Optimistic Planning Algorithms For State-Constrained Optimal Control Problems

Olivier Bokanowski, Nidhal Gammoudi, Hasnaa Zidani

Optimistic Planning Algorithms For State-Constrained Optimal Control Problems

Olivier Bokanowski

Université de Paris, LJLL, F-75013 Paris, France and Sorbonne Université, CNRS, LJLL, F-75005 Paris, France

Nidhal Gammoudi

Unité de Mathématiques Appliquées (UMA), Ensta Paris, 828 Bd des Maréchaux, 91762 Palaiseau, France

Hasnaa Zidani

Normandie Univ, INSA Rouen Normandie, Laboratoire de Mathématiques Appliquées (LMI), 76000 Rouen, France

Abstract

In this work, we study optimistic planning methods to solve some state-constrained optimal control problems in finite horizon. While classical methods for calculating the value function are generally based on a discretization in the state space, optimistic planning algorithms have the advantage of using adaptive discretization in the control space. These approaches are therefore very suitable for control problems where the dimension of the control variable is low and allow to deal with problems where the dimension of the state space can be very high. Our algorithms also have the advantage of providing, for given computing resources, the best control strategy whose performance is as close as possible to optimality while its corresponding trajectory comply with the state constraints up to a given accuracy.

Keywords: Optimistic planning algorithm, optimal control problems, state constraints

2010 MSC: 49M30, 49M25, 65K05

1. Introduction

This paper concerns a numerical approach for solving some state-constrained non-linear optimal control problems. Let $T > 0$ be a fixed final horizon, and let A be a given non-empty compact subset of a metric

Email addresses: olivier.bokanowski@math.univ-paris-diderot.fr (Olivier Bokanowski), Nidhal.Gammoudi@ensta-paris.fr (Nidhal Gammoudi), Hasnaa.Zidani@insa-rouen.fr (Hasnaa Zidani)

Acknowledgments. The third author acknowledges a co-funding by the European Union with the European regional development fund (ERDF,20E07206) and by the Normandie Regional Council via the "Chaire COPTI" project.

space. Consider two closed subsets $\mathcal{K}$ and $\mathcal{C}$ of $\mathbb{R}^d$ ($d \geq 1$). For $x \in \mathbb{R}^d$, the control problem is the following

$$\vartheta(x) := \inf \left\{ \int_0^T \ell(y_x^\alpha(s), \alpha(s))\, ds + \Phi(y_x^\alpha(T)) \;\middle|\; \alpha(s) \in A, \quad y_x^\alpha(s) \in \mathcal{K} \;\text{ for } s \in [0,T], \quad \text{and} \quad y_x^\alpha(T) \in \mathcal{C} \right\}, \tag{1}$$

where the state $y_x^\alpha(\cdot)$ and the control $\alpha(\cdot)$ are related by the following non-linear differential equation

$$\dot{y}(s) = f(y(s), \alpha(s)) \text{ a.e. } s \in (0,T), \qquad y(0) = x. \tag{2}$$

In this problem, the functions $f : \mathbb{R}^d \times A \to \mathbb{R}^d$, $\ell : \mathbb{R}^d \times A \to \mathbb{R}^d$ and $\Phi : \mathbb{R}^d \to \mathbb{R}$ are Lipschitz continuous (the precise assumptions are given in section 2). Several mathematical and numerical contributions have been devoted to the analysis and approximation of the solutions of (1). Most of the classical optmisation-based methods are only able to guarantee the computation of local solutions, but there are also some mathematical methods that aim at computing the global solutions. Among these methods, the dynamic programming approach gives a mathematical characterization of the value function ϑ and provides the optimal control law in a feedback form, which is interesting for many engineering applications. However, the numerical cost of this approach restricts its application to problems where the state space is of small dimension.

The approach that we would like to investigate in the present work is based on optimistic planing (OP) algorithms [9, 10]. These algorithms use a discretization in the space of controls (and do not require any discretization of the state space). This approach is very interesting especially for many applications where the control dimension q is much lower compared to the state dimension d. On a given discretization of the time interval, our approach will seek to identify the best control strategy to apply on each time sub-interval. The OP methods perform the optimal control search by branch and bound on the control set, always refining the region with the best lower bound of the optimal value (this is what justifies the label "optimistic"). An interesting feature of these algorithms is the close relationship between computational resources and quasi-optimality, which exploits some ideas of reinforcement learning [18]. Indeed, for given computational resources, the OP approaches provide a sub-optimal strategy whose cost is *as close as possible* to the optimal value (the measure of "close to the optimum" will be clarified in section 3).

Many OP variants have been proposed in the literature (see [8, 13, 14, 15] and the references therein) with mainly heuristic rules for the refinement and without convergence analysis. In [9, 10], some OP methods are proposed for continuous actions. These methods use selection rules for adaptive refinement of the control. A rigorous analysis of convergence and of the complexity of the methods are also presented in [9], within the framework of an infinite horizon problem with a discount factor and without state constraints.

Our main contribution is to extend some optimistic planning algorithms to solve finite horizon control problems in presence of state constraints. We first use some ideas introduced in [2, 4] to reformulate the original control problem as an auxiliary problem without state constraints. In this auxiliary problem, the set of controls remain the same, however, the dimension of the state space is increased by one more component. We adapt two algorithms introduced in [10, 9] (OPC - "Optimistic Planning with Continuous actions", and SOP(or SOPC) - "Simultaneous OPC") to solve the auxiliary control problem and then to get an approximation of the optimal control strategy for the original state-constrained problem. We show that in this framework, we can obtain convergence results similar to those established in [10]. Besides, we improve the analysis of the complexity of these algorithms and provide simplified proof arguments for this analysis. Furthermore, we propose an algorithm which combines both the optimistic planning approach with ideas from the MPC (Model Predictive Control). This algorithm gives numerical results similar to those of SOP but reduces significantly the computational time. We illustrate the relevance of the OP algorithms on several non-linear optimal control problems (in one of these examples, the dimension of the state is 10^3).

Notations. In all this paper, $\mathbb{R}$ denotes the set of real numbers, $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^r$(for any $r \geq 1$). For any set $S \subseteq \mathbb{R}^N$, ∂S, denotes its boundary. The distance function to S is $\mathrm{dist}(x,S) = \inf\{\|x-y\| :\ y \in S\}$. For any $a_1, \cdots, a_q \in \mathbb{R}$, the notation $\bigvee_{i=1}^{q} a_i$ stands for $\max(a_1, \cdots, a_q)$. For any finite set U, $|U|$ is the cardinality of U. Finally, the abbreviation "w.r.t." stands for "with respect to", and "a.e." means "almost everywhere".

2. State-constrained control problems.

Let $T > 0$ be a fixed final time, and let A be a given non-empty compact subset of $\mathbb{R}^q$ ($q \geq 1$) . Consider the controlled system:

$$\dot{y}(s) = f(y(s), \alpha(s)) \ \text{ a.e. } s \in (t,T), \qquad \text{and } \ y(0) = x, \tag{3}$$

where $\alpha(\cdot)$ is an admissible control in

$$\mathcal{A}_{\mathtt{ad}} := \Big\{ \alpha : (0,T) \to \mathbb{R}^q \text{ measurable, } \alpha(t) \in A \text{ a.e.} \Big\},$$

and the dynamics $f : \mathbb{R}^d \times A \to \mathbb{R}^d$ is a Lipschitz continuous function satisfying:

(A1) There exist $L_{f,x} > 0$ and $L_{f,a} > 0$, for any $x, y \in \mathbb{R}^d$, for all $a, a' \in A$, we have:

$$\|f(x,a) - f(y,a')\| \leq L_{f,x}\|x-y\| + L_{f,a}\|a-a'\|.$$

Under assumption **(A1)**, for any $\alpha \in \mathcal{A}_{\mathtt{ad}}$ and for any $x \in \mathbb{R}^d$, there exists a unique absolutely continuous trajectory $y = y_x^\alpha$ satisfying (3). Let $\mathcal{K}$ and $\mathcal{C}$ be two non-empty closed sets of $\mathbb{R}^d$. A trajectory y_x^α will be said *feasible* (on the time interval $(0,T)$) if

$$y_x^\alpha(s) \in \mathcal{K}, \quad \text{for all } s \in (0,T), \qquad \text{and } \; y_x^\alpha(T) \in \mathcal{C}. \tag{4}$$

Now, consider two Lipschitz continuous functions $\ell : \mathbb{R}^d \times A \to \mathbb{R}$ and $\Phi : \mathbb{R}^d \to \mathbb{R}$ satisfying:

(A2) There exists $L_{\ell,x} \geq 0$ and $L_{\ell,a} \geq 0$ such that for any $x, y \in \mathbb{R}^d$ and for any $a, a' \in A$, we have:

$$|\ell(x,a) - \ell(y,a')| \leq L_{\ell,x}\|x-y\| + L_{\ell,a}\|a-a'\|.$$

(A3) There exists $L_\Phi \geq 0$ such that for any $x, y \in \mathbb{R}^d$, we have: $|\Phi(x) - \Phi(y)| \leq L_\Phi \|x-y\|$.

Throughout the paper, we will assume that assumptions **(A1)-(A3)** are satisfied. The state-constrained control problem is formulated as follows:

$$\vartheta(x) := \inf\left\{ \int_0^T \ell(y_x^\alpha(s), \alpha(s))\, ds + \Phi(y_x^\alpha(T)) \;\middle|\; \alpha \in \mathcal{A}_{\mathtt{ad}},\ y_x^\alpha(s) \in \mathcal{K}\ \forall s \in [0,T], \ \text{ and } y_x^\alpha(T) \in \mathcal{C} \right\}, \tag{5}$$

with the convention that $\inf \emptyset = +\infty$. In general, when $\mathcal{K} \neq \mathbb{R}^d$ or $\mathcal{C} \neq \mathbb{R}^d$, it may happen that the set of feasible trajectories is empty, and in this case the function ϑ may take an infinite value. Indeed, unless some controllability assumptions are satisfied, we may have no control input α that can act on the system and force it to comply with the state constraints [20, 12]. So, in general, it is not an easy task to know apriori if the value $\vartheta(x)$ is finite or not. The approach which we will use thereafter is based on an idea introduced in [2]. It provides the value $\vartheta(x)$ and an optimal trajectory associated to (5) (if any). When the value $\vartheta(x)$ is infinite (i.e., when the set of feasible trajectories is empty), our approach computes trajectories which minimize the cost and remain as close as possible to the sets of constraints.

Let us consider two Lipschitz continuous functions g and Ψ verifying the following property:

$$\forall y \in \mathbb{R}^d, \quad g(y) \leq 0 \iff y \in \mathcal{K} \quad \text{and} \quad \Psi(y) \leq 0 \iff y \in \mathcal{C}. \tag{6}$$

In the sequel, we denote by L_g and L_Ψ the Lipschitz constants of g and Ψ respectively. Since $\mathcal{K}$ and $\mathcal{C}$ are closed sets, the existence of Lipschitz functions satisfying (6) is guaranteed. For instance, g and Ψ can be

chosen, respectively, as the signed distance to $\mathcal{K}$ and the signed distance to $\mathcal{C}$:

$$d_{\mathcal{K}}(x) := \begin{cases} \mathrm{dist}(x, \partial\mathcal{K}) & \text{if } x \notin \mathcal{K} \\ -\mathrm{dist}(x, \partial\mathcal{K}) & \text{if } x \in \mathcal{K} \end{cases} \quad \text{and} \quad d_{\mathcal{C}}(x) := \begin{cases} \mathrm{dist}(x, \partial\mathcal{C}) & \text{if } x \notin \mathcal{C} \\ -\mathrm{dist}(x, \partial\mathcal{C}) & \text{if } x \in \mathcal{C} \end{cases}$$

The auxiliary control problem is defined, for $(x, z) \in \mathbb{R}^d \times \mathbb{R}$, as follows (with $a \vee b \vee c := \max(a, b, c)$):

$$w(x, z) := \inf_{\alpha \in \mathcal{A}_{\mathrm{ad}}} \left\{ \Big[\int_0^T \ell(y_x^\alpha(s), \alpha(s))\, ds + \Phi(y_x^\alpha(T)) - z \Big] \bigvee \max_{s \in [0,T]} g(y_x^\alpha(s)) \bigvee \Psi(y_x^\alpha(T)) \right\}. \tag{7}$$

Let us emphasize on that the auxiliary control problem is free of state constraints. The cost function in (7) is called the improvement function. Similar auxiliary problems have been introduced and analyzed in the context of finite-dimensional optimization problems in [17, 19] (see also [3]). For continuous time control problems, the properties of the auxiliary problem have been established in [2, 4]. Among other things, it holds that a couple $(\bar{y}, \bar{u})$ is an optimal solution for (5) if and only if it is also an optimal solution of (7) with $z = \vartheta(x)$. Moreover, the value of $\vartheta(x)$ can be recovered from $w(x, \cdot)$ (see [2, 4] for a precise statement).

