_ε is nonempty.

Note that ($\mathcal{P}$) has a minimal infeasibility if and only if the optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad h_c(x) \tag{$\mathcal{H}$}$$

is FWD. If so, the optimal value of ($\mathcal{H}$) equals the minimal infeasibility of ($\mathcal{P}$).

Remark 2.3. Consider Problem ($\mathcal{P}$). The set Θ_0 is the feasible set, $\Theta_\infty = \mathbb{R}^n$, and there necessarily exists $\varepsilon \geq 0$ such that $\Theta_\varepsilon \neq \emptyset$. However ($\mathcal{P}$) may be infeasible with no minimal infeasibility. For example, with $c : x \in \mathbb{R} \mapsto e^x \in \mathbb{R}$ and $\varphi \equiv 0$, there is no smallest value $0 \leq \varepsilon \leq +\infty$ for which $\Theta_\varepsilon \neq \emptyset$.

The concepts introduced above for Problem ($\mathcal{P}$) are extended to Reformulation ($\mathcal{R}$) and to Sub-problems ($\mathcal{S}(y)$), replacing h_c by $h_{c(y,\cdot)}$, for all $y \in \mathbb{R}^p$.

2.2 Relationships between Problem ($\mathcal{P}$) and Reformulation ($\mathcal{R}$)

The equality

$$\begin{pmatrix} \underset{x \in \mathbb{R}^n}{\inf} & \varphi(x) \\ \text{subject to} & c(x) \leq 0 \end{pmatrix} = \underset{y \in \mathbb{R}^p}{\inf} \begin{pmatrix} \underset{z \in \mathbb{R}^q}{\inf} & \varphi(y, z) \\ \text{subject to} & c(y, z) \leq 0 \end{pmatrix}$$

holds true, even when the infima are not attained, or even when they are equal to $\pm\infty$. Hereafter we investigate when the infima are attained (and thus are minima) and what relations can be obtained

between the minimizers. The next proposition (whose proof follows directly from Definition 2.1) establishes an equivalence result on the FWD nature of ($\mathcal{P}$) and the FWD nature of ($\mathcal{R}$), and, if so, derives some relations between their respective optimal solutions.

Proposition 2.4. The following statements hold:

1. If ($\mathcal{P}$) is FWD, then, for any optimal solution $x^* = (y^*, z^*)$ of ($\mathcal{P}$), y^* is an optimal solution of ($\mathcal{R}$) (which is FWD) and z^* is an optimal solution of $\mathcal{S}(y^*)$ (which is FWD).

2. If ($\mathcal{R}$) is FWD and has an optimal solution y^* for which $\mathcal{S}(y^*)$ is FWD, then, for any optimal solution z^* of $\mathcal{S}(y^*)$, $x^* = (y^*, z^*)$ is an optimal solution of ($\mathcal{P}$) (which is FWD).

The second item of Proposition 2.4 guarantees that we can recover an optimal solution of ($\mathcal{P}$) from an optimal solution y^* of ($\mathcal{R}$), provided that $\mathcal{S}(y^*)$ is FWD. The next counterexamples illustrate this requirement, with two different obstructions. In these examples (and the others of this section), note that the singularity affecting the variable y is not mandatory but allows to remain consistent with our setting.

Example 2.5. Consider $\varphi : (y, z) \in \mathbb{R}^2 \mapsto e^{|y|+z} \in \mathbb{R}$ and $c \equiv 0$. In that context ($\mathcal{R}$) is FWD and any $y \in \mathbb{R}$ is an optimal solution of ($\mathcal{R}$). However, for any $y \in \mathbb{R}$, ($\mathcal{S}(y)$) is FD but not FWD. Note that ($\mathcal{P}$) is FD, but is not FWD.

Example 2.6. Consider $\varphi : (y, z) \in \mathbb{R}^2 \mapsto |y| \, e^z + |y - 1| \, z^2 \in \mathbb{R}$ and $c \equiv 0$. In that context ($\mathcal{P}$) is FWD and its unique optimal solution is $x^* = (0, 0)$. Furthermore ($\mathcal{R}$) is FWD and has two optimal solutions $y_1^* = 0$ and $y_2^* = 1$. Note that $\mathcal{S}(y_1^*)$ is FWD (with $z_1^* = 0$ an optimal solution), while $\mathcal{S}(y_2^*)$ is FD but not FWD.

Proposition 2.4 provides only a theoretical result establishing relations between the optimal solution $x^* = (y^*, z^*)$ of ($\mathcal{P}$), the optimal solution y^* of ($\mathcal{R}$) and the optimal solution z^* of $\mathcal{S}(y^*)$. These relations rely in particular on the FWD nature of $\mathcal{S}(y^*)$. Note that, in some cases, ($\mathcal{S}(y)$) may be FWD if and only if $y = y^*$ (see Example 2.7). It is also possible that ($\mathcal{S}(y)$) is infeasible for some $y \in \mathbb{R}^p$ arbitrarily close to y^* (see Example 2.8).

Example 2.7. Consider $\varphi : (y, z) \in \mathbb{R}^2 \mapsto |y| \, e^z \in \mathbb{R}$ and $c \equiv 0$. In that context ($\mathcal{P}$) has an optimal solution $x^* = (y^*, z^*) = (0, 0)$. However, when $y \neq y^*$, $\mathcal{S}(y)$ is FD but is not FWD.

Example 2.8. Consider $\varphi : (y, z) \in \mathbb{R}^2 \mapsto |y| + z^2 \in \mathbb{R}$ and $c : (y, z) \in \mathbb{R}^2 \mapsto (1 - y - z, 1 - y + z) \in \mathbb{R}^2$. In that context ($\mathcal{P}$) has an optimal solution $x^* = (y^*, z^*) = (1, 0)$, while ($\mathcal{S}(y)$) is infeasible for any $y < 1$.

Examples 2.7 and 2.8 show that, when looking for a (numerical) optimal solution of ($\mathcal{R}$), we may be confronted to subproblems ($\mathcal{S}(y)$) that are FD but not FWD, or even infeasible. These situations have to be addressed carefully because any attempt to evaluate the objective value of ($\mathcal{R}$) at some $y \in \mathbb{R}^p$ requires, roughly speaking, a (numerical) optimal value of ($\mathcal{S}(y)$). A FD subproblem ($\mathcal{S}(y)$) is not hard to handle, as we may obtain a feasible point $z \in \mathbb{R}^q$ with objective value arbitrarily close to the infimum, but an infeasible subproblem ($\mathcal{S}(y)$) is more problematic. The next section discusses a dedicated numerical process to face such an infeasibility. It relies on the next proposition, whose proof is simple and left to the reader.

Proposition 2.9. If $h_{c(y,\cdot)}$ admits a minimum for all $y \in \mathbb{R}^p$, then $\mathcal{S}(y)$ has a minimal infeasibility, denoted by μ_y, for all $y \in \mathbb{R}^p$. If moreover ($\mathcal{P}$) is ∞-FD, then $\mathcal{S}(y)$ is μ_y-FD for all $y \in \mathbb{R}^p$.

2.3 Issues concerning the numerical resolution of Reformulation ($\mathcal{R}$)

In the hybrid method, the objective function of ($\mathcal{R}$) is treated as a *blackbox*, that is, as a function with intractable expression which requires here, roughly speaking, to solve a subproblem ($\mathcal{S}(y)$) depending on the input $y \in \mathbb{R}^p$. This blackbox has to be handled carefully since two main issues (discussed and illustrated in Section 2.2) need to be addressed. First ($\mathcal{S}(y)$) may be infeasible and, second, denoting by μ_y its minimal infeasibility (when it exists), ($\mathcal{S}(y)$) may be μ_y-FD but not μ_y-FWD. Assuming that

the conditions expressed in Proposition 2.9 are satisfied, the following two-phases numerical process is proposed to handle these drawbacks:

(i) Solve with a SO algorithm the smooth unconstrained FWD problem

$$\underset{z \,\in\, \mathbb{R}^q}{\text{minimize}} \quad h_{c(y,\cdot)}(z). \tag{$\mathcal{H}_y$}$$

Denote by $\tilde{z}_y$ a (numerical) optimal solution of $(\mathcal{H}_y)$ and get $\tilde{\mu}_y = h_{c(y,\cdot)}(\tilde{z}_y)$ as (numerical) approximation of the minimal infeasibility of $(\mathcal{S}(y))$.