3. Discrete time setting

First, for $N \geq 1$, consider a uniform partition of $[0, T]$ with $N + 1$ time steps: $t_k = k\Delta t$, $k = 0, \ldots, N$, where $\Delta t = \frac{T}{N}$ is the step size. We introduce a discrete dynamics defined as follows (by Heun scheme):

$$F(x, a) := x + \frac{\Delta t}{2}\Big(f(x, a) + f(x + \Delta t f(x, a), a) \Big).$$

For a sequence of actions $a = (a_k)_{0 \leq k \leq N-1} \in A^N$, we consider the discrete time dynamical system:

$$\begin{cases} y_0 = x, \\ y_{k+1} = F(y_k, a_k) \quad k = 0, ..., N-1. \end{cases} \tag{8}$$

For every $x \in \mathbb{R}^d$ and every sequence $a = (a_k)_k \in A^N$, equation (8) admits a unique solution that will be denoted by $y^{x,a} = (y_k^{x,a})_{k=0}^N$. This trajectory is said feasible if:

$$y_k^{x,a} \in \mathcal{K}, \quad \text{for any } \; k = 0, ..., N \quad \text{and} \quad y_N^{x,a} \in \mathcal{C}.$$

Consider also an instantaneous cost function ρ that approximates the integral of ℓ over an interval $[t_k, t_{k+1}]$,

for $k = 0, ..., N-1$, by a quadrature rule, as follows:

$$\rho(x,a) := \frac{\Delta t}{2}\Big(\ell(x,a) + \ell(F(x,a),a)\Big).$$

One can easily check that Under assumption (A1) and (A2), the functions F and ρ are Lipschitz continuous:

$$\|F(x,a) - F(x',a')\| \le L_{F,x}\|x - x'\| + L_{F,a}\|a - a'\|,$$
$$|\rho(x,a) - \rho(x',a')| \le L_{\rho,x}\|x - x'\| + L_{\rho,a}\|a - a'\|,$$

for every $x \in \mathbb{R}^d$ and $a \in A$, where

$$L_{F,x} := 1 + \Delta t L_{f,x}\Big(1 + \frac{\Delta t}{2}L_{f,x}\Big), \quad L_{F,a} := \Delta t L_{f,a}\Big(1 + \frac{\Delta t}{2}L_{f,x}L_{f,a}\Big), \quad \text{and} \tag{9a}$$
$$L_{\rho,x} := \frac{\Delta t}{2}L_{\ell,x}\Big(1 + L_{F,x}\Big), \quad L_{\rho,a} := \Delta t\Big(L_{\ell,a} + \frac{1}{2}L_{\ell,x}L_{F,a}\Big). \tag{9b}$$

Note that other approximations of the dynamics and of the integral term would lead to similar estimates as in (9). The approximations presented in this section are the ones that will be used in the numerical simulations of Section 5.

Now, we define the state-constrained optimal control problem as follows:

$$V(x) := \inf_{a=(a_k)_k \in A^N}\left\{\sum_{k=0}^{N-1}\rho(y_k^{x,a},a_k) + \Phi(y_N^{x,a}) \;\middle|\; y_k^{x,a} \in \mathcal{K},\ \ \forall k = 0,...,N \ \text{ and } \ y_N^{x,a} \in \mathcal{C}\right\}. \tag{10}$$

Then, for the discrete auxiliary control problem, we introduce the cost functional J defined by

$$J(x,z,a) := \Big(\sum_{k=0}^{N-1}\rho(y_k^{x,a},a_k) + \Phi(y_N^{x,a}) - z\Big) \bigvee \Big(\max_{0\le k\le N} g(y_k^{x,a})\Big) \bigvee \Psi(y_N^{x,a}) \tag{11}$$

(for $(x,z,a=(a_k)) \in \mathbb{R}^d \times \mathbb{R} \times A^N$) and the corresponding value is defined as follows, for $(x,z) \in \mathbb{R}^d \times \mathbb{R}$:

$$W(x,z) := \inf_{a=(a_k)_k \in A^N} J(x,z,a). \tag{12}$$

It is worth to mention that $W(x,z)$ converges to $w(0,x,z)$ as $N \to \infty$ (i.e. $\Delta t \to 0$), and the sequence of discrete-time optimal trajectories (for $N \in \mathbb{N}$) provide convergent approximations of optimal trajectories of the continuous problem (a precise statement and its proof can be found in [4]).

By using similar arguments as in [4, 5], one can prove the folowing result:

Proposition 3.1. *We have:*

$$V(x) = \inf\{z \mid W(x,z) \leq 0\}. \tag{13}$$

Furthermore, a pair $(\bar{y}, \bar{a})$ *is optimal for problem* (10) *if and only if* $(\bar{y}, \bar{a})$ *is optimal for problem* (12) *with* $z = V(x)$.

Remark 3.2. *A classic approach to take into account the constraints in* (10) *consists in approaching* $V(x)$ *by a penalized problem as follows:*

$$V_\epsilon(x) := \inf_{a=(a_k)_k \in A^N} J_\epsilon(x,a),$$

with $J_\epsilon(x,a) := \sum_{k=0}^{N-1} \rho(y_k^{x,a}, a_k) + \Phi(y_N^{x,a}) + \frac{1}{\epsilon}\Big(\sum_{k=0}^{N} g(y_k^{x,a})^+ + \Psi(y_N^{x,a})^+\Big)$. *The advantage of the penalized problem is that it does not increase the dimension of the state variable. However, the Lipschitz constant of the penalized criterion* J_ϵ *involves a term in* $\frac{1}{\epsilon}$ *which tends to infinity when the penalization parameter* ϵ *goes to* 0. *As it will be shown in the paper, the Lipschitz constants are used in the elaboration of the optimistic algorithms and in their complexity analysis. Let us emphasize on that the auxiliary problem involves a cost function whose Lipschitz constant does not explode, which makes this problem more suitable than the classical penalization approach for using optimistic methods and taking into account the state constraints.*

Now, let us state a preliminary result that will be useful in the rest of the paper.

Proposition 3.3. *For any* $(x,z) \in \mathbb{R}^d \times \mathbb{R}$ *and* $a = (a_k)_k$, $\bar{a} = (\bar{a}_k)_k \in A^N$, *the following estimate is satisfied:*

$$|J(x,z,a) - J(x,z,\bar{a})| \leq \Big(\sum_{k=0}^{N-1} \beta_k \|a_k - \bar{a}_k\|\Big) \bigvee \Big(\sum_{k=0}^{N-1} \gamma_k \|a_k - \bar{a}_k\|\Big), \tag{14}$$

where $\beta_k := L_{\rho,x} L_{F,a} \dfrac{L_{F,x}^{N-k-1} - 1}{L_{F,x} - 1} + L_{\rho,a} + L_\Phi L_{F,a} L_{F,x}^{N-k-1}$, *and* $\gamma_k := \Big(L_g \bigvee L_\Psi\Big) L_{F,a} L_{F,x}^{N-k-1}$.

A quite similar estimate of the difference of performances is established, in [10, 9], for the case of an infinite horizon sum with a positive discount factor and under a boundedness assumption on the instantaneous reward. Here, the cost functional J is defined, in finite horizon, as a maximum of several terms, and without discount factor.

Proof. Denote by $y = (y_k)_k$ the discrete trajectory associated to the input a, and by $\bar{y} = (\bar{y}_k)_k$ the discrete trajectory associated to $\bar{a}$. Both trajectories start from the initial position x. By Lipschitz continuity of the

dynamics F, straightforward calculations give:

$$\|y_k - \bar{y}_k\| \leq L_{F,a} \sum_{j=0}^{k-1} L_{F,x}^{k-1-j} \|a_j - \bar{a}_j\|, \quad 0 \leq k \leq N. \tag{15}$$

Now, by using the Lipschitz regularity of the cost functions, we obtain:

$$|J(x,z,a) - J(x,z,\bar{a})| \leq \Big(\sum_{k=0}^{N-1} L_{\rho,x} \|y_k - \bar{y}_k\| + \sum_{k=0}^{N-1} L_{\rho,a} \|a_k - \bar{a}_k\| + L_\Phi \|y_N - \bar{y}_N\| \Big) \bigvee \Big(L_g \max_{0 \leq k \leq N} \|y_k - \bar{y}_k\| \Big) \bigvee \Big(L_\Psi \|y_N - \bar{y}_N\| \Big). \tag{16}$$

From (15), we get:

$$\sum_{k=0}^{N-1} \|y_k - \bar{y}_k\| \leq L_{F,a} \sum_{k=0}^{N-1} \sum_{j=0}^{k-1} L_{F,x}^{k-1-j} \|a_j - \bar{a}_j\| \leq L_{F,a} \sum_{k=0}^{N-1} \frac{L_{F,x}^{N-k-1} - 1}{L_{F,x} - 1} \|a_k - \bar{a}_k\|,$$

and for any $k = 0, ..., N-1$, we have also: $\|y_k - \bar{y}_k\| \leq L_{F,a} \sum_{j=0}^{N-1} L_{F,x}^{N-j-1} \|a_j - \bar{a}_j\|$. Therefore,

$$\Big(L_g \max_{0 \leq k \leq N} \|y_k - \bar{y}_k\| \Big) \bigvee \Big(L_\Psi \|y_N - \bar{y}_N\| \Big) \leq \Big(L_g \bigvee L_\Psi \Big) L_{F,a} \sum_{k=0}^{N-1} L_{F,x}^{N-k-1} \|a_k - \bar{a}_k\|.$$

Combining the above inequalities gives the desired result (14). □

Corollary 3.4 (Upper bound)**.** *There exists a constant $C > 0$, independent of N, such that for any $(x,z) \in \mathbb{R}^d \times \mathbb{R}$ and $a = (a_k)_k, a_k = (\bar{a}_k)_k \in A^N$, we have:*

$$|J(x,z,a) - J(x,z,\bar{a})| \leq C\Delta t \sum_{k=0}^{N-1} \big(\frac{1}{L_{F,x}}\big)^k \|a_k - \bar{a}_k\|. \tag{17}$$

Proof. From (9), we know that $L_{F,x} = 1 + \Delta t \overline{L}$ with $\overline{L} := L_{f,x}(1 + \frac{\Delta t}{2} L_{f,x}) > 0$. Thus, for every $0 \leq k \leq N-1$:

$$\frac{L_{F,x}^{N-k-1} - 1}{L_{F,x} - 1} \leq \frac{L_{F,x}^{N-1}}{L_{F,x} - 1} (L_{F,x})^{-k} \leq \frac{e^{N\Delta t \overline{L}}}{\Delta t \overline{L}} (L_{F,x})^{-k}.$$

Using the definitions of $L_{\rho,x}$ and $L_{F,a}$ (see (9)), we obtain:

$$L_{\rho,x} L_{F,a} \frac{L_{F,x}^{N-k-1} - 1}{L_{F,x} - 1} \leq C_{1,1} \Delta t \Big(\frac{1}{L_{F,x}} \Big)^k,$$

with $C_{1,1} := \frac{e^{T\overline{L}}}{2\overline{L}} L_{\ell,x}(1+L_{F,x})L_{f,a}(1+\frac{T}{2}L_{f,x}L_{f,a})$, a constant independent of N. Again from (9), we get:

$$L_{\rho,a} = \Big(\frac{1}{L_{F,x}}\Big)^k L_{\rho,a}(L_{F,x})^k \leq \Big(\frac{1}{L_{F,x}}\Big)^k L_{\rho,a} e^{k\Delta t\overline{L}} \leq C_{1,2}\Delta t\Big(\frac{1}{L_{F,x}}\Big)^k,$$

with $C_{1,2} := e^{T\overline{L}}\left(L_{\ell,a} + \frac{1}{2}L_{\ell,x}L_{F,a}\right)$ is a constant independent of N. With similar argument, we also obtain:

$$L_\Phi L_{F,a} L^{N-k-1} \leq C_{1,3}\Delta t\Big(\frac{1}{L_{F,x}}\Big)^k \qquad \text{with} \quad C_{1,3} := L_\Phi e^{T\overline{L}} L_{f,a}\left(1+\frac{T}{2}L_{f,x}L_{f,a}\right).$$

By setting $C_1 = C_{1,1} + C_{1,2} + C_{1,3}$, we conclude that: $\beta_k \leq C_1\Delta t\Big(\frac{1}{L_{F,x}}\Big)^k$. Similar arguments lead to an upper bound of γ_k as follows:

$$\gamma_k = C_2\Delta t\Big(\frac{1}{L_{F,x}}\Big)^k, \quad \text{where} \quad C_2 := \left(L_g \bigvee L_\Psi\right) e^{T\overline{L}} L_{f,a}\left(1+\frac{T}{2}L_{f,x}L_{f,a}\right).$$

In conclusion, we obtain the claim with $C := C_1 \bigvee C_2$. □

4. Optimistic planning

For sake of simplicity and without loss of generality, we suppose that the control is of dimension $q = 1$ and we denote by D its maximal diameter ($\forall a, a' \in A, \|a - a'\| \leq D$). Let us emphasize that the approach can be generalized to control variables with multiple dimensions.