(ii) Solve with a SO algorithm the smooth constrained FD problem

$$\begin{aligned} \underset{z \,\in\, \mathbb{R}^q}{\text{minimize}} \quad & \varphi(y, z) \\ \text{subject to} \quad & h_{c(y,\cdot)}(z) \le \tilde{\mu}_y \end{aligned} \tag{$\tilde{\mathcal{S}}_y$}$$

from the starting point $\tilde{z}_y$. Denote by $\tilde{z}_y^*$ the (numerical) optimal solution of $(\tilde{\mathcal{S}}_y)$ and get $\varphi(y, \tilde{z}_y^*)$ as (numerical) approximation of the optimal value of $(\tilde{\mathcal{S}}_y)$. In practice it is not restrictive that $(\tilde{\mathcal{S}}_y)$ may be not FWD, since it is FD and solved up to some numerical thresholds. Return $\tilde{\mu}_y$ and $\varphi(y, \tilde{z}_y^*)$.

The goal of the above two-phases process is the following. If $(\mathcal{S}(y))$ is feasible for a given $y \in \mathbb{R}^p$, then the first step leads to a numerically feasible solution of $(\mathcal{S}(y))$, the second step numerically solves $(\mathcal{S}(y))$ and the process returns a numerically optimal value of $(\mathcal{S}(y))$. Otherwise, if $(\mathcal{S}(y))$ is infeasible, then an arbitrary choice has to be made to assign a value to the blackbox at y. We could set the value $+\infty$, but such a choice would not provide any information to pursue efficiently the numerical resolution of $(\mathcal{R})$. Instead, via the above process, we recover an infeasibility measure, as well as a corresponding finite objective value. Recall that DFBBO algorithms can exploit these two (possibly conflictual) metrics. For example the MADS algorithm [16] implemented in NOMAD [17] handles them via its *progressive barrier* formulation [27]. It constructs an approximated Pareto front of undominated infeasible solutions on a diagram showing objective value versus infeasibility. The front is truncated to eliminate solutions with an infeasibility above a threshold decreasing with the iterations. Each iteration of MADS starts from the solution with the lowest objective function value among the truncated front, so that the algorithm eventually finds a feasible solution with a low objective value. Example 2.10 illustrates the relevancy of this approach.

Example 2.10. Consider Example 2.8 and an attempt to numerically solve $(\mathcal{R})$ from the initial guess $y = 0$. Subproblem $(\mathcal{S}(y))$ is infeasible for any $y < 1$ and the above numerical two-phases process returns $\tilde{\mu}_y = 2(1 - y)^2$ and $\varphi(y, \tilde{z}_y^*) = y$. Hence any value $0 < y \le 1$ has an higher objective function value than $y = 0$ but a lower infeasibility. An algorithm solving $(\mathcal{R})$ can be driven to $y = 1$, leading to a feasible subproblem $(\mathcal{S}(y))$, via an iterative reduction of the maximal admissible infeasibility. Then it eventually reaches $x = (y, \tilde{z}_y^*) = (1, 0)$ which is the optimal solution of $(\mathcal{P})$.

3   Numerical setup to solve Problem $(\mathrm{P_{Generic}})$

A full discretization of Problem $(\mathrm{P_{Generic}})$, using a Runge-Kutta method for example, can be written as

$$\begin{aligned} \underset{\substack{\{x_k\}_{k=1}^N \subset \mathbb{R}^n, \\ \{u_k\}_{k=0}^{N-1} \subset \mathbb{R}^m}}{\text{minimize}} \quad & g\left(\{x_k\}_{k \in \Omega}\right) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k) \\ \text{subject to} \quad & x_{k+1} = f_k(x_k, u_k), \quad k \in [\![0, N-1]\!], \\ & c_k(x_k, u_k) \le 0, \qquad k \in [\![0, N-1]\!], \\ & c_N(x_N) \le 0 \end{aligned} \tag{P}$$

where $n \ge 1$, $m \ge 1$ and $N \ge 1$ are fixed positive integers, $\{x_k\}_{k=1}^N \subset \mathbb{R}^n$ are the *state variables* and $\{u_k\}_{k=0}^{N-1} \subset \mathbb{R}^m$ are the *control variables*, $x_0 \in \mathbb{R}^n$ is the fixed initial state, $\Omega \subseteq [\![1, N]\!]$ is the

set (with $\omega \geq 0$ elements) of the indices k such that the piecewise constant *Mayer cost function* $g : (\mathbb{R}^n)^\omega \to \mathbb{R}$ actually depends on x_k, each $\ell_k : \mathbb{R}^n \to \mathbb{R}$ is the *Lagrange cost function at time k*, each $f_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the *dynamics at time k*, each $c_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{d_k}$ (with $d_k \geq 1$) is the *mixed state-control constraints function at time k* and $c_N : \mathbb{R}^n \to \mathbb{R}^{d_N}$ (with $d_N \geq 1$) stands for the *final state constraints function*.

Note that Problem (P) is a specific instance of Problem ($\mathcal{P}$). Therefore, following the approach developed in Section 2, Section 3.1 proposes several possible reformulations of (P) to handle its singularities. Then Section 3.2 provides a numerical setup to introduce the DFBBO-SO hybrid method based on these reformulations, and to compare it with two non-hybrid approaches (one based on a SO algorithm exclusively, and one based on a DFBBO algorithm exclusively). Some numerical experiments, showing the efficiency of the hybrid method for optimal control problems with a piecewise constant Mayer cost function, are provided in the next Section 4.

3.1   Several reformulations of Problem (P)

Take $(i,j) \in [\![0,N]\!]^2$ with $j > i$ and $(x_i, x_j) \in (\mathbb{R}^n)^2$. The search of a trajectory linking x_i at time i to x_j at time j, with minimal partial Lagrange cost and satisfying the state-control constraints of Problem (P), leads to the subproblem given by

$$
\begin{aligned}
&\underset{\substack{\{x_k\}_{k=i+1}^{j-1} \subset \mathbb{R}^n, \\ \{u_k\}_{k=i}^{j-1} \subset \mathbb{R}^m}}{\text{minimize}} && \sum_{k=i}^{j-1} \ell_k(x_k, u_k) \\
&\text{subject to} && x_{k+1} = f_k(x_k, u_k), \quad k \in [\![i, j-1]\!], \\
& && c_k(x_k, u_k) \leq 0, \qquad k \in [\![i, j-1]\!].
\end{aligned}
\tag{S(i, x_i, j, x_j)}
$$

This subproblem is not always FWD in general. Hereafter the (possibly extended-real) lower bound of its objective function over its feasible set is denoted by $\underline{S}(i, x_i, j, x_j)$.

Now, following the approach developed in Section 2 and using the notation introduced above, Problem (P) has the reformulation

$$
\begin{aligned}
&\underset{\{x_k\}_{k=1}^{N} \subset \mathbb{R}^n}{\text{minimize}} && g\left(\{x_k\}_{k \in \Omega}\right) + \sum_{k=0}^{N-1} \underline{S}(k, x_k, k+1, x_{k+1}) \\
&\text{subject to} && c_N(x_N) \leq 0
\end{aligned}
\tag{R$^{[\![1,N]\!]}$}
$$

where the variables are the entire trajectory, instead of the entire trajectory-control pair in Problem (P). For any given trajectory $\{x_k\}_{k=1}^{N}$, the search of a control $\{u_k\}_{k=0}^{N-1}$ driving x_0 along this trajectory is implicit. The search of each control u_k linking x_k to x_{k+1}, while minimizing $\ell_k(x_k, u_k)$ and satisfying the constraint $c_k(x_k, u_k) \leq 0$, is done in the smooth Subproblem (S$(k, x_k, k+1, x_{k+1})$). However there is still an important number of variables (precisely nN) left to Reformulation (R$^{[\![1,N]\!]}$).