4.1. Optimistic planning approach

Planning algorithms are based on the principles of optimistic optimisation (see [11]). In order to minimize the objective function J over the space A^N, we refine, in an iterative way the search space into smaller subsets.

A search space, called node and denoted by $\mathbb{A}_i$ with $i \in \mathbb{N}$, is a Cartesian product of sub-intervals of A *i.e.* $\mathbb{A}_i := A_{i,0} \times A_{i,1} \times ... \times A_{i,N-1} \subseteq A^N$, where $A_{i,k}$ represents the control interval at time step k, for $k = 0...N-1$. The collection of nodes will be organized into a tree Υ that will be constructed progressively by expanding the tree nodes. Expanding a node $\mathbb{A}_i$, with $i \in \mathbb{N}$, consists in choosing an interval $A_{i,k}$, for $k = 0...N-1$, and splitting it uniformly to M sub-intervals where $M > 1$ is a parameter of the algorithm.

In figure 1, we represent a simple example of a tree construction to explain. At the beginning of the process, the tree contains only its root $\mathbb{A}_0 := [0,1]^N$, from which we generate M=3 children nodes $\mathbb{A}_1 = [0, \frac{1}{3}] \times A^{N-1}$, $\mathbb{A}_2 = [\frac{1}{3}, \frac{2}{3}] \times A^{N-1}$ and $\mathbb{A}_3 = [\frac{2}{3}, 1] \times A^{N-1}$, after splitting its first interval. Then, suppose that $\mathbb{A}_1$ is the second node to be refined by splitting its second interval. Hence, we get $\mathbb{A}_4 = [0, \frac{1}{3}] \times [0, \frac{1}{3}] \times A^{N-2}$,

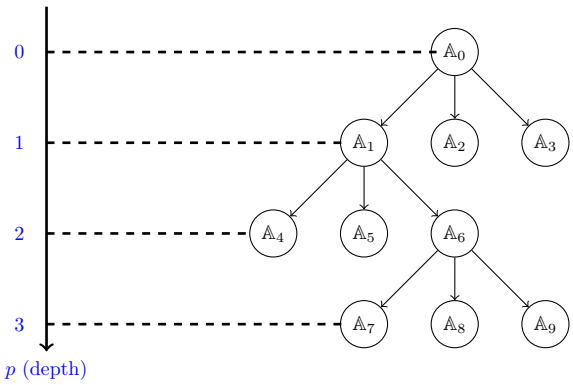

Figure 1: Illustrative example of refinement of A^N with $M = 3$ after splitting 3 nodes $\mathbb{A}_0$, $\mathbb{A}_1$ and $\mathbb{A}_6$.

$\mathbb{A}_5 = [0, \frac{1}{3}] \times [\frac{1}{3}, \frac{2}{3}] \times A^{N-2}$ and $\mathbb{A}_6 = [0, \frac{1}{3}] \times [\frac{2}{3}, 1] \times A^{N-2}$. Finally, we choose $\mathbb{A}_6$ and we split its first interval $[0, \frac{1}{3}]$ to generate $\mathbb{A}_7 = [0, \frac{1}{9}] \times [\frac{2}{3}, 1] \times A^{N-2}$, $\mathbb{A}_8 = [\frac{1}{9}, \frac{2}{9}] \times [\frac{2}{3}, 1] \times A^{N-2}$ and $\mathbb{A}_9 = [\frac{2}{9}, \frac{1}{3}] \times [\frac{2}{3}, 1] \times A^{N-2}$. The order of expanded nodes and the intervals that have to be split will be made precise later. For now, we introduce some useful notations related to the tree Υ:

- We associate, for any node $\mathbb{A}_i \in \Upsilon$, a sample sequence of controls $a_i := (a_{i,k})_{k=0}^{N-1} \in \mathbb{A}_i$ such that $a_{i,k}$ corresponds to the midpoint of the interval $A_{i,k}$ for any $k = 0, ..., N-1$.

- For any node $\mathbb{A}_i \in \Upsilon$ corresponds a split function $s_i(\cdot)$ such that $s_i(k)$ indicates the number of splits needed to obtain the interval $A_{i,k}$ for $k = 0, ..., N-1$. For example, in figure 1, $s_0(\cdot) \equiv 0$ for the root A_0. As for A_7, we have $s_7(0) = 2$, $s_7(1) = 1$ and $s_7(k) = 0$ for $k \geq 2$.

- Denote $d_{i,k}$, for $k = 0, ..., N-1$, the diameter of the interval $A_{i,k}$ of some node $\mathbb{A}_i \in \Upsilon$. In particular,
$$d_{i,k} = \frac{D}{M^{s_i(k)}}.$$

- The depth of a node $\mathbb{A}_i$, denoted p_i, is the total number of splits effectuated to obtain this node:
$$p_i := \sum_{k=0}^{N-1} s_i(k). \tag{18}$$
By $\texttt{Depth}(\Upsilon)$, we designate the maximum depth in the tree Υ.

- A node $\mathbb{A}_i$ is a tree leaf if it has not been expanded. The set of the tree leaves is denoted by Λ.

- Finally, we denote by Λ_p the set of leaves of Υ at a depth equal to $p \in \mathbb{N}$: $\Lambda_p := \Big\{ \mathbb{A}_i \in \Upsilon \quad s.t. \quad p_i = p \Big\}$.

Remark 4.1. *By selecting controls at the intervals centers and by taking M odd, we guarantee that after expanding a node $\mathbb{A}_i$ we generate at least one node $\mathbb{A}_j$ with $J(x,z,a_j) \leq J(x,z,a_i)$. Indeed, the middle child $\mathbb{A}_j$ contains the same control sequence as $\mathbb{A}_i$ hence $J(x,z,a_j) = J(x,z,a_i)$.*

Proposition 4.2. *By the tree construction, there exists at least a leaf node $\mathbb{A}_i \in \Lambda$ containing an optimal control sequence and verifying:*

$$J(x,z,a_i) - \sigma_i \leq W(x,z) \leq J(x,z,a_i), \tag{19}$$

where a_i is the sample control sequence in $\mathbb{A}_i$ and

$$\sigma_i := \frac{1}{2}\Big(\sum_{k=0}^{N-1} \beta_k d_{i,k}\Big) \bigvee \Big(\sum_{k=0}^{N-1} \gamma_k d_{i,k}\Big), \tag{20}$$

with β_k and γ_k are given in proposition 3.3.

Proof. Since the set of leaves covers, at any level, the entire space A^N, there exists at least a leaf node $\mathbb{A}_i \in \Lambda$ that contains an optimal control sequence $a_i^* = (a_{i,k}^*)_k \in \mathbb{A}_i$. From Proposition 3.3 and since $W(x,z) = J(x,z,a_i^*)$, we have:

$$|J(x,z,a_i) - W(x,z)| \leq \Big(\sum_{k=0}^{N-1} \beta_k \|a_{i,k} - a_{i,k}^*\|\Big) \bigvee \Big(\sum_{k=0}^{N-1} \gamma_k \|a_{i,k} - a_{i,k}^*\|\Big).$$

On the other hand, since $a_i = (a_{i,k})_k$ is chosen at the center of the interval $\mathbb{A}_i$, we obtain :

$$\forall k = 1, \cdots, N \quad \|a_{i,k} - a_{i,k}^*\| \leq \frac{d_{i,k}}{2}, \quad \text{and} \quad |J(x,z,a_i) - W(x,z)| \leq \sigma_i.$$

□

In the optimistic planning algorithms, at each iteration, one or several optimistic nodes are chosen and split to get from each node M children ($M > 1$ is a fixed parameter of the algorithm). The choice of the optimistic nodes is based on some criterion that we explain hereafter.

To expand a node $\mathbb{A}_i$, we choose an interval from $A_{i,0} \times A_{i,1} \times ... \times A_{i,N-1}$ and we partition it uniformly to M sub-intervals. If we choose to split the interval $A_{i,k}$, for some $k = 0, ..., N-1$, then we will generate M nodes with an error term $\sigma_i^+(k)$ defined by:

$$\sigma_i^+(k) := \Big(\sum_{j=0, j\neq k}^{N-1} \beta_j d_{i,j} + \beta_k \frac{d_{i,k}}{M}\Big) \bigvee \Big(\sum_{j=0, j\neq k}^{N-1} \gamma_j d_{i,j} + \gamma_k \frac{d_{i,k}}{M}\Big).$$

Henceforth, in order to minimize the error $\sigma_i^+(k)$, the best choice of the interval to split, κ_i^*, is given by:

$$\kappa_i^* \in \underset{0\leq k\leq N-1}{\operatorname{argmin}} \ \sigma_i^+(k). \tag{21}$$

The following result gives an upper bound on the error term σ_i, of any node $\mathbb{A}_i \in \Upsilon$. This result will be used later in the convergence analysis of OP and SOP algorithms. The proof is given in Appendix 6.

Theorem 4.3. *Assume that $M > L_{F,x} > 1$ and let $N \geq 2$. Let $\tau := \left\lceil \frac{\log M}{\log(L_{F,x})} \right\rceil$. Consider a node $\mathbb{A}_i$ at some depth $p_i = p$. For p large enough, the error σ_i (defined in (20)) is bounded as follows:*

$$\sigma_i \leq \delta_p := \ c_1(N)\, \Delta t\, M^{-\frac{p}{N}}, \tag{22}$$

where $c_1(N) := C \frac{1}{1-\frac{M^{1/\tau}}{L_{F,x}}} M^{q(N)}$, $q(N) := 2 - (N-1)\frac{\tau-2}{2\tau(\tau-1)}$, and $C \geq 0$ independent of N and p.

Notice that, by definition, we have $\tau \geq 2$, and $L_{F,x}^{-1} M^{1/\tau} \leq 1$. Hence (22) is meaningful only when the strict inequality $L_{F,x}^{-1} M^{1/\tau} < 1$ holds. Moreover, $q(N) \leq 2$, hence $c_1(N)$ is bounded independently of N.

We introduce some additional definitions that will be useful for the presentation of the algorithms and for their analysis. For a given depth $p \in \mathbb{N}$, we define the set of nodes that will be expanded by optimistic planning algorithms:

$$\Upsilon_p^* := \{\mathbb{A}_i \in \Upsilon \ \text{ at depth } \ p \ \mid \ J(x, z, a_i) - \delta_p \leq W(x, z)\},$$

where δ_p is defined as in (22). The set containing all expanded nodes : $\Upsilon^* := \bigcup_{q\geq 0} \Upsilon_q^*$ is in general smaller than the whole tree Υ.

Definition 4.4. *For a given depth $p \in \mathbb{N}$, we define the asymptotic branching factor m, as the smallest real $m \in [1, M]$ such that*

$$\exists R \geq 1, \ \forall p \geq 0, \ |\Upsilon_p^*| \leq R m^p. \tag{23}$$

Remark 4.5. *The asymptotic branching factor m lies in $[1, M]$ because at any depth p, there is at least one node in Υ_p^* (the one containing the optimal solution) and at most M^p. The factor m is a measure of the complexity of the optimistic algorithm. Indeed, the nodes that will be expanded are contained in Υ_p^*, whose size is bounded by Rm^p. A problem with low resolution complexity will correspond to a small value of m. The complexity constant R (in front of the main growth factor m^p) will be used in the convergence analysis of the algorithms later on.*

4.2. Optimistic Planning (OP) Algorithm

In this section, we will present the rules for refining the search of an optimal control strategy. In the first algorithm, at each iteration, the node $\mathbb{A}_{i^*}$ minimizing the lower bound $(J(x,z,a_i)-\sigma_i)$ will be selected and split to M children. More precisely, we identify an interval $A_{i^*,\kappa^*_{i^*}}$ whose partition in M sub-intervals will produce the lowest error $\sigma_{i^*}(\kappa^*_{i^*})$.

Algorithm 1: Optimistic Planning (OP)

Require: The number of intervals N, the split factor M, the maximal number of expanded nodes $I_{\max}$

1: Initialize Υ with a root $\mathbb{A}_0 := A^N$ and $n=0$ ($n :=$ number of expanded nodes).
2: **while** $n < I_{\max}$ **do**
3: Select an optimistic node to expand: $\mathbb{A}_{i^*} \in \underset{\mathbb{A}_i \in \Lambda}{\operatorname{argmin}} (J(x,z,a_i)-\sigma_i)$.
4: Select $\kappa^*_{i^*}$, defined in (21), the interval to split for the node $\mathbb{A}_{i^*}$.
5: Update Υ by expanding $\mathbb{A}_{i^*}$ along $\kappa^*_{i^*}$ and adding its M children.
6: Update $n = n+1$.
7: **end while**
8: **return** Control sequence $a_{i^*} = (a_{i^*,k})_k \in A^N$ of the node $\mathbb{A}_{i^*} \in \underset{\mathbb{A}_i \in \Lambda}{\operatorname{argmin}} J(x,z,a_i)$ and $J^*(x,z) := J(x,z,a_{i^*})$.