It is possible to reduce this number of variables. The variant

$$
\begin{aligned}
&\underset{x_N \in \mathbb{R}^n}{\text{minimize}} && g(x_N) + \underline{S}(0, x_0, N, x_N) \\
&\text{subject to} && c_N(x_N) \leq 0
\end{aligned}
\tag{R$^{\{N\}}$}
$$

has only the final state x_N as a variable. For any given $x_N \in \mathbb{R}^n$, the search of a control linking x_0 to x_N, with minimal Lagrange cost and satisfying the state-control constraints of Problem (P), is left to the unique subproblem (S$(0, x_0, N, x_N)$).

When N is even and $\Omega \subseteq \{N/2, N\}$, another variant in which the variables are the final state x_N and the midterm state $x_{N/2}$ is possible. In that case there are two subproblems. The first one links x_0 to $x_{N/2}$ and the second one links $x_{N/2}$ to x_N. Denoting by $\lambda_0 = 0$, $\lambda_1 = N/2$, $\lambda_2 = N$, the

corresponding reformulation is given by

$$\begin{array}{ll} \underset{\{x_{\lambda_1},\, x_{\lambda_2}\}\, \subset\, \mathbb{R}^n}{\text{minimize}} & g\left(\{x_k\}_{k\in\Omega}\right) + \sum_{i=0}^{1} \underline{S}\left(\lambda_i, x_{\lambda_i}, \lambda_{i+1}, x_{\lambda_{i+1}}\right) \\ \text{subject to} & c_N(x_{\lambda_2}) \le 0. \end{array} \qquad (\text{R}^{\{N/2,N\}})$$

Finally, from a general point of view, it is possible to choose *a priori* which states remain variables of the reformulated problem. To this aim, take $r \in [\![1,N]\!]$ and $\Lambda = \{\lambda_1, \ldots, \lambda_r\} \subseteq [\![1,N]\!]$ the set of all indices of the states given as variables to the reformulated problem, with increasing values and satisfying $\lambda_r = N$ and $\Omega \subseteq \Lambda$. Taking $\lambda_r = N$ is mandatory because the constraint $c_N(x_N) \le 0$ cannot appear in the last subproblem, and taking $\Omega \subseteq \Lambda$ is required because the Mayer cost cannot appear in any subproblem. Define also for convenience $\lambda_0 = 0$. Finally the general reformulation of Problem (P) is given by

$$\begin{array}{ll} \underset{\{x_{\lambda_i}\}_{i=1}^{r}\, \subset\, \mathbb{R}^n}{\text{minimize}} & g\left(\{x_k\}_{k\in\Omega}\right) + \sum_{i=0}^{r-1} \underline{S}\left(\lambda_i, x_{\lambda_i}, \lambda_{i+1}, x_{\lambda_{i+1}}\right) \\ \text{subject to} & c_N(x_N) \le 0 \end{array} \qquad (\text{R}^\Lambda)$$

with nr variables and admitting r subproblems, each trying to find a control linking x_{λ_i} to $x_{\lambda_{i+1}}$ with a minimal partial Lagrange cost while satisfying the state-control constraints of Problem (P). Note that Reformulations ($\text{R}^{[\![1,N]\!]}$), ($\text{R}^{\{N\}}$) and ($\text{R}^{\{N/2,N\}}$) are particular instances of Reformulation (R^Λ).

The choices of $r \in [\![1,N]\!]$ and $\Lambda = \{\lambda_1, \ldots, \lambda_r\} \subseteq [\![1,N]\!]$ follow the knowledge of which states x_k may affect the singularities of Problem (P). Furthermore note that the smaller r is, then the smaller the number of variables left to Reformulation (R^Λ) solved with a DFBBO algorithm is. In that context note that (P) and (R^Λ) are specific instances of ($\mathcal{P}$) and ($\mathcal{R}$) from Section 2. Indeed the p singular variables are the state variables indexed by Λ and, for any input $y = (x_{\lambda_i})_{i=1}^p$, Reformulation ($\mathcal{R}$) has a (unique) Subproblem ($\mathcal{S}(y)$) obtained by aggregating all independent subproblems ($\text{S}(\lambda_i, x_{\lambda_i}, \lambda_{i+1}, x_{\lambda_{i+1}})$) together.

3.2   Numerical setup

In the next Section 4, numerical simulations will be performed on examples of Problem (P) in which the Mayer cost function involves the floor function $\lfloor \cdot \rfloor$, causing discontinuities. This section describes three numerical methods that will be implemented and compared. The first one is a SO method, the second one is a DFBBO method and the third one is the DFBBO-SO hybrid method based on the reformulations introduced in Section 3.1. Before coming to these methods, we introduce first a simple initialization used for each method.

Uncontrolled trajectory $\mathcal{T}_{\text{uncontrolled}}$ This trajectory results from the induction $x_{k+1} = f_k(x_k, u_k)$ for all $k \in [\![0, N-1]\!]$, with each control u_k set to zero. It is used as initialization of the three numerical methods introduced below.

The $\mathcal{M}_{\text{SO}}$ method The purpose of the $\mathcal{M}_{\text{SO}}$ method is to solve Problem (P) by using a SO algorithm, via the JuMP modeling language [28] and the IPOPT solver, despite the discontinuities due to the floor function $\lfloor \cdot \rfloor$. Unfortunately, IPOPT cannot deal with discontinuous problems. To overcome this obstacle, we use a smooth approximation of the floor function (see Remark 3.1 below) which leads to a smoothed version of (P) accepted by the code. However the $\mathcal{M}_{\text{SO}}$ method remains likely inefficient. Indeed, even after a smoothing, the plateaus resulting from the floor function remain and lead to partial derivatives close to zero. Hence, the $\mathcal{M}_{\text{SO}}$ method may fail to detect the significant gains on the objective value it would obtain by making some variables cross the floor function plateaus. The presumed ineffectiveness of the $\mathcal{M}_{\text{SO}}$ method is confirmed by the numerical results in Section 4.

The $\mathcal{M}_{\text{DFBBO}}$ method The $\mathcal{M}_{\text{DFBBO}}$ method aims to solve Problem (P) thanks to the DFBBO algorithm MADS via the JuMP interface of the NOMAD solver [29]. However NOMAD is usually restricted

to small-scale problems [14, Section 1.4] while (P) usually has a large scale. Fortunately a dimension reduction is possible, as we can replace all state variables x_k by the trajectory generated by the control variables u_k and the equality constraints $x_{k+1} = f_k(x_k, u_k)$. However, since each control u_k remains a variable, the scale of the reduced problem remains too large for NOMAD, in which a critical value of 50 is indicated as the maximal number of variables that can be easily handled [17]. Hence, the $\mathcal{M}_{\mathrm{DFBBO}}$ method is likely inefficient, which is confirmed by the numerical results obtained in Section 4. Note that, for fair comparison with the $\mathcal{M}_{\mathrm{SO}}$ method, the $\mathcal{M}_{\mathrm{DFBBO}}$ method is executed in Section 4 on the smoothed version of Problem (P), in addition to the above dimension reduction.

The $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method The $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method is the main contribution of the present paper. Instead of numerically solving Problem (P), the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method focuses on Reformulation (R^{Λ}) introduced in Section 3.1 and numerically solves it via NOMAD, while its objective function is evaluated by solving the smooth subproblems $(S(\lambda_i, x_{\lambda_i}, \lambda_{i+1}, x_{\lambda_{i+1}}))$ via IPOPT. This evaluation relies on the numerical two-phases process presented in Section 2.3, where the infeasibility and the corresponding finite objective value are returned to NOMAD to be used in its implementation of the progressive barrier variant of the MADS algorithm [27]. Unlike the $\mathcal{M}_{\mathrm{SO}}$ and $\mathcal{M}_{\mathrm{DFBBO}}$ methods, the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ applies each solver to a problem it is designed for. Indeed NOMAD solves a small-scale blackbox optimization problem with complex internal structure, while IPOPT solves smooth optimization problems. Note that, for fair comparison with the two methods above, the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method is executed in Section 4 on the reformulation of a smoothed version of Problem (P), and not on the reformulation of (P). Also, as in Section 4 the examples have singularities affected by the state variables x_N and $x_{N/2}$ only, the two variants $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ and $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2,N\}}$ of the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method will be tested.