In this algorithm, the number of expanded nodes corresponds to the number of iterations, since at each iteration only one node is expanded. The number $I_{\max}$ represents a maximum available computational resource.

Lemma 4.6. *(See [9, Proposition 3]) The OP algorithm expands only nodes satisfying $J(x,z,a_i)-\sigma_i \leq W(x,z)$ (thus only nodes in Υ^*). Furthermore, the returned value $J(x,z,a_{i^*})$ satisfies $0 \leq J(x,z,a_{i^*}) - W(x,z) \leq \sigma_{\min}$ where $\sigma_{\min}$ corresponds to the smallest computed value σ_i among all the expanded nodes.*

From Theorem 4.3 and Lemma 4.6, we derive the following result whose proof is given in Appendix 6.

Theorem 4.7. *Assume that $M > L_{F,x} > 1$ and let $N \geq 2$. Let a_{i^*} and $J(x,z,a_{i^*})$ be the output of the OP algorithm, and let $n \geq 1$ be the corresponding number of expanded nodes. For n large enough, the following error bound holds:*

$$0 \leq J(x,z,a_{i^*}) - W(x,z) \leq \mathcal{E}_{OP}(N,n) := \begin{cases} c_2(N)\, n^{-\frac{\log(M)}{N\log(m)}} & \text{if } m > 1, \\ c_2(N)(\log(n))^{-\frac{\log(M)}{NR}} & \text{if } m = 1, \end{cases} \tag{24}$$

where (m,R) satisfy (23), $c_2(N) := c_1(N) M^{\frac{1}{N} - \frac{\log(\frac{(m-1)}{R})}{N\log(m)}}$ (for $m>1$), or $c_2(N) := c_1(N) M^{\frac{1}{N}}$ (for $m=1$), and where $c_1(N)$ is the same constant as in Theorem 4.3.

As already mentioned, $c_1(N)$ is bounded independently of N, and so is the constant $c_2(N)$. From Theorem 4.7 we deduce that the error, between the optimal cost computed by OP algorithm and the (global) optimal value, converges to zero as $n \to \infty$ (for a fixed number of time steps N).

Remark 4.8. *The bound $\mathcal{E}_{OP}$ in Theorem 4.7 improves slightly the error estimates given in [9]. Indeed, in [9, Theorem 6], when $m > 1$, the error is bounded by $O(\gamma^{C\sqrt{\log(n)}})$ for some $\gamma \in (0,1)$ (discount factor) and some constant $C > 0$, while in Theorem 4.7, $\mathcal{E}_{OP}$ is of order $O(\beta^{log(n)})$, for some constant $\beta \in (0,1)$, which is therefore asymptotically lower.*

4.3. Simultaneous Optimistic Planning (SOP) Algorithm

The SOP algorithm expands at each iteration several nodes which are supposed to be optimistic.

Algorithm 2: Simultaneous Optimistic Planning (SOP)

Require: The number of intervals N, the split factor M, a maximal number of expanded nodes $I_{\max}$ and a maximal depth function $p_{\max}(.) \geq 1$.

Initialize Υ with root $\mathbb{A}_0 := A^N$ and set $n = 0$ ($n :=$ number of expanded nodes).
while $n < I_{\max}$ **do**
  $p = \min_{\mathbb{A}_i \in \Lambda} p_i$: the minimal depth among the tree leaves of Υ.
  while $p \leq p_{\max}(n)$ **do**
    Select an optimistic node at depth p: $\mathbb{A}_{i^*} \in \operatorname{argmin}_{\mathbb{A}_i \in \Lambda_p} J(x, z, a_i)$.
    Select $\kappa_{i^*}^*$, as defined in (21), the interval to split for the node $\mathbb{A}_{i^*}$.
    Update Υ by expanding $\mathbb{A}_{i^*}$ along $\kappa_{i^*}^*$ and adding its M children at depth $p+1$.
    Update $p \leftarrow p+1$ and $n \leftarrow n+1$
  end while
end while
return Control sequence $a^* := (a_{i^*,k})_k \in A^N$ of the node $\mathbb{A}_{i^*} = \operatorname{argmin}_{\mathbb{A}_i \in \Lambda} J(x, z, a_i)$ and $J^*(x, z) := J(x, z, a^*)$.

In Algorithm 2, $p_{\max}(n)$ denotes a maximal depth that the tree should not exceed, at iteration n. As in Algorithm 1, $I_{\max}$ represents a maximum available computational resource.

The following result gives a lower bound on the depth of the optimal node generated by the SOP algorithm. This is a direct consequence of [18, Lemma 4.1]. For convenience of the reader, a sketch of the proof is given in Appendix 6.

Lemma 4.9. *Let $p_{\max}(n) \geq 1$. Let $p(n) \geq 1$ be the smallest integer $p \geq 1$ such that:*

$$p_{\max}(n) \sum_{q=0}^{p} R m^q \geq n \tag{25}$$

and let $p_n^* := \min\{p(n)-1, p_{\max}(n)\}$. *Then the SOP algorithm, after* n *iterations, has expanded an optimal node at depth* p_n^* *and the solution obtained has an error bounded by* $\delta_{p_n^*}$ *(recall that* δ_p *is defined in* (22)*).*

The SOP algorithm gives a sub-optimal solution with an upper bound estimate given in the following theorem whose proof is postponed to Appendix 6.

Theorem 4.10. *Consider the SOP algorithm with* $p_{\max}(n) := n^\eta$ *for some given* $\eta \in]0,1[$. *Let* n *be a given number of expanded nodes in the SOP algorithm and let* $J^*(x,z)$ *be the corresponding returned value. The following upper bound holds as* $n \to \infty$*:*

$$0 \leq J^*(x,z) - W(x,z) \leq \mathcal{E}_{SOP}(N,n) := \begin{cases} c_3(N)\,\Delta t\, n^{-\frac{(1-\eta)}{\log(m)}\frac{\log(M)}{N}} & \text{if } m > 1 \text{ and } \eta \in]0,1[\\ c_3(N)\,\Delta t\, M^{-\frac{n^{1-\eta}}{R}} & \text{if } m = 1 \text{ and } \eta \in [\frac{1}{2},1[, \end{cases}$$

with $c_3(N) := c_1(N) M^{\frac{\log(R)}{N\log(m)}+\frac{2}{N}}$ *if* $m > 1$ *and* $c_3(N) := c_1(N) M^{-\frac{2}{N}}$ *if* $m = 1$. *In particular, the error of the SOP algorithm converges to zero as* $n \to \infty$ *(for a fixed number of time steps* N*).*

Remark 4.11. *The SOP estimate* $\mathcal{E}_{SOP}$*, given in Theorem 4.10, improves the estimate presented in [10, Theorem 11]. Firstly, one can choose* $p_{\max}(n) := \sqrt{n}$ *for all possible values of the branching factor* $m \in [1, M]$ *(which is not the case of [10, Theorem 11]). Secondly, in the case when* $m > 1$*, we obtain an error of* $O(\beta^{\log(n)})$ *for some constant* $\beta \in (0,1)$*. This error is therefore asymptotically lower than the error of [10, Theorem 11], that is of order* $O(\beta'^{\sqrt{\log(n)}})$ *for some* $\beta' \in (0,1)$*. In the case* $m = 1$*, our error can be written* $O(\beta^{n^{1/2}})$ *for some* $\beta \in (0,1)$*, while in [10, Theorem 11] the bound is of order* $O(\beta'^{n^{1/6}})$ *for some* $\beta' \in (0,1)$ *(and* $p_{\max} = n^{1/3}$*).*

4.4. Simultaneous Optimistic Planning with Multiple Steps (SOPMS) Algorithm

We start by introducing algorithm 3 (`SOP-tree-update`) which is a generic elementary algorithm that describes how some given tree Υ' should be updated in a similar way to SOP.

The SOPMS algorithm uses the elementary algorithm `SOP-tree-update` in order to optimize the objective function from an initial state $x \in \mathbb{R}^d$ and an auxiliary variable $z \in \mathbb{R}$. In this algorithm, we will need the definition of the following cost function J_k, starting from a time step k, for any $k = 0, ..., N-1$, as follows:

$$J_k(y,z,a) := \Big(\sum_{i=k}^{N-1} \rho(y_i^a, a_i) + \Phi(y_N^a) - z \Big) \bigvee \Big(\max_{k \leq i \leq N} g(y_i^a) \Big) \bigvee \Psi(y_N^a),$$

Algorithm 3: `SOP-tree-update`$(\Upsilon', I, p_{\max}(\cdot), J_k(y,z,\cdot))$

Require: A tree Υ', a maximal number of expanded nodes I and a maximal depth function $p_{\max}(\cdot)$.

Initialize $n = 0$ ($n :=$ number of expanded nodes).
while $n < I$ **do**
  $p = \min_{\mathbb{A}_i \in \Lambda} p_i$: the minimal depth among the tree leaves.
  while $p \leq p_{\max}(n)$ **do**
    Select an optimistic node at depth p: $\mathbb{A}_{i^*} \in \operatorname{argmin}_{\mathbb{A}_i \in \Lambda_p} J_k(x,z,a_i)$.
    Select $\kappa^*_{i^*}$, defined in (21), the interval to split for the node $\mathbb{A}_{i^*}$.
    Update Υ' by expanding $\mathbb{A}_{i^*}$ along $\kappa^*_{i^*}$ and adding its M children at depth $p+1$.
    Update $p \leftarrow p+1$ and $n \leftarrow n+1$
  end while
end while
return Control sequence a_{i^*} of the node $\mathbb{A}_{i^*} = \operatorname{argmin}_{\mathbb{A}_i \in \Lambda} J_k(y,z,a_i)$ and $J^*_k := J_k(y,z,a_{i^*})$.

where $y \in \mathbb{R}^d$, $z \in \mathbb{R}$, $a := (a_i)_i \in A^{N-k}$ and $(y^a_i)_i$ is the discrete trajectory starting from y and associated to the control sequence $(a_i)_i$.

Algorithm 4: Simultaneous Optimistic Planning with Multiple Steps (SOPMS)

Require: The number of intervals N, the split factor M, a maximal number of expanded nodes $I_{\max}$, a local initial number of expanded nodes $I_{eval,0}$, a maximal depth function $p_{\max}(\cdot)$, a tolerance $\epsilon > 0$

Initialize $k \leftarrow 0$, $y_0 = x$ and $n \leftarrow 0$ ($n :=$ total number of expanded nodes)
while $k \leq N-1$ **do**
  Initialize Υ_k with controls $a^k := (a^k_i)_{k \leq i \leq N-1} \in A^{N-k}$, root $\mathbb{A}_0 := \prod_{i=k}^{N-1} \overline{A_i}$, and $W_k := J_k(y_k, z, a^k)$ (see Remark 4.12 for the definition of $\overline{A_i}$)
  $\ell := 0$
  loop
    $I_{eval} := 2^\ell I_{eval,0}$
    $(a^*, W^{temp}_k) \leftarrow$ `SOP-tree-update`$(\Upsilon_k, I_{eval}, p_{\max}(\cdot), J_k(y_k,z,\cdot))$: expand nodes of Υ_k
    update n ($n \leftarrow n + I_{eval}$)
    if $|\frac{W^{temp}_k - W_k}{W_k}| \leq \epsilon$ **and** $\ell \geq 1$ **then**
      Accept the first component $a^*_k \in A$ of the control input a^*, set $y_{k+1} = F(y_k, a^*_k)$
      $k \leftarrow k+1$, and **go to** 2
    end if
    $W_k \leftarrow W^{temp}_k$
    $\ell \leftarrow \ell + 1$
    if $n \geq I_{\max}$ **then**
      Accept the control sequence $(a^*_k, a^*_{k+1}, ..., a^*_{N-1}) \in A^{N-k}$ and **go to** 20
    end if
  end loop
end while
return Control sequence $(a^*_k)_k \in A^N$, value $J^*(x,z) = J(x,z,a^*)$, budget n.

Remark 4.12. (Initialisation step) *In the initialization step of SOPMS algorithm (line 3), when $k = 0$, the initial control interval sequence is $\mathbb{A}_0 := [0,1]^N$. Then, for $k \geq 1$, the initial control $(a_j)_{j\geq k}$ is taken as the computed control at iteration $k-1$ ($a_j := a_j^*$ for $j \geq k$) and we choose the corresponding initial intervals as the intervals centered in a_i and of maximal possible diameter, contained in $[0,1]$ (that is, $\mathbb{A}_0 := \prod_{i=k}^{N-1} \overline{A_i}$, with $\overline{A_i} := [a_i - r_i, a_i + r_i]$ and $r_i := \min(a_i, 1-a_i)$).*

At each time step $k = 0, ..., N-1$, the SOPMS algorithm optimizes the cost functional J_k over control sequences in a subset of A^{N-k}. When the cost functional cannot be improved anymore (line 9), up to a relative threshold ϵ, the optimization procedure is stopped for the current time step, the first control component (as well as the first position) is accepted and we move forward to the next time step. The remaining control sequence is utilized as a starting point for the next iteration.

We do not have a convergence proof for the SOPMS algorithm. However, because of the presence of a tolerance parameter used to stop the successive computations, we will numerically observe a reduced number of expanded nodes n (compared to SOP algorithm) and hence a reduced complexity of the overall algorithm, leading to good precision up to a very small residual error and for a reduced budget (see Section 5.1).

Notice that by choosing $I_{eval,0} = I_{\max}$, SOPMS becomes equivalent to SOP and provides the same error estimate.

4.5. Resolution procedure for a state-constrained problem

We describe here the procedure to get an approximation of the optimal value $V(x)$, and an approximated optimal trajectory for the state-constrained problem (10) starting from an initial state x. Let us first remark that the function $z \to W(x,z)$ is a non-increasing function. By using the properties of the functions F, ρ, Φ, g, and Ψ, one can get two bounds $Z_{\min}$ and $Z_{\max}$, such that: $V(x) = \inf\{z \in [Z_{\min}, Z_{\max}] \mid W(x,z) \leq 0\}$.

The computation of an approximation of $V(x)$ will be performed by a dichotomy algorithm combined to an optimistic method for evaluating $W(x,z)$ up to a prescribed precision $\varepsilon > 0$. First, notice the the value $Z_{\min}$ can always be chosen such that $W(x, Z_{\min}) > \epsilon$. We assume that $Z_{\max}$ can also be chosen such that $W(x, Z_{\max}) \leq \epsilon$ (if this is not the case, it means that there is no admissible trajectory that satisfies the constraints and $V(x) = +\infty$).

Remark 4.13. *In Algorithm 5 (line 3), we need to compute $J(x,c,a^c)$ up to a precision ϵ. To fulfill this requirement, a sufficiently large computational budget should be allowed in the optimistic algorithm in order to obtain a control strategy a^c such that the error $J(x,c,a^c) - W(x,c)$ is lower than ϵ. In practice, this budget can be large, and difficult to evaluate from the theoretical estimates obtained in Theorems 4.7 and 4.10. In the*

Algorithm 5: dichotomy algorithm combined with OP (resp. SOP/SOPMS) method

Require: A threshold $\epsilon > 0$, $Z_{\min} < Z_{\max}$ such that $W(x, Z_{\min}) > \epsilon$, $W(x, Z_{\max}) \leq \epsilon$

$b_\ell \leftarrow Z_{\min}$, $b_r \leftarrow Z_{\max}$

while $b_r - b_\ell > \epsilon$ **do**

  $c \leftarrow (b_\ell + b_r)/2$; Compute a strategy a^c by OP (or SOP/SOPMS) such that $J(x, c, a^c) - W(x, c) \leq \epsilon$ (see Remark 4.13);

  if $J(x, c, a^c) > \epsilon$, set $b_\ell := c$;

  if $J(x, c, a^c) \leq \epsilon$, set $b_r := c$;

end while

return $z^\epsilon = b_r$, the value $J(x, z^\epsilon, a^\epsilon)$ along with its corresponding optimal control sequence $a^\epsilon := (a^\epsilon_k)_k$ and optimal trajectory $y^\epsilon := (y^\epsilon_k)_k$.

sequel, we will assume that the computation budget is large enough and we will study the asymptotic behavior of the method when the threshold parameter ϵ converges to zero.

Let us first introduce the following problem with relaxed constraints:

$$V^\epsilon(x) := \inf_{a \in A^N} \left\{ \sum_{k=0}^{N-1} \rho(y_k^{x,a}, a_k) + \Phi(y_N^{x,a}) \mid \max\left(\Psi(y_N^{x,a}), \max_{0 \leq k \leq N} g(y_k^{x,a})\right) \leq \epsilon \right\}. \tag{26}$$

Lemma 4.14. *Assume that the dichotomy algorithm is combined with OP (resp. SOP). Assume that the computation budget is large enough so that $\mathcal{E}_{\texttt{OP}}(N, n) \leq \epsilon$ (resp. $\mathcal{E}_{\texttt{SOP}}(N, n) \leq \epsilon$). Then the output $z = z^\epsilon$ of the dichotomy algorithm satisfies*

$$-\epsilon < W(x, z^\epsilon) \leq \epsilon, \quad V^\epsilon(x) \leq z^\epsilon + \epsilon, \quad \text{and} \quad V(x) \geq z^\epsilon - \epsilon.$$

Proof. Assume that the dichotomy algorithm is combined with OP method and the calculations are performed with $\mathcal{E}_{\texttt{OP}}(N, n) \leq \epsilon$. The output $z^\epsilon = b_r$ of the algorithm satisfies $W(x, z^\epsilon) \leq J(x, z^\epsilon, a^\epsilon) = J(x, z^\epsilon, a^\epsilon) \leq \epsilon$. Moreover at the last iteration of the algorithm, we have $J(x, b_\ell, a^b_\ell) > \epsilon$ which leads to $W(x, b_\ell) \geq J(x, b_\ell, a^b_\ell) - \mathcal{E}_{\texttt{OP}} > 0$. Since $W(x, \cdot)$ is 1-Lipschitz, we obtain that $W(x, z^\epsilon) \geq W(x, b_\ell) - |z^\epsilon - b_\ell| > -\epsilon$.

Now, $W(x, z^\epsilon) \leq \epsilon$ implies that:

$$\sum_{k=0}^{N-1} \rho(y_k^\epsilon, a_k^\epsilon) + \Phi(y_N^\epsilon) - z^\epsilon \leq \epsilon \qquad \text{and} \qquad \max\left(\Psi(y_N^\epsilon), \max_{0 \leq k \leq N} g(y_k^\epsilon)\right) \leq \epsilon.$$

Hence the trajectory y^ϵ is feasible for problem $V^\epsilon(x)$, and $V^\epsilon(x) \leq z^\epsilon + \epsilon$. Finally, since $W(x, b_\ell) > 0$ and because the application $z \longmapsto W(x, z)$ is non-increasing, we deduce that $b_\ell < V(x)$. This inequality along

with the fact that $b_r - b_\ell \leq \epsilon$ lead to $V(x) > b_\ell \geq b_r - \epsilon = z^\epsilon - \epsilon$. □

In the following, $\mathcal{O}(x)$ denotes the set of optimal solutions associated to the exact value $V(x)$. We define also the distance to set $\mathcal{O}(x)$ by: $d(y, \mathcal{O}(x)) := \inf\{\max_{0\leq k\leq N} \|y_k - \bar{y}_k\|,\ \bar{y} \in \mathcal{O}(x)\}$.

Theorem 4.15. *Let $x \in \mathbb{R}^d$ such that $\mathcal{O}(x) \neq \emptyset$. Let $(y^\epsilon, a^\epsilon, z^\epsilon)$ be the outcome of Algorithm 5 for $\epsilon > 0$. Then,*

(i) $\lim_{\epsilon\to 0} d(y^\epsilon, \mathcal{O}(x)) = 0$.

(ii) $\lim_{\epsilon\to 0} z^\epsilon = \lim_{\epsilon\to 0} V^\epsilon(x) = V(x)$;

Proof. *(i)* Assume that $\limsup_{\epsilon\to 0} d(y^\epsilon, \mathcal{O}(x)) > 0$. Then, by compactness of A and (A1), there exists a subsequence (ϵ_n), $\bar{z} \in \mathbb{R}$ and a trajectory $\bar{y}$ such that z^{ϵ_n} converges to $\bar{z}$, y^{ϵ_n} converges to $\bar{y}$ and $d(\bar{y}, \mathcal{O}(x)) = \lim_{\epsilon_n\to 0} d(y^{\epsilon_n}, \mathcal{O}(x)) > 0$. However, from the inequality $W(x, z^{\epsilon_n}) \leq \epsilon_n$, we get $\max\left(\Psi(y_N^{\epsilon_n}), \max_{0\leq k\leq N} g(y_k^{\epsilon_n})\right) \leq \epsilon_n$. Therefore, by passing to the limit we obtain $\max\left(\Psi(\bar{y}_N), \max_{0\leq k\leq N} g(\bar{y}_k)\right) = 0$, which proves that $\bar{y}$ is an admissible trajectory. Moreover, $W(x, \bar{z}) = \lim_{\epsilon_n\to 0} W(x, z^{\epsilon_n}) \leq 0$. Then $V(x) \leq \bar{z}$. On the other hand, we have $V(x) \geq \lim_{\epsilon_n\to 0} (z^{\epsilon_n} - \epsilon_n) = \bar{z}$. Therefore, $V(x) = \bar{z}$ and $\bar{y} \in \mathcal{O}$, which contradicts the fact that $d(\bar{y}, \mathcal{O}(x)) > 0$. We conclude that the claim (i) holds.

(ii) From the previous Lemma, we have: $V^\epsilon - \epsilon \leq z^\epsilon \leq V(x) + \epsilon$. By classical compactness arguments of A^N, we conclude the desired result. □

5. Numerical simulations

Computations for Examples 1, 2 and 4 where realized with a c++ gnu 6.5.0 on a 2.2 GHz Intel i7 64 bits computer (with 16 GB RAM). Computations for Example 3 where realized using c++ gnu on a 2.4 GHz Intel XEON E5-2695 CPU 64 bits computer (with 128 GB RAM).

5.1. Example 1: a 2D example without state constraints

We first consider a simple example with no state constraints. Here, the dynamics f is defined by:

$$f((x_1, x_2), a) = \begin{pmatrix} -x_2 \\ a \end{pmatrix}, \qquad \text{for } (x_1, x_2) \in \mathbb{R}^2, \quad a \in A := [-1, 1].$$

The time horizon is set to $T = 1$, and the distributed and final cost functions are defined by $\ell(x, a) = 0$ and $\Phi(x) = \|x\|$. In this context, the structure of the exact solutions of the continuous and discrete problems

are known[2]. Table 1 gives some reference values for the exact value $\vartheta(x)$ of the time-continuous problem (second column) for three initial positions $x = (1, 0)$, $x = (1.5, 0)$ and $x = (2, 0)$. The values $V(x)$ of the corresponding discretized problem are also reported, for different values of N, in the third column of Table 1.

x	$\vartheta(x)$	$V(x)$		
		$N = 10$	$N = 20$	$N = 40$
$(2.0, 0)$	1.638580	1.639542	1.639149	1.638626
$(1.5, 0)$	1.159678	1.160946	1.160262	1.159701
$(1.0, 0)$	0.687630	0.689017	0.688202	0.687676

Table 1: Some reference values (rounded to 6 decimal places) for the continuous and discretized problems.

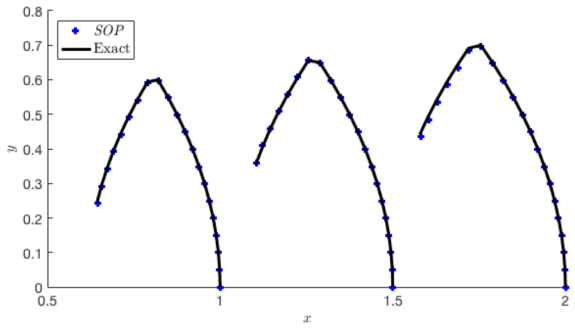

Figure 2: (Example 1) Trajectories obtained by the SOP algorithm for $N = 20$, $I_{\max} = 10^4$, from different initial positions.

We use the optimistic algorithms to compute an approximation $J^*(x)$ of the optimal value $V(x)$, for different values of N. In these simulations, the parameters used are $I_{eval,0} := 10$ and $\epsilon := 10^{-5}$ for SOPMS, and the function $p_{\max}(n) := 5\sqrt{n}$ for SOP. This choice of $p_{\max}(n)$ comes from the analysis of Remark 4.11. Note that for SOPMS algorithm, the number of expanded nodes, n, may differ from the budget I_{max} (while for OP and SOP we have always $n = I_{max}$). The optimal trajectories of the discretized problem, with $N = 20$, are displayed in Figure 2 and compared to the solutions computed by SOP with $I_{\max} = 10^4$. From this figure, one can see that the computed solutions are very close to the exact ones. Besides, Figure 3 shows the error $|J^*(x) - V(x)|$ with respect to the number of expanded nodes n (left graphics) and with respect to the CPU time (right graphics), for algorithms OP, SOP and SOPMS, in the case when $N = 20$. Figure 3 confirms that the SOP algorithm is much more efficient than OP in term of accuracy and CPU time. We

[2]For the continuous problem, with an initial position $x_0 = (x_{01}, 0)$, the optimal control law satisfies: $a(t) = 1$ on $[0, t^*[$, and $a(t) = -1$ on $]t^*, T]$, where the switching time $t = t^*$ is a root of the equation $((2t-T)-t)(\frac{1}{2}(T-2t)^2 - t^2 + x_{01}) + (2t-T) = 0$. The optimal control sequence $(a_k^*)_k$ of the discrete problem has the following structure: $a_k^* = 1$ for $k < \overline{k}$,

notice also that SOPMS algorithm has very interesting performances and in particular in term of CPU time.