Remark 3.1. As IPOPT cannot handle discontinuities from the floor function $\lfloor \cdot \rfloor$, a smooth approximation is introduced as follows. First, note that the Heaviside function $\mathbb{1}_{\mathbb{R}_+}(x) = 0$ if $x < 0$ and $\mathbb{1}_{\mathbb{R}_+}(x) = 1$ if $x \geq 0$, can be approximated by $\mathbb{1}_{\mathbb{R}^+}(x) \approx exp(-e^{-x})$ for any $x \in \mathbb{R}$. Strengthening the approximation by a factor $\tau \in \mathbb{R}_+^*$ and shifting the discontinuity from 0 to any value $q \in \mathbb{R}$, we obtain

$$\forall (\tau, q) \in \mathbb{R}_+^* \times \mathbb{R}, \quad \forall x \in \mathbb{R}, \quad \mathbb{1}_{[q,+\infty[}(x) \approx exp\left(-e^{-\tau(x-q)}\right).$$

Hence a smooth approximation of $\lfloor \cdot \rfloor$ can be derived on any interval $[a, b] \subset \mathbb{R}$, with $(a, b) \in \mathbb{Z}^2$, $a < b$, and with any strengthening factor $\tau \in \mathbb{R}_+^*$, as follows:

$$\lfloor \cdot \rfloor_{\tau;a,b} : \left\{ \begin{array}{ccl} [a,b] & \to & [a,b] \\ x & \mapsto & a + \displaystyle\sum_{q=a+1}^{b-1} exp\left(-e^{-\tau(x-q)}\right). \end{array} \right. \qquad (\lfloor \cdot \rfloor_{\tau;a.b})$$

Note that $\lfloor \cdot \rfloor_{\tau;a.b}$ is of class $\mathcal{C}^{\infty}$ for any admissible couple $(\tau; a, b)$. Its derivative is close to 0 at any x far from $\mathbb{Z}$. Also note that $\lfloor \cdot \rfloor_{\tau;a.b}$ is not constant on the intervals $[q, q+1)$ with $q \in \mathbb{Z}$, and it does not reach integer values when $x \in \mathbb{Z}$. Figure 3 shows $\lfloor \cdot \rfloor$, as well as some approximations $\lfloor \cdot \rfloor_{\tau;a.b}$ for some values of τ, a and b. The numerical tests presented in Section 4 are performed with $\tau = 75$.
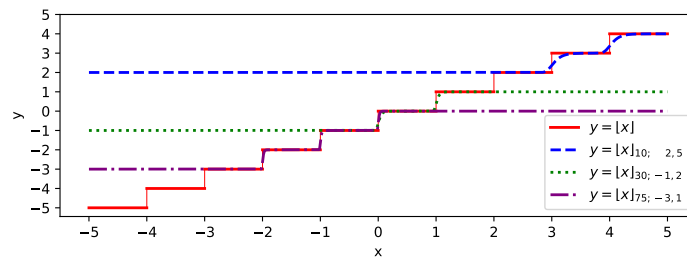

Figure 3: Floor function $\lfloor \cdot \rfloor$ and smooth approximations $\lfloor \cdot \rfloor_{\tau;a.b}$.

4 Computational study

The examples studied in this section are of the form of Problem (P_{Generic}) where the Mayer cost function involves the floor function $\lfloor \cdot \rfloor$, causing discontinuities. We discretize the timelapse $[0, T]$ (with $T > 0$) with a constant timestep T/N (with $N \in \mathbb{N}^*$), the control system via the standard fourth-order Runge-Kutta method [11] and the integral Lagrange cost via the left rectangle rule. This discretization procedure leads to Problem (P) which preserves the singularities due to the floor function. We implement and compare the numerical methods introduced in Section 3.2 in the *Julia language* using the modeling tool JuMP [28]. Both Ipopt and Nomad [29] are interfaced with JuMP. As explained in Section 3.2, for fair comparisons between the numerical methods, Problem (P) is preconditioned using the smoothing procedure described in Remark 3.1.

This section is organized as follows. Section 4.1 deals with a controlled harmonic oscillator, where the Mayer cost involves the final kinetic energy minus a truncation of the final potential energy. Section 4.2 discusses a Zermelo-type problem representing a boat on a river with a stream, trying to reach a relevant parking in an harbour. Section 4.3 addresses a similar Zermelo-type problem but with two targets to reach, one at the midtime $T/2$ and one at the final time T. Finally Section 4.4 deals with a truncated version of the Lotka-Volterra stabilization problem and shows that the best choice of Λ in the $\mathcal{M}_{\text{hybrid}}^{\Lambda}$ method may be nontrivial.

Notation Hereafter $\mathcal{AC}([0, T], \mathbb{R}^n)$ denotes the space of absolutely continuous functions defined on $[0, T]$ with values in $\mathbb{R}^n$ and $\mathcal{L}^\infty([0, T], \mathbb{R}^m)$ denotes the usual Lebesgue space of essentially bounded control functions defined almost everywhere on $[0, T]$ with values in $\mathbb{R}^m$. The acronym *a.e.* stands for *for almost every*. In view of using different weighted Euclidean norms in Sections 4.3 and 4.4, the notation $\|X\|_\alpha$ stands for the Euclidean norm of the vector $(\alpha_1 X_1, \ldots, \alpha_n X_n)$ for all $X, \alpha \in \mathbb{R}^n$.

4.1 Truncated variant of the harmonic oscillator problem

Consider

$$
\begin{aligned}
&\underset{\substack{(x, v) \in \mathcal{AC}([0, T], \mathbb{R}^2) \\ u \in \mathcal{L}^\infty([0, T], \mathbb{R})}}{\text{minimize}} && \frac{M}{2} v(T)^2 - \frac{K}{2} \lfloor x(T) \rfloor^2 + \frac{1}{T} \int_0^T u(t)\, dt \\
&\text{subject to} && \dot{x}(t) = v(t), && \text{a.e. } t \in [0, T], \\
& && \dot{v}(t) = \tfrac{1}{M}(-Kx(t) + u(t)), && \text{a.e. } t \in [0, T], \\
& && x(0) = 1, \\
& && v(0) = 0, \\
& && u(t) \in [0, 1], && \text{a.e. } t \in [0, T]
\end{aligned}
\tag{$\mathcal{HO}$}
$$

where the control system mimics a spring-mass system $M\ddot{x}(t) + Kx(t) = u(t)$, $x(t)$ and $v(t)$ denote the position and velocity of a point of mass $M > 0$ attached to a spring of constant $K > 0$, and where $u(t) \in [0, 1]$ denotes a constrained external force. Here the objective is to minimize the (smooth) final kinetic energy $\frac{M}{2} v(T)^2$, plus the average control value $\frac{1}{T} \int_0^T u(t)\, dt$, minus the (singular) final truncated potential energy $\frac{K}{2} \lfloor x(T) \rfloor^2$. In this example, note that the Mayer cost function is not piecewise constant, but involves the floor function $\lfloor \cdot \rfloor$, causing discontinuities.