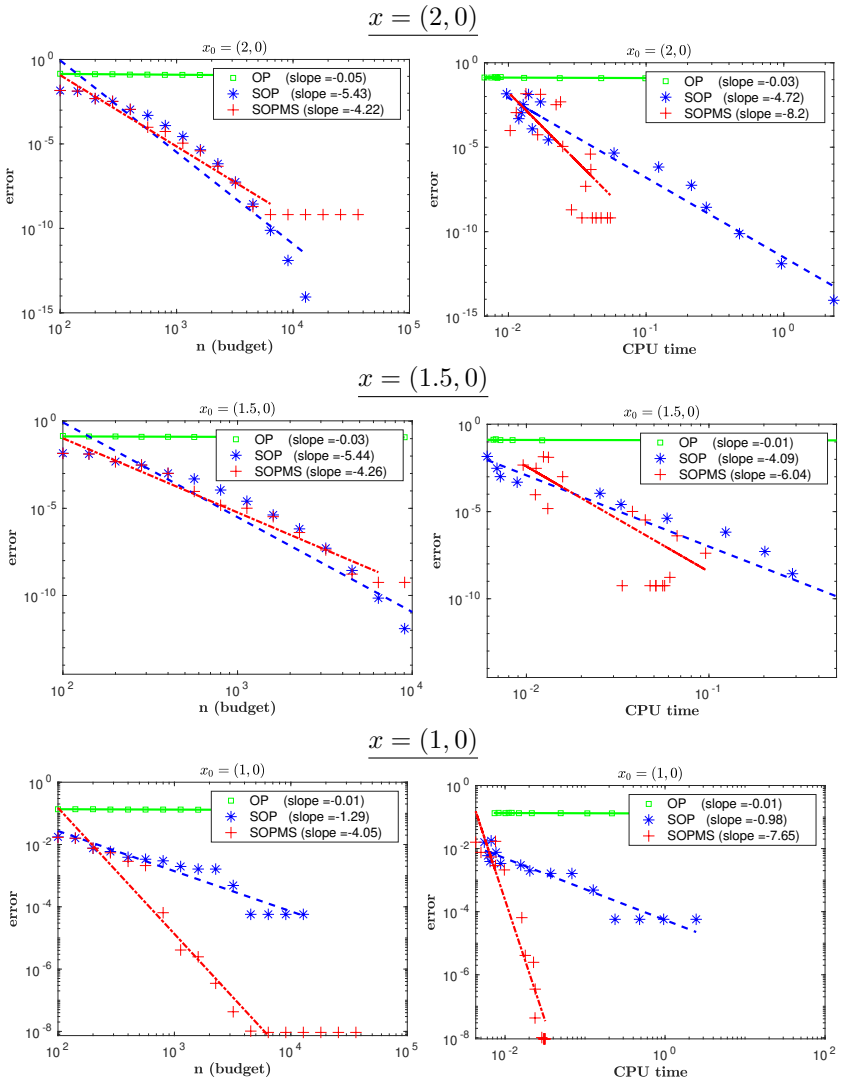

Figure 3: (Example 1) Error versus budget - number of expaned nodes - n (left) and versus CPU time (right) for OP, SOP and SOPMS algorithms, with $N = 20$ time steps, for initial point $x_0 = (2, 0)$ (top), $x_0 = (1.5, 0)$ (middle), $x_0 = (1, 0)$ (bottom).

Tables 2, 3 and 4 give further information on the results of OP, SOP and SOPMS methods respectively. In particular, the computed optimal values, the error and the CPU time are given as the budget I_{max} increases (exponentially), for the points $x = (1, 0)$ and $x = (2, 0)$ (the results for $x = (1.5, 0)$ are similar to the case of $x = (2, 0)$ and have been omitted). We chose not to report the errors and the optimal values when the computational time (CPU) exceeds 100s.

		$x=(1,0)$			$x=(2,0)$		
N	$I_{\max}$	$J^*(x)$	$J^*(x)-V(x)$	CPU(s)	$J^*(x)$	$J^*(x)-V(x)$	CPU(s)
10	400	0.792857	1.03e-01	0.011	1.739936	1.00e-01	0.016
	1600	0.750902	6.18e-02	0.086	1.701032	6.14e-02	0.086
	6400	0.732020	4.30e-02	1.751	1.681600	4.20e-02	1.797
	25600	0.729926	4.09e-02	92.594	1.677283	3.77e-02	97.685
	102400	-	-	$\geq$100.000	-	-	$\geq$100.000
20	400	0.820738	1.32e-01	0.013	1.765899	1.26e-01	0.014
	1600	0.819215	1.31e-01	0.098	1.759675	1.20e-01	0.096
	6400	0.817693	1.29e-01	1.859	1.759675	1.20e-01	1.809
	25600	-	-	$\geq$100.000	-	-	$\geq$100.000
	102400	-	-	$\geq$100.000	-	-	$\geq$100.000
40	400	0.853756	1.66e-01	0.017	1.822498	1.83e-01	0.014
	1600	0.840536	1.52e-01	0.167	1.806458	1.67e-01	0.112
	6400	0.830636	1.43e-01	3.610	1.786826	1.48e-01	1.950
	25600	-	-	$\geq$100.000	-	-	$\geq$100.000
	102400	-	-	$\geq$100.000	-	-	$\geq$100.000

Table 2: (Example 1) Values and errors for the OP algorithm with different parameters N and I_{max}, for $x=(1,0)$ and $x=(2,0)$.

Table 2 shows that, as expected, the convergence of OP method is very slow, and even an accuracy of order 10^{-2} requires huge computational resources, especially when N is large. On the other hand, the CPU time results given in Tables 2 & 3 show that SOP is faster than OP for a same $I_{\max}$ value. Indeed, each iteration of OP selects the node to expand among all the tree leaves, while an SOP iteration selects the nodes only among the leaves of a given depth of the tree. A first glance to Table 4 shows that SOPMS algorithm costs less CPU time than SOP (recall that for this simulation, the tolerance used in SOPMS is $\epsilon = 10^{-5}$). The SOP algorithm constructs a tree of $(M-1)I_{\max}+1$ leaves with control and trajectory sequences of lengths N and $N+1$ respectively. For any $k \geq 1$, SOPMS uses a tree of fewer leaves compared to SOP, with control and trajectory sequences of lengths $N-k$ and $N-k+1$ respectively. Moreover, we observe that SOPMS uses a reduced total number of nodes $n << I_{\max}$ (while for SOP, the total number of nodes coincides with $I_{\max}$) which results in reduced CPU times.

In conclusion, in this example, SOPMS performs in general better than SOP and OP. In the case of the initial point $x=(1,0)$, which happens to be numerically more difficult for OP or SOP, the SOP algorithm is limited to a precision of order 10^{-3}, for $N=40$ while SOPMS goes beyond this limitation and reaches an accuracy of about 10^{-10} in less than 1s.

		$x=(1,0)$			$x=(2,0)$		
N	$I_{\max}$	$J^*(x)$	$J^*(x)-V(x)$	CPU(s)	$J^*(x)$	$J^*(x)-V(x)$	CPU(s)
10	400	0.689020	3.85e-06	0.009	1.639547	4.31e-06	0.026
	1600	0.689017	6.23e-11	0.068	1.639543	7.30e-11	0.060
	6400	0.689017	7.77e-16	0.588	1.639543	0.00e-16	0.627
	25600	0.689017	5.55e-16	17.393	1.639543	0.00e-16	18.596
	102400	-	-	$\geq$100.000	-	-	$\geq$100.000
20	400	0.692103	3.90e-03	0.034	1.640247	1.09e-03	0.037
	1600	0.689831	1.62e-03	0.050	1.639154	4.51e-06	0.089
	6400	0.688259	5.72e-05	0.598	1.639149	7.65e-11	0.662
	25600	0.688206	3.78e-06	17.058	1.639149	1.55e-15	17.704
	102400	-	-	$\geq$100.000	-	-	$\geq$100.000
40	400	0.707336	1.96e-02	0.022	1.653411	1.47e-02	0.019
	1600	0.692945	5.26e-03	0.070	1.639752	1.12e-03	0.073
	6400	0.691294	3.61e-03	0.765	1.638630	4.62e-06	0.798
	25600	0.690882	3.20e-03	16.385	1.638626	7.83e-11	17.713
	102400	-	-	$\geq$100.000	-	-	$\geq$100.000

Table 3: (Example 1) Values and errors for the SOP algorithm with different parameters N and I_{max}, for $x=(1,0)$ and $x=(2,0)$.

		$x=(1,0)$				$x=(2,0)$			
N	$I_{\max}$	$J^*(x)$	$J^*(x)-V(x)$	n	CPU(s)	$J^*(x)$	$J^*(x)-V(x)$	n	CPU(s)
10	400	0.689020	3.67e-06	440	0.008	1.639547	4.31e-06	440	0.024
	1600	0.689017	5.12e-10	630	0.007	1.639543	2.40e-10	630	0.012
	6400	0.689017	5.12e-10	630	0.006	1.639543	2.40e-10	630	0.006
	25600	0.689017	5.12e-10	630	0.006	1.639543	2.40e-10	630	0.006
	102400	0.689017	5.12e-10	630	0.006	1.639543	2.40e-10	630	0.006
20	400	0.691085	2.88e-03	440	0.010	1.640268	1.11e-03	440	0.011
	1600	0.688204	2.47e-06	1147	0.023	1.639153	3.91e-06	1127	0.030
	6400	0.688202	9.27e-09	1352	0.031	1.639149	6.54e-10	1332	0.034
	25600	0.688202	9.27e-09	1352	0.030	1.639149	6.54e-10	1332	0.030
	102400	0.688202	9.27e-09	1352	0.030	1.639149	6.54e-10	1332	0.028
40	400	0.707266	1.95e-02	440	0.019	1.653411	1.47e-02	440	0.026
	1600	0.692419	4.74e-03	1729	0.125	1.638652	2.66e-05	1676	0.051
	6400	0.687676	4.84e-07	3292	0.170	1.638626	4.92e-07	1930	0.070
	25600	0.687676	5.02e-11	4512	0.308	1.638626	1.13e-10	2748	0.135
	102400	0.687676	3.01e-12	5352	0.459	1.638626	1.13e-10	2748	0.135

Table 4: (Example 1) Values and errors for the SOPMS algorithm with different parameters N and I_{max}, for $x=(1,0)$ and $x=(2,0)$ (the tolerance parameter is set to $\epsilon=10^{-5}$). The value of n corresponds to the total budget used by the algorithm.

5.2. Example 2: Zermelo problem with state constraints

Consider the Zermelo problem where a boat tries to reach a circular target $\mathcal{C}$ at time $T = 1$ with minimal fuel consumption. The dynamics is given by:

$$\begin{cases} \dot{y}_1(s) = u(s)\cos(\theta(s)) + c - bx_2(s)^2, \\ \dot{y}_2(s) = u(s)\sin(\theta(s)), \end{cases}$$

where the controls are the speed $u(s)$ and the angle of orientation $\theta(s)$ of the boat (such that $(u(s), \theta(s)) \in A := [0, u_{\max}] \times [0, 2\pi]$, for a.e. $s \in [0, T]$), and the term $c - bx_2^2$ represents the current drift along the x_1-axis (with $b := 0.5$ and $c := 2$). The target $\mathcal{C}$ is centered at $(1.5, 0)$ and of radius $r_0 = 0.1$. We consider also two rectangular obstacles with horizontal and vertical half lengths (r_x, r_y) (see Figure 4). The first obstacle is centered at $(-0.5, 0.5)$ with $(r_x, r_y) = (0.4, 0.4)$, and the second obstacle is centered at $(-1, -1.5)$ with $(r_x, r_y) = (0.2, 1)$. To take into account the obstacle avoidance and the achievement of the target, we define the functions g and Ψ as follows:

$$g(x) := \Big(0.4 - \|x - (-0.5, 0.5)\|_\infty\Big) \bigvee \min\big(0.2 - |x_1 + 1|, 1 - |x_2 + 1.5|\big) \quad \text{and} \quad \Psi(x) := \|x - (1.5, 0)\|_\infty - r_0.$$

The cost to minimize is $Q(x, \alpha) := \int_0^T u(s)ds$, where x is the initial position of the boat and $\alpha(\cdot) := (u(\cdot), \theta(\cdot))$ is the control law. For a given N, the discrete control problem becomes:

$$V(x) = \inf\Big\{\frac{1}{N}\sum_{k=0}^{N-1} u_k \quad \text{with} \quad a = ((u_k, \theta_k))_k \in A^N, \quad g(y_k^a) \le 0 \quad \text{for } k = 0, \cdots, N, \quad \text{and} \quad \Psi(y_N^a) \le 0\Big\},$$

where $(y_k^a)_k$ is the discrete state variable, corresponding to the control policy $a = ((u_k, \theta_k))_k \in A^N$, and starting at the initial position x. Henceforth, the discrete auxiliary value function is defined as:

$$W(x, z) = \inf_{(a_k)_k \in A^N} \Big\{\Big(\frac{1}{N}\sum_{k=0}^{N-1} u_k - z\Big) \bigvee \Big(\max_{0 \le k \le N} g(y_k^a)\Big) \bigvee \Psi(y_N^a)\Big\}.$$

For this example, we use Algorithm 5 combined with SOP or SOPMS approach to compute, for any given $\epsilon > 0$, an approximation z^ϵ of $V(x)$ and an approximation value of $W(x, z^\epsilon)$. For all simulations in this example, the parameters used are $p_{\max}(n) := 5\sqrt{n}$ for SOP, $I_{eval,0} = 10$ and tolerance 10^{-5} for SOPMS, and the dichotomy approach is initialized with $Z_{\min} = 0$ and $Z_{\max} = 2.5$.

Figure 4 shows optimal trajectories obtained from three different initial positions $x_1 = (-2.5, -1)$, $x_2 =$

$(-2, 0.5)$ and $x_3 = (-1.5, 1.5)$. These trajectories are computed by dichotomy algorithm combined with. We remark that the computed trajectories verify the constraints, by avoiding the obstacles, and reach the target at the final time step (for $N = 10$ in figure 4(left) and $N = 40$ in in figure 4(right).

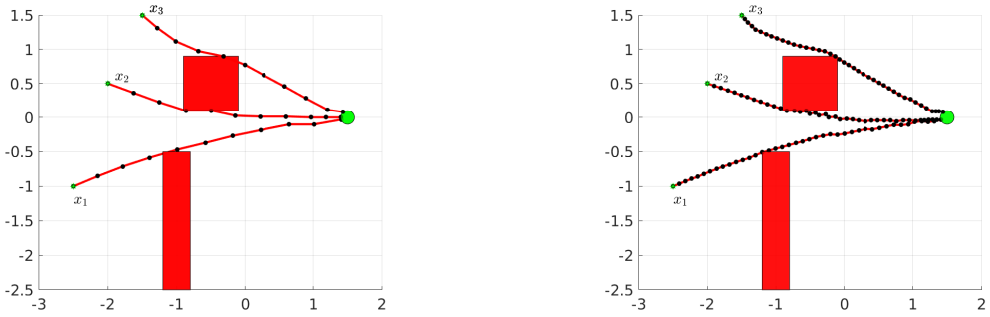

Figure 4: (Example 2): Trajectories obtained by dichotomy with SOP in order to reach the target (in green) and to avoid the obstacles (in red), from different initial positions, with $N = 10$ (left) and $N = 40$ (right).

Now, we fix $N = 10$ and we analyze the sensitivity of the dichotomy approach with respect to the tolerance ϵ and the total budget $I_{\max}$. For this, some simulations are performed for two different initial positions, with values of ϵ ranging from 10^{-1} to 10^{-6} and with $I_{\max} = 3200, 6400$ or 12800.

		$x = (-2.5, -1)$			$x = (-1.5, 1.5)$		
$I_{\max}$	ϵ	z^ϵ	$W(x, z^\epsilon)$	CPU(s)	z^ϵ	$W(x, z^\epsilon)$	CPU(s)
3200	10^{-2}	2.28515625	-1.47e-03	5.26	1.96289062	-1.35e-03	5.35
	10^{-3}	2.28210449	-9.20e-05	7.29	1.95983887	-1.10e-03	8.00
	10^{-4}	2.28202820	-8.20e-05	8.78	1.95968628	-1.00e-03	10.46
	10^{-5}	2.28202820	-8.20e-05	10.23	1.95966721	-1.01e-03	12.20
	10^{-6}	2.28198826	-7.59e-05	12.16	1.95966721	-1.01e-03	14.46
6400	10^{-2}	2.28515625	-1.55e-03	44.52	1.96289062	-2.38e-03	51.74
	10^{-3}	2.28210449	-1.45e-04	62.91	1.95983887	-1.23e-03	69.65
	10^{-4}	2.28195190	-7.92e-06	75.62	1.95968628	-1.24e-03	82.57
	10^{-5}	2.28195190	-7.92e-06	87.40	1.95966721	-1.23e-03	95.81
	10^{-6}	2.28194714	-1.16e-05	103.70	1.95966661	-1.23e-03	115.16
12800	10^{-2}	2.28515625	-2.04e-03	322.89	1.95312500	-3.18e-04	291.91
	10^{-3}	2.28210449	-1.46e-04	456.10	1.95312500	-3.18e-04	403.99
	10^{-4}	2.28195190	-5.36e-05	557.97	1.95312500	-3.18e-04	485.43
	10^{-5}	2.28193283	-3.49e-05	644.22	1.95310593	-3.12e-04	566.87
	10^{-6}	2.28193283	-3.49e-05	760.31	1.95310593	-3.12e-04	677.94

Table 5: (Example 2) Approximation of $V(x)$ by Algorithm 5 (dichotomy) combined with SOP, for $x = (-2.5, -1)$ and $x = (-1.5, 1.5)$ and with different values of ϵ and $I_{\max}$.

The numerical results are reported in Table 5 (with SOP) and Table 6 (with SOPMS). Columns 3-5 correspond to simulations for $x_1 = (-2.5, -1)$, and columns 6-8 correspond to $x_3 = (-1.5, 1.5)$. For each initial point and for a set of threshold parameters ϵ (see algorithm 5), we give the corresponding computed value z^ϵ (an approximation of $V(x)$), the computed approximation of the auxiliary value $W(x, z^\epsilon)$, and the CPU time.

		$x = (-2.5, -1)$			$x = (-1.5, 1.5)$		
$I_{\max}$	ϵ	z^ϵ	$W(x, z^\epsilon)$	CPU(s)	z^ϵ	$W(x, z^\epsilon)$	CPU(s)
3200	10^{-2}	2.28515625	-1.75e-03	1.27	1.96289062	-1.52e-03	1.26
	10^{-3}	2.28210449	-1.40e-04	2.13	1.95983887	-1.26e-03	3.12
	10^{-4}	2.28202820	-1.10e-04	2.48	1.95968628	-1.27e-03	4.68
	10^{-5}	2.28199959	-1.07e-04	3.29	1.95968628	-1.27e-03	5.93
	10^{-6}	2.28199244	-1.06e-04	4.95	1.95968628	-1.27e-03	8.04
6400	10^{-2}	2.28515625	-1.75e-03	2.39	1.96289062	-1.52e-03	1.87
	10^{-3}	2.28210449	-1.40e-04	3.67	1.95983887	-1.26e-03	6.69
	10^{-4}	2.28202820	-1.10e-04	4.16	1.95968628	-1.27e-03	12.38
	10^{-5}	2.28199959	-1.07e-04	5.53	1.95968628	-1.27e-03	15.06
	10^{-6}	2.28199244	-1.06e-04	8.93	1.95968628	-1.27e-03	20.06
12800	10^{-2}	2.28515625	-1.75e-03	2.50	1.96289062	-1.52e-03	2.56
	10^{-3}	2.28210449	-1.40e-04	4.29	1.95983887	-1.26e-03	13.66
	10^{-4}	2.28202820	-1.10e-04	4.92	1.95968628	-1.27e-03	27.73
	10^{-5}	2.28199959	-1.07e-04	7.35	1.95968628	-1.27e-03	35.36
	10^{-6}	2.28199244	-1.06e-04	15.17	1.95968628	-1.27e-03	48.39

Table 6: (Example 2) Approximation of $V(x)$ by Algorithm 5 (dichotomy) combined with SOPMS, for $x = (-2.5, -1)$ and $x = (-1.5, 1.5)$ and with different values of ϵ and $I_{\max}$.

From Tables 5 and 6 we first notice that auxiliary values $W(x, z^\epsilon)$ are all negative, which indicates that the computed optimal trajectories are admissible (i.e., the constraints of the optimal control problem are satisfied). Table 5 and 6 show also that, in general, the approximation z^ϵ of $V(x)$ decreases when ϵ decreases. Moreover, for a given tolerance ϵ, the value of z^ϵ decreases when the number of nodes $I_{\max}$ increases.

Besides Tables 5 and 6 show that the dichotomy algorithm performs faster when combined with SOPMS than when it is combined with SOP (roughly a gain from order 2 to 10, depending of the initial point and the threshold ϵ), up to a certain given precision.

5.3. Example 3: Optimal control of a heat equation

In this example, taken from [1], we want to illustrate the performances of our approach for solving a control problem of a partial differential equation. The discrete version of the problem leads to a control

problem in a high dimensional state space. Consider the following heat equation:

$$\begin{cases} \frac{\partial y}{\partial t}(s,x) = \sigma \frac{\partial^2 y}{\partial x^2}(s,x) + y_0(x)\alpha(s), \quad \text{for} \quad (s,x) \in [0,T]\times]0,1[, \\ y(s,0) = y(s,1) = 0, \quad \text{for} \quad s \in [0,T], \\ y(0,x) = y_0(x), \quad \text{for} \quad x \in [0,1], \end{cases} \tag{27}$$

where $\sigma = 0.1$, the control $\alpha(\cdot)$ takes values in $A := [-1,1]$, $T = 1$ and $y_0(x) = -x^2 + x$, for $x \in [0,1]$.

Our purpose is to minimize, by using the control law $\alpha(\cdot)$, the temperature $y^\alpha(t,x)$, solution of (27), for $t \in [0,T]$ and $x \in [0,1]$. For this reason, we consider the following cost functional of Bolza type, with $\gamma > 0$:

$$Q(y_0,\alpha) = \int_0^T \Big(\int_0^1 (y^\alpha(s,x))^2 dx + \gamma a^2(s) \Big) ds + \int_0^1 (y^\alpha(T,x))^2 dx. \tag{28}$$

An approximation of the state equation (27) can be performed by a classical implicit scheme. Consider a time grid with $N = 20$ time steps, $t_k = k\Delta t$ for $k = 0, ..., N$, where $\Delta t = \frac{T}{N}$ and a space grid with $d = 10^3$ points on $]0,1[$, $x_j = j\Delta x$ for $j = 0, ..., d+1$ and where the space step is given by $\Delta x = \frac{1}{d+1}$. Hence, the implicit scheme approximating (27) is given by:

$$\begin{cases} \frac{y_{k+1}^i - y_k^j}{\Delta t} = \sigma \frac{y_{k+1}^{j+1} - 2y_{k+1}^j + y_{k+1}^{j-1}}{\Delta x^2} + y_0(x_j)a_k, \quad \text{for } 0 \le k \le N-1, \quad 1 \le j \le d, \\ y_k^0 = y_k^{d+1} = 0 \end{cases} \tag{29}$$

where y_k^j is an approximation of $y(t_k, x_j)$, and $a = (a_k)_{1 \le k \le N}$ is a piece-wise constant control law. Equation (29) can be rewritten as follows:

$$Y_{k+1} = F(Y_k, a_k) \tag{30}$$

with $Y_k := (y(t_k, x_j))_{1 \le j \le d} \in \mathbb{R}^d$. Furthermore, the cost function Q can be approximated by:

$$\mathcal{J}(Y_0, a) = \sum_{k=0}^{N-1} \rho(Y_k, a_k) + \Phi(Y_N),$$

where the instantaneous cost ρ and the terminal cost Φ are defined, for $Y \in \mathbb{R}^d$ and $a \in A$, by:

$$\rho(Y,a) = \frac{\Delta t}{2}\Big(\|Y\|^2 + \|F(Y,a)\|^2 + 2\gamma a^2 \Big) \quad \text{and} \quad \Phi(Y) = \|Y\|^2.$$

The uncontrolled solution, presented in the left of figure 5, corresponds to a numerical solution of (27)

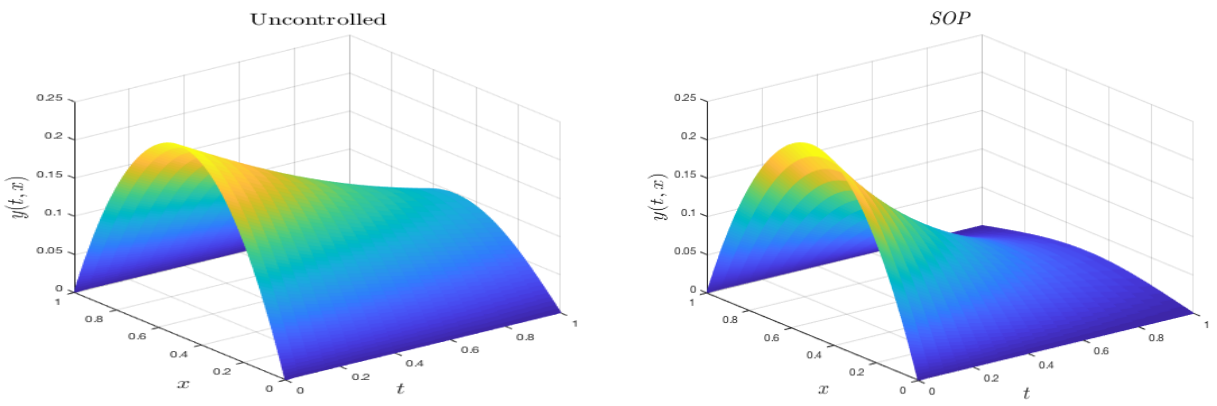

Figure 5: (Example 3) Uncontrolled solution (left) and optimal solution with SOP (right), for $I_{\max} = 10^4$, $\gamma = 0.01$ and $d = 10^3$.

while taking the control law $a(\cdot) \equiv 0$. The approximation of the optimal solution obtained by SOP is displayed in Figure 5 (right). Furthermore, the solutions obtained by SOP and SOPMS are better than the one computed with OP. This observation can be confirmed by a comparison between the solutions norms and the cost values computed with different planning algorithms as presented in figure 6. Indeed in the figure 6(middle), we can see that the optimal cost computed by SOP is identical to the one computed by SOPMS, and is lower to the optimal value computed by OP.