Figure 4 reports the numerical results obtained with the methods introduced in Section 3.2 on Problem ($\mathcal{HO}$), with the parameters $T = 60$, $N = 600$, $M = 2$ and $K = 1/2$. They follow our expectations from Section 3.2. Among the listed methods, only the $\mathcal{M}_{\text{hybrid}}^{\Lambda}$ method performs well. The $\mathcal{M}_{\text{SO}}$ method fails to detect the gain resulting from a large final spring elongation, and the $\mathcal{M}_{\text{DFBBO}}$ method fails because Nomad faces too many variables. Both methods $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ and $\mathcal{M}_{\text{hybrid}}^{\{N/2, N\}}$ succeed because they use Nomad with few variables to detect a relevant final state and Ipopt to recover a trajectory joining it.
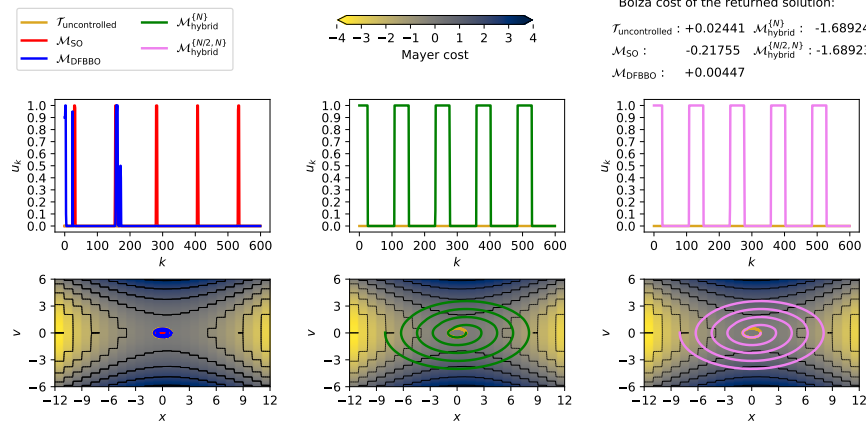
Figure 4: Numerical results obtained on Problem ($\mathcal{HO}$). The $\mathcal{M}_{\mathrm{SO}}$ and $\mathcal{M}_{\mathrm{DFBBO}}$ methods are both represented on the left graph and remain close to their initialization. Both are outperformed by the $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ and $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2,N\}}$ methods.

4.2   A first Zermelo-type problem

In this section we consider a variant of the Zermelo problem, with a piecewise constant Mayer cost function, given by

$$
\begin{aligned}
& \underset{\substack{(x,y)\,\in\,\mathcal{AC}([0,T],\mathbb{R}^2) \\ (u,\theta)\,\in\,\mathcal{L}^\infty([0,T],\mathbb{R}^2)}}{\text{minimize}} && -\left\lfloor \frac{x(T)}{10} \right\rfloor + \frac{1}{T}\int_0^T u(t)\,dt \\
& \text{subject to} && \dot{x}(t) = s_1(x(t),y(t)) + u(t)\cos\theta(t), \quad \text{a.e. } t\in[0,T], \\
&&& \dot{y}(t) = s_2(x(t),y(t)) + u(t)\sin\theta(t), \quad \text{a.e. } t\in[0,T], \\
&&& x(0) = 0, \\
&&& y(0) = 0, \\
&&& 0 \le y(t) \le y_{\max}, \qquad\qquad\qquad\quad \forall t\in[0,T], \\
&&& y(T) = y_{\max}, \\
&&& u(t)\in[0,1], \qquad\qquad\qquad\qquad\ \text{a.e. } t\in[0,T]
\end{aligned}
$$
$$(\mathcal{Z}_1)$$

where $(x(t),y(t))$ is the position of a boat on a river $\mathbb{R}\times[0,y_{\max}]$ of width $y_{\max}>0$, navigating from $(0,0)$ and trying to reach the upper shore $\mathbb{R}\times\{y_{\max}\}$. The boat is equipped with a motor delivering a driving force $u(t)(\cos\theta(t),\sin\theta(t))$ with controllable norm $u(t)\in[0,1]$ and direction $\theta(t)\in\mathbb{R}$. The river has a stream $s=(s_1,s_2)$ (defined in Figure 5). Note that the stream and the driving force actually do not represent physical forces. Indeed the stream is a vector field of velocities and the control $(u(t),\theta(t))$ induces an additive term to the velocity of the boat. There are parking lots of width 10 in the upper shore. The objective is to reach the rightmost parking lot at the final time, by taking into account the average value of $u(t)$.

Figure 6 reports the numerical results obtained with the methods introduced in Section 3.2 on Problem ($\mathcal{Z}_1$), with the parameters $T=40$, $N=400$ and $y_{\max}=6$. These results follow our expectations from Section 3.2. Recall that all methods start from the uncontrolled trajectory which is infeasible since $y_N\neq y_{\max}$ (see Figure 5). The $\mathcal{M}_{\mathrm{SO}}$ first generates a feasible trajectory (with $y_N=6$) but with a high objective value (x_N is far on the negative values). Then, it struggles to optimize it because the Mayer cost is constant on each interval $[10(q-1),10q]$ with $q\in\mathbb{Z}$ (up to slight variations due to the smoothing, see Remark 3.1). This causes the $\mathcal{M}_{\mathrm{SO}}$ method to converge to a local optimum. The $\mathcal{M}_{\mathrm{DFBBO}}$ method fails to find a feasible solution. Both the $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ and $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2,N\}}$ methods recover a feasible trajectory with a low objective function value. The $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ and $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2,N\}}$ methods both outperform the other methods by a significant amount.
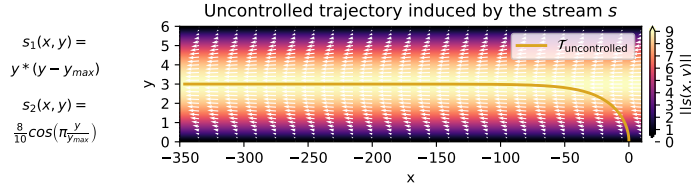
Figure 5: The stream $s = (s_1, s_2) : \mathbb{R} \times [0, y_{\max}] \to \mathbb{R}^2$ with $y_{\max} = 6$. **It repels the boat from the shores. The uncontrolled trajectory (with the motor turned off, that is, with $(u, \theta) \equiv (0, 0)$) goes to $x \to -\infty$, $y \to \frac{y_{\max}}{2}$. The horizontal component s_1 of the stream is powerful in the middle of the river and cannot be overcame by the driving force of the boat delivered by the motor.**
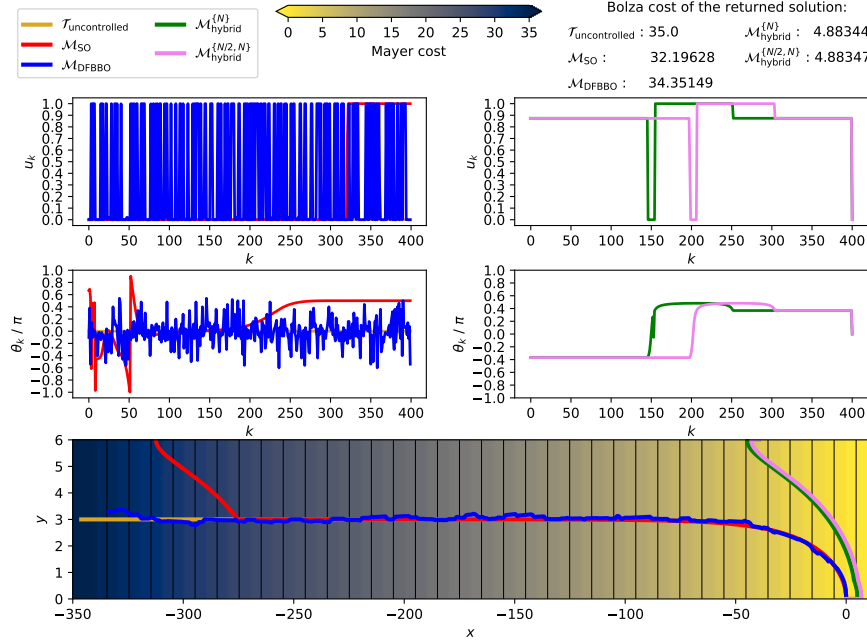

Figure 6: Numerical results obtained on Problem $(\mathcal{Z}_1)$. The $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ and $\mathcal{M}_{\text{hybrid}}^{\{N/2,N\}}$ methods perform well. The $\mathcal{M}_{\text{SO}}$ method successfully reaches the feasible set but remains in a locally optimal solution, and the $\mathcal{M}_{\text{DFBBO}}$ method fails to find a feasible trajectory.

4.3   A second Zermelo-type problem

In this section our objective is to highlight the expected superiority of the $\mathcal{M}_{\text{hybrid}}^{\{N/2,N\}}$ method over the $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ method when considering a problem with two targets A and B to approach (with respect to a discrete distance), one at the final time T and one at the midtime $T/2$. To this aim we consider a second variant of the Zermelo problem given by

$$\begin{aligned}
&\underset{\substack{X \in \mathcal{AC}([0,T], \mathbb{R}^2) \\ (u,\theta) \in \mathcal{L}^\infty([0,T], \mathbb{R}^2)}}{\text{minimize}} && \lfloor \|X(T) - A\|_\alpha \rfloor + \lfloor \|X(\tfrac{T}{2}) - B\|_\beta \rfloor + \frac{1}{T} \int_0^T u(t)\, dt \\
&\text{subject to} && \dot{x}(t) = s_1(X(t)) + u(t)\cos\theta(t), \quad \text{a.e. } t \in [0,T], \\
& && \dot{y}(t) = s_2(X(t)) + u(t)\sin\theta(t), \quad \text{a.e. } t \in [0,T], \\
& && X(0) = (0,0), \\
& && X(t) \in \mathbb{R} \times [0, y_{\max}], \qquad\qquad \forall t \in [0,T], \\
& && u(t) \in [0,1], \qquad\qquad\qquad\quad \text{a.e. } t \in [0,T]
\end{aligned} \qquad (\mathcal{Z}_2)$$

with $X = (x, y)$, and using the control system and parameters from Section 4.2 and where A, α, B, $\beta \in \mathbb{R}^2$.

Note that the $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ method is theoretically unsuitable for this problem since, with notations from Section 3, we have $\Omega = \{N/2, N\}$ and $\Lambda = \{N\}$ while the inclusion $\Omega \subseteq \Lambda$ is required. However, to illustrate the lack of performance of the $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ method on this problem, we will implement it by adding the term $\lfloor \|X(T/2) - B\|_{\beta} \rfloor$ to the objective of Subproblem $(S(0, X_0, N, X_N))$ which is solved by IPOPT.

Figure 7 reports the numerical results obtained with the methods introduced in Section 3.2 on Problem $(\mathcal{Z}_2)$, with the parameters T, N, $y_{\max}$ and s as in Section 4.2 and with $A = (-325, 0)$, $\alpha = (\frac{1}{20}, \frac{5}{6})$, $B = (-175, 6)$ and $\beta = (\frac{1}{20}, \frac{1}{2})$. The plateaus associated with B are represented on the lower graph on the locus $]-225, 0] \times [0, 6]$, while the plateaus associated with A appear on the locus $[-350, -225[\times [0, 6]$.

The $\mathcal{M}_{\text{SO}}$ and $\mathcal{M}_{\text{DFBBO}}$ methods leave the boat uncontrolled and miss the targets. The $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ and $\mathcal{M}_{\text{hybrid}}^{\{N/2, N\}}$ methods perform better, although only the $\mathcal{M}_{\text{hybrid}}^{\{N/2, N\}}$ method approaches the midterm target. As expected, only the method explicitly handling both singularities is reliable. This example shows a case where the dynamics and the Lagrange cost repel the system from a target, since the stream repels the boat from the shores while confronting it is penalized by the Lagrange cost. Then, confronting the stream appears irrelevant for each method except the $\mathcal{M}_{\text{hybrid}}^{\{N/2, N\}}$ method.
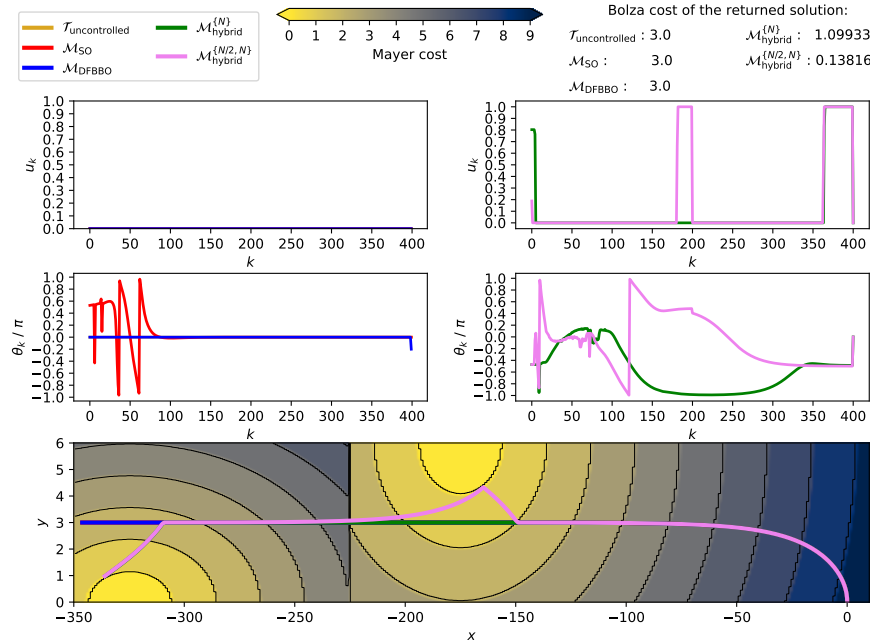

Figure 7: **Numerical results obtained on Problem $(\mathcal{Z}_2)$. The $\mathcal{M}_{\text{SO}}$ and $\mathcal{M}_{\text{DFBBO}}$ methods fail to differ from the uncontrolled trajectory. The $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ method reaches the lowest-cost plateau at the final time but fails to approach the midterm reference state. The $\mathcal{M}_{\text{hybrid}}^{\{N/2, N\}}$ method reaches both lowest-cost plateaus.**

4.4 A truncated variant of the Lotka-Volterra stabilization problem

In the three previous problems, the analysis of the numerical methods follows our expectations from Section 3.2. In particular the $\mathcal{M}_{\text{hybrid}}^{\Lambda}$ method performs better than the others, and Λ must be chosen accordingly to the singularities appearing in the Mayer cost function. Precisely Λ must contain the set Ω and, at a first glance, there is no need to make it larger. However, and surprisingly, some singular problems with $\Omega = \{N\}$ may make the $\mathcal{M}_{\text{hybrid}}^{\{N/2, N\}}$ method perform better than the $\mathcal{M}_{\text{hybrid}}^{\{N\}}$ method. In this section we enlighten and analyze this phenomenon on an example, based on a simplified two-

dimensional Lotka-Volterra biological system, given by

$$
\begin{aligned}
&\underset{\substack{X = (x,y) \in \mathcal{AC}([0,T], \mathbb{R}^2) \\ u \in \mathcal{L}^\infty([0,T], \mathbb{R})}}{\text{minimize}} && \lfloor \|X(T) - \overline{X}\|_\alpha \rfloor + \frac{1}{T} \int_0^T u(t)\, dt \\
&\text{subject to} && \dot{x}(t) = x(t)(y(t) - \overline{y} - u(t)), \quad \text{a.e. } t \in [0,T], \\
& && \dot{y}(t) = y(t)(\overline{x} - x(t)), \qquad\quad \text{a.e. } t \in [0,T], \\
& && x(0) = x_0, \\
& && y(0) = y_0, \\
& && u(t) \in [0, \tfrac{1}{4}], \qquad\qquad\qquad \text{a.e. } t \in [0,T]
\end{aligned}
\qquad (\mathcal{LV})
$$

where $X(t) = (x(t), y(t))$ represents the number of thousands of individuals in a specie of predators and preys respectively, $X_0 = (x_0, y_0) \in (\mathbb{R}_+^*)^2$ stands for the initial condition, $\overline{X} = (\overline{x}, \overline{y}) \in (\mathbb{R}_+^*)^2$ denotes the equilibrium of the Lotka-Volterra system, and $u(t) \in [0, 1/4]$ is the ratio of predators decimated by an human intervention at time t. With no intervention (that is, with $u \equiv 0$), the Lotka-Volterra system has a periodical motion. The objective of Problem $(\mathcal{LV})$ is to stabilize the system by approaching the equilibrium $\overline{X}$ (with respect to a discrete distance), and minimizing the average human intervention. We use the parameters $T = 60$, $N = 600$, $X_0 = (x_0, y_0) = (2.25, 4.25)$ and $\overline{X} = (\overline{x}, \overline{y}) = (1,1)$. They lead to the final state of the uncontrolled trajectory $X_N \approx (0.05, 1.59)$.

The controls and trajectories found by each method developed in Section 3.2 are shown on Figure 8. The $\mathcal{M}_{\mathrm{SO}}$ and $\mathcal{M}_{\mathrm{DFBBO}}$ methods fail to differ from the uncontrolled trajectory. Contrary to the prior problems, the $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ method also fails. The $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ method only pushes one control value at $1/4$ while all the others remain at 0, to slightly alter the final state X_N and make it cross one single plateau. This behavior is intriguing. Since this method leaves only the variable X_N to Nomad, at a first glance, there is no reason to explain why the method fails to propose a final state X_N closer to $\overline{X}$. This issue is of prime importance for further research works and is discussed in more details in the next paragraph. Finally note that the $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2, N\}}$ method performs well since it recovers a trajectory reaching the lowest-cost plateau.
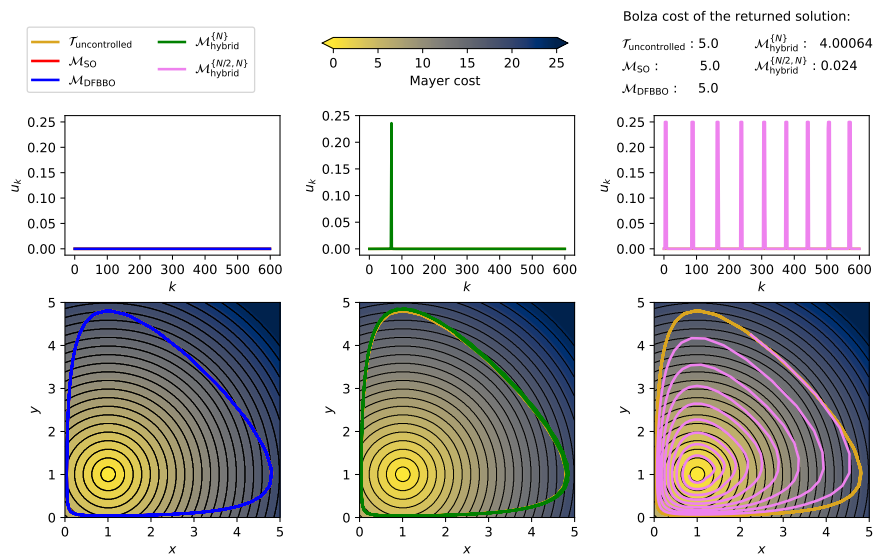
Figure 8: Numerical results obtained on Problem $(\mathcal{LV})$. The $\mathcal{M}_{\mathrm{SO}}$ and $\mathcal{M}_{\mathrm{DFBBO}}$ methods are both represented on the left graph and remain at their initialization. The $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ method also remains on a trajectory close to the uncontrolled one. The $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2, N\}}$ method performs well and successfully reaches the lowest-cost plateau.

As suggested by the numerical results obtained, Problem $(\mathrm{R}^{\{N\}})$ is difficult to solve numerically in the context of this section. Recall that, in Problem $(\mathrm{R}^{\{N\}})$, the only variable is the final state X_N,

while the search of a trajectory linking X_0 to X_N is performed via Subproblem $(\mathrm{S}(0, X_0, N, X_N))$. Recall also that IPOPT looks for a (feasible and) optimal solution of the subproblem, but it has no guarantee to localize neither an optimal solution nor a feasible one. As it aims primarily a feasible solution before optimizing it, IPOPT may converge to an infeasible solution in its attempt to solve a feasible subproblem, because this solution is a local minimum of the infeasibility measure. Hence the variable X_N may trigger an hidden constraint in Problem $(\mathrm{R}^{\{N\}})$, as it may generate a subproblem for which IPOPT fails to find a feasible solution. This constraint does not appear on the mathematical formulation of Problem $(\mathcal{LV})$, as it comes from the computer code implementing the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method, but it has an impact on the returned solution. This phenomenon explains why NOMAD struggles to solve Problem $(\mathrm{R}^{\{N\}})$ in the context of this section. Indeed, consider Figure 9 which shows the infeasibility of the solution returned by IPOPT on Subproblem $(\mathrm{S}(0, X_0, N, X_N))$ for each point X_N. The set of X_N leading to a subproblem for which the solution returned by IPOPT has a high infeasibility measure surrounds the equilibrium $\overline{X}$. Therefore, starting from the final state of the uncontrolled trajectory $X_N \approx (0.05, 1.59)$ in the large black ribbon of Figure 9, NOMAD sees its initialization as a feasible solution but also sees the colored zone as a zone of high infeasibility, and thus remains in the black ribbon. Finally, as the $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2, N\}}$ method performs better than the $\mathcal{M}_{\mathrm{hybrid}}^{\{N\}}$ method, we suspect that Subproblem $(\mathrm{S}(0, X_0, N, X_N))$ recovering the entire trajectory is harder to solve numerically by IPOPT than the two Subproblems $(\mathrm{S}(0, X_0, N/2, X_{N/2}))$ and $(\mathrm{S}(N/2, X_{N/2}, N, X_N))$ recovering half of the trajectory each. Furthermore we even found that IPOPT fails to find a feasible solution of the feasible Subproblem $(\mathrm{S}(0, X_0, N, X_N))$ taking as X_N the final state of the trajectory returned by the $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2, N\}}$ method. This subproblem is feasible (since the trajectory returned by the $\mathcal{M}_{\mathrm{hybrid}}^{\{N/2, N\}}$ method is a feasible solution), but IPOPT fails to detect a feasible solution.
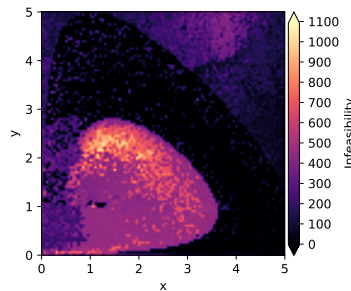
Figure 9: **The infeasibility of the solution returned by Ipopt attempting to solve Subproblem** $(\mathrm{S}(0, X_0, N, X_N))$ **for each** $X_N = (x, y) \in [0, 5]^2$. **It may differ from the true minimal infeasibility because Ipopt may converge to a locally infeasible solution in its attempt to solve a feasible Subproblem** $(\mathrm{S}(0, X_0, N, X_N))$.

5 Discussion

This work introduces an hybrid numerical method, relying on two algorithms from DFBBO and SO, to solve optimal control problems with a piecewise constant Mayer cost function. This hybrid method is based on a reformulation of the discretized problem which splits the search of a full trajectory-control pair into several subproblems searching for partial trajectory-control pairs. The state variables that may impact the singularities of the Mayer cost function are optimized on the main problem via a DFBBO algorithm, while the search of a trajectory joining those is left to smooth subproblems solved via a SO algorithm.

As evidenced in Section 4, the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method solves efficiently optimal control problems with a piecewise constant Mayer cost function, while non-hybrid approaches fail. Although we voluntarily restricted our work to optimal control problems with a piecewise constant Mayer cost function only, our method can be extended to a larger class of optimal control problems, including for example non-differentiable cost functions, piecewise continuous cost functions, extended-real valued cost functions,

etc., or cases where the control variables may also impact the singularities. From a general point of view, any cost or constraint function, or any dynamics, involving a singularity handled by DFBBO algorithms may be handled by an adapted hybrid method. However, this last class of problems is large and some other methods may be more appropriate for some of its subclasses. A future development could rely in a better identification of the class of problems on which hybrid methods work well.

As explained above, the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method can be extended in several directions. It can also be improved in several aspects. First, we know from Section 4.4 that a relevant choice of the subset Λ does not simply consist in making it as small as possible, even if a subset Λ with small cardinality is preferred. Hence, a possible improvement is to fix the cardinal of Λ and to leave its elements as variables of the main problem. Note that such a process would not require any algorithm from combinatorial optimization, since DFBBO algorithms handle problems with both discrete and continuous variables [30]. Preliminary numerical experiments are promising and this extension is expected to be the topic of future research works.

Second, we noticed that the feasibility of the subproblems highly depends on the values of the states optimized on the main problem, to the point that it may be difficult to find states leading to feasible subproblems with a DFBBO algorithm (see Figure 9 for an example). A possible improvement would be to relax the constraint forcing the subproblems to match exactly the states proposed by the main problem. The trajectory-finding subproblems may be relaxed to simply approach the states instead of matching them. This could be done by adding a penalization term to the objective functions of the subproblems, penalizing the distance between the states of the system at the times indexed by Λ and the proposed states at these times.

Third, the known problem structure may be exploited to guide the main DFBBO algorithm. For example, because of the dynamics of the control system constraining the trajectory, forcing the system to reach a given state at a given time restricts the space of admissible states at future times. Hence, in the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method, there are links between the state variables optimized at the main level. Furthermore, we may construct surrogates [15, 20, 31, 32] of the subproblems to obtain a fast approximation of their optimal solutions. For example, we may launch a SO algorithm and stop its execution after a few iterations, and consider the low-precision solution. The DFBBO algorithm may use this procedure to estimate how relevant a potential solution is, with no need to systematically solve the subproblems with full accuracy. Such a development could significantly reduce the computational time of the $\mathcal{M}_{\mathrm{hybrid}}^{\Lambda}$ method.

References

[1] A. Rao. A survey of numerical methods for optimal control. Advances in the Astronautical Sciences, 135, 01 2010.

[2] E. Trélat. Contrôle optimal : théorie et applications. Vuibert supérieur - Mathématiques concrètes. Vuibert, 2005. URL https://www.ljll.math.upmc.fr/trelat/.

[3] A. Bressan and B. Piccoli. Introduction to the Mathematical Theory of Control. AIMS on applied math. American Institute of Mathematical Sciences, 2007. URL https://books.google.it/books?id=gPtUAAAAYAAJ.

[4] A.E. Bryson and Y.-C. Ho. Applied Optimal Control: optimization, estimation and control. Halsted Press book'. Taylor & Francis, 1975. URL https://books.google.fr/books?id=P4TKxn7qW5kC.

[5] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. The mathematical theory of optimal processes. Interscience Publishers John Wiley & Sons, Inc. New York-London, 1962. Translated from the Russian by K. N. Trirogoff; edited by L. W. Neustadt.

[6] F. Clarke. Functional Analysis, Calculus of Variations and Optimal Control, volume 264. Springer-Verlag London, 01 2013. URL https://doi.org/10.1007/978-1-4471-4820-3.

[7] Halina Frankowska. The first order necessary conditions for nonsmooth variational and control problems. SIAM Journal on Control and Optimization, 22(1):1–12, 1984. URL https://doi.org/10.1137/0322001.

[8] R. Vinter. Optimal Control. Modern Birkhäuser Classics. Birkhäuser Boston, 2010. URL https://books.google.fr/books?id=9fSOky11oswC.

[9] T. Haberkorn and E. Trélat. Convergence results for smooth regularizations of hybrid nonlinear optimal control problems. SIAM Journal on Control and Optimization, 49(4):1498–1522, 2011. URL https://doi.org/10.1137/100809209.

[10] T. Bayen and L. Pfeiffer. Second order conditions for a control problem with discontinuous cost. In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 1480–1485, 2019. URL https://doi.org/10.1109/CDC40024.2019.9029314.

[11] A. Schwartz and E. Polak. Consistent approximations for optimal control problems based on Runge–Kutta integration. SIAM Journal on Control and Optimization, 34(4):1235–1269, 1996. URL https://doi.org/10.1137/S0363012994267352.

[12] J. Nocedal and S.J. Wright. Numerical Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Berlin, second edition, 2006. URL http://www.springer.com/mathematics/book/978-0-387-30303-1.

[13] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming, 106(1):25–57, 2006. URL https://dx.doi.org/10.1007/s10107-004-0559-y.

[14] C. Audet and W. Hare. Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland, 2017. URL https://dx.doi.org/10.1007/978-3-319-68913-5.

[15] A.R. Conn, K. Scheinberg, and L. Vicente. Introduction to Derivative-Free Optimization. Society for Industrial and Applied Mathematics, USA, 2009.

[16] C. Audet and J.E. Dennis, Jr. Mesh Adaptive Direct Search Algorithms for Constrained Optimization. SIAM Journal on Optimization, 17(1):188–217, 2006. URL https://dx.doi.org/Doi:10.1137/040603371.

[17] C. Audet, S. Le Digabel, V. Rochon Montplaisir, and C. Tribes. NOMAD version 4: Nonlinear optimization with the MADS algorithm. Technical Report G-2021-23, Les cahiers du GERAD, 2021. URL http://www.optimization-online.org/DB_HTML/2021/04/8351.html.

[18] C. Audet, A.R. Conn, S. Le Digabel, and M. Peyrega. A progressive barrier derivative-free trust-region algorithm for constrained optimization. Computational Optimization and Applications, 71(2):307–329, 2018. URL https://link.springer.com/article/10.1007/s10589-018-0020-4.

[19] A.M. Bagirov, M. Gaudioso, N. Karmitsa, M.M. Mäkelä, and S. Taheri. Numerical Nonsmooth Optimization: State of the Art Algorithms. Springer International Publishing, 2020. URL https://www.springer.com/gp/book/9783030349097.

[20] A. Custódio, K. Scheinberg, and L. Vicente. Chapter 37: Methodologies and Software for Derivative-Free Optimization, pages 495–506. Society for Industrial and Applied Mathematics, 04 2017. URL https://doi.org/10.1137/1.9781611974683.ch37.

[21] G. Liuzzi, S. Lucidi, F. Rinaldi, and L. Vicente. Trust-region methods for the derivative-free optimization of nonsmooth black-box functions. SIAM Journal on Optimization, 29:3012–3035, 01 2019. doi: 10.1137/19M125772X.

[22] F. Fisch. Development of a Framework for the Solution of High-Fidelity Trajectory Optimization Problems and Bilevel Optimal Control Problems. Dissertation, Technische Universität München, München, 2011.

[23] F.M Petersen, D.E. Cornick, G.L. Bauer, and J.R. Rehder. A two-level trajectory decomposition algorithm featuring optimal intermediate target selection. Journal of Spacecraft and Rockets, 14(11):676–682, 1977.

[24] S. Dempe. Bilevel Optimization. Springer Optimization and Its Applications. Springer International Publishing, Cham, 2020. URL https://doi.org/10.1007/978-3-030-52119-6_20.

[25] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. IEEE Transactions on Evolutionary Computation, 22(2):276–295, 4 2018. doi: 10.1109/TEVC.2017.2712906.

[26] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Mathematical Programming, Series A, 91:239–269, 2002. URL https://dx.doi.org/10.1007/s101070100244.

[27] C. Audet and J.E. Dennis, Jr. A Progressive Barrier for Derivative-Free Nonlinear Programming. SIAM Journal on Optimization, 20(1):445–472, 2009. URL https://dx.doi.org/10.1137/070692662.

[28] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. SIAM Review, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

[29] A. Montoison, P. Pascal, and L. Salomon. NOMAD.jl: A Julia interface for the constrained blackbox solver NOMAD. https://github.com/bbopt/NOMAD.jl, July 2020.

[30] C. Audet, S. Le Digabel, and C. Tribes. The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables. SIAM Journal on Optimization, 29(2):1164–1189, 2019. URL https://dx.doi.org/10.1137/18M1175872.

[31] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. Structural and Multidisciplinary Optimization, 17(1):1–13, 1999. URL https://dx.doi.org/10.1007/BF01197708.

[32] B. Talgorn. sgtelib: Surrogate model library for derivative-free optimization. https://github.com/bbopt/sgtelib, 2019. URL https://github.com/bbopt/sgtelib.