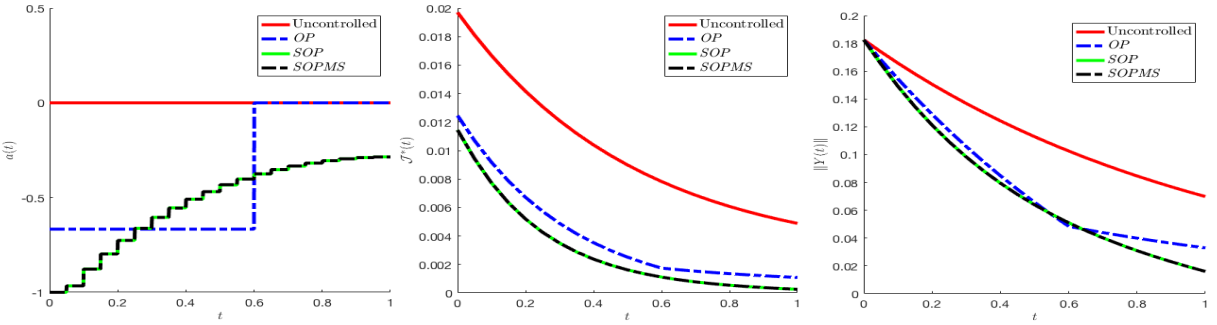

Figure 6: (Example 3) Controls computed by OP, SOP and SOPMS algorithms (left), time comparison of cost functions (middle), norms of the solutions (right), for $I_{\max} = 10^4$, $\gamma = 0.01$ and $d = 10^3$.

Finally, we consider a state constraint: $y^a(t,x) \geq \frac{1}{3}y_0(x)$ for every $t \in [0,T]$ and $x \in [0,1]$. The corresponding constrained solution, obtained by using the dichotomy algorithm combined with SOP, is represented in figure 7.

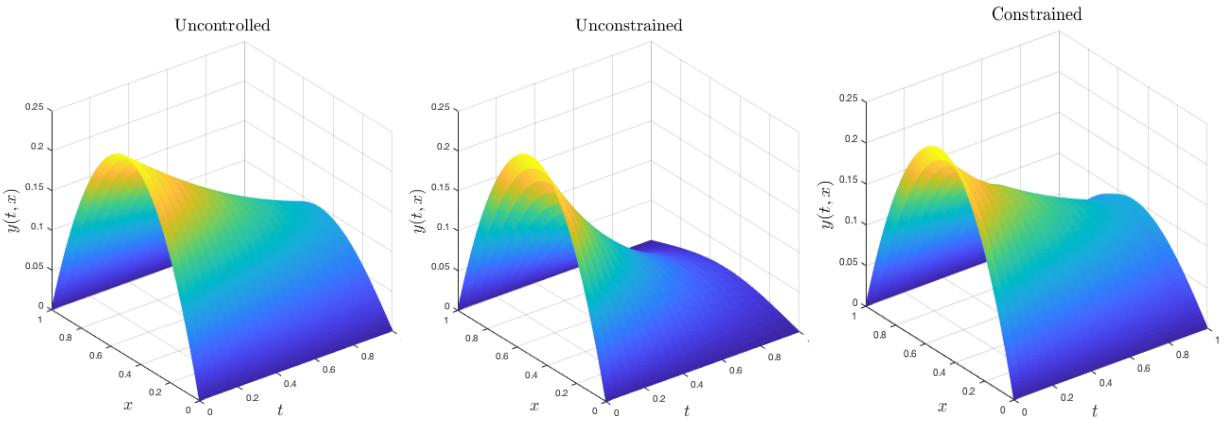

Figure 7: (Example 3) Uncontrolled solution (left), unconstrained and controlled solution obtained by SOP (middle) and constrained and controlled solution obtained by dichotomy combined with SOP for $d = 10^3$, $\gamma = 0.01$ and $I_{\max} = 10^4$.

5.4. Example 4: Windshear problem

We consider the abort landing problem of an aircraft in presence of wind disturbances [16, 6, 7]. We follow the formulation of the problem as in [4] (where an Hamilton-Jacobi approach is used). Here, we use an optimistic planning algorithm to compute the optimal solution. The flight of the aircraft is modeled in a vertical plane over a flat earth, assuming that all forces act on the center of gravity of the aircraft and lie in the same plane of symmetry. The aircraft's motion is described by (see [16, 7, 4] for more details):

$$\begin{cases} \dot{x}(s) &= u(s)\cos(\gamma(s)) + \omega_x(x(s)) \\ \dot{h}(s) &= u(s)\sin(\gamma(s)) + \omega_h(x(s), h(s)) \\ \dot{u}(s) &= \frac{F_T(u(s))}{m}\cos(\theta(s)+\delta) - \frac{F_D(u(s),\theta(s))}{m} - g\sin(\gamma(s)) - \dot{\omega}_x(x(s))\cos(\gamma(s)) - \dot{\omega}_h(x(s),h(s))\sin(\gamma(s)) \\ \dot{\gamma}(s) &= \frac{1}{u(s)}\left(\frac{\beta F_T(u(s))}{m}\sin(\theta(s)+\delta) + \frac{F_L(u(s),\theta(s))}{m} - g\cos(\gamma(s)) + \dot{\omega}_x(x(s))\sin(\gamma(s)) - \dot{\omega}_h(x(s),h(s))\cos(\gamma(s))\right) \\ \dot{\theta}(s) &= \alpha(s), \end{cases} \tag{31}$$

where x is the horizontal distance, h denotes the altitude, u is the aircraft velocity, γ is the relative path inclination, θ is the angle of attack, $\delta > 0$ is a parameter of the model, α the control variable is the angular velocity. Moreover, ω_x and ω_h are respectively supposed to be known horizontal and vertical components of the wind velocity vector, $\dot{\omega}_x$ and $\dot{\omega}_h$ are their derivatives, F_T, F_L and F_D denote respectively the thrust, lift and drag forces whose expressions can be found in [4, 6, 7]. We represent the state variables by a vector

$y \in \mathbb{R}^5$ given by $y := (x, h, u, \gamma, \theta)^\top$, hence the dynamical system (31) can be written as $\dot{y}(s) = f(y(s), \alpha(s))$. The sets of control, A, and of state constraints, $\mathcal{K}$, are of the form:

$$A := [a_{\min}, a_{\max}] \quad \text{and} \quad \mathcal{K} := \mathbb{R}^4 \times [\theta_{\min}, \theta_{\max}],$$

with $a_{\min} = -a_{\max} = -3$ deg s^{-1}, $\theta_{\min} = -180$ deg and $\theta_{\max} = 17.2$ deg.

The aim of the control problem is to maximize the minimal altitude that can be reached during an interval of time $[0, T]$, where $T := 40$. The maximum running cost function Φ is defined as $\Phi(y) := h_* - h$, where h is the aircraft altitude and $h_* := 1000$ is a reference altitude (all the parameters of this example are similar to those in [4]). In order to solve this problem, we discretize uniformly $[0, T]$ with N sub-intervals and consider the discrete time control problem:

$$V(y_0) = \inf_{a=(a_k)_k \in A^N} \Big\{ \max_{k=1,\cdots,N} \Phi(y_k^a) \quad | \quad y_k^a \in \mathcal{K}, \quad \text{for all} \quad k = 1, \cdots, N \Big\}.$$

where $(y_k^a)_k$ is the discrete trajectory corresponding to the control sequence $a = (a_k)_k \in A^N$, starting from the initial state y_0. The corresponding auxiliary problem is of the form:

$$W(y_0, z) = \inf_{(a_k) \in A^N} \max_{0 \le k \le N} \Big\{ \Big(\Phi(y_k^a) - z \Big) \bigvee g(y_k^a) \Big\},$$

with the function g, is defined, for $y = (x, h, u, \gamma, \theta) \in \mathbb{R}^5$, by: $g(y) = \max(\theta_{\min} - \theta, \theta - \theta_{\max})$. The numerical simulations for this example are performed by Algorithm 5 combined with SOP. The parameter $p_{\max}$ in SOPis chosen as $p_{\max}(n) = \sqrt{n}$. The dichotomy approach is initialized with $Z_{\min} = 0$ and $Z_{\max} = 1000$, and the tolerance parameter is $\epsilon = 10^{-5}$.

We fix $N = 40$. Figure 8 shows the numerical results obtained for two different initial configurations

$$y_0 = (0, 600, 239.7, -2.249 \text{ deg}, 7.373 \text{ deg}) \quad \text{and} \quad y_1 = (0, 650, 239.7, -3.400 \text{ deg}, 7.373 \text{ deg}).$$

Table 7 presents the numerical results obtained with different values of the budget allowed in SOPapproach. In particular, column 3 presents the values z^ϵ obtained by dichotomy as approximations of $V(y_0)$ or $V(y_1)$. Column 4 gives the minimal altitude of the aircraft. Table 7 confirms that the performances of the computed trajectories are improved when $I_{\max}$ increases.

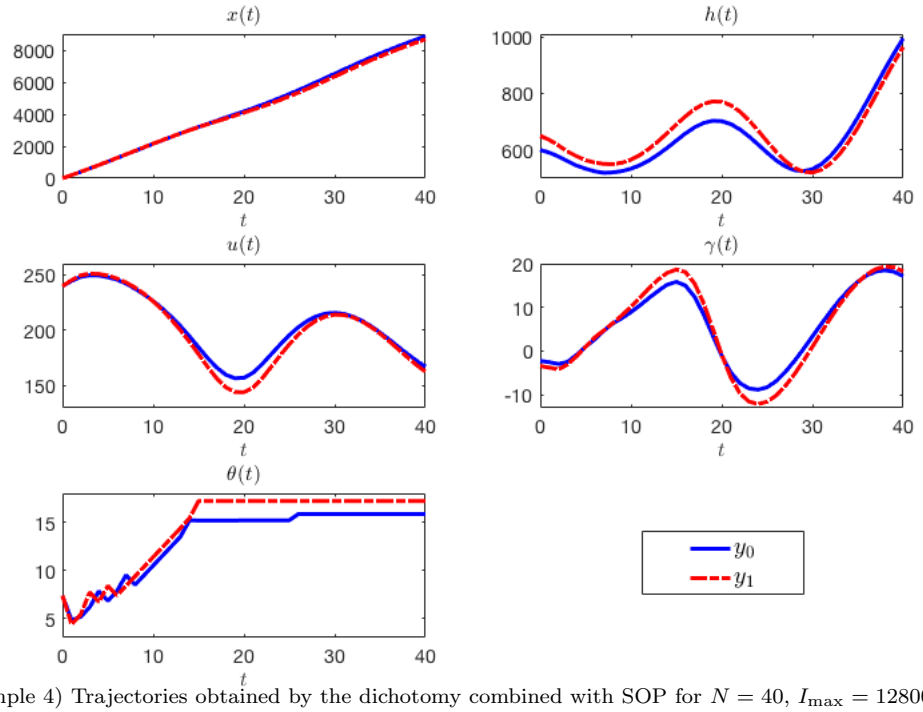

Figure 8: (Example 4) Trajectories obtained by the dichotomy combined with SOP for $N = 40$, $I_{\max} = 12800$ and from initial configurations y_0 and y_1.

Initial configuration	$I_{\max}$	z^{ϵ} (ft)	$h^{*}_{\min}$ (ft)	CPU (s)
y_0	3200	555.496	404.504	8.21
	6400	499.893	500.107	22.65
	12800	480.721	519.279	75.91
	25600	480.721	519.279	342.13
y_1	3200	524.147	475.853	11.88
	6400	480.957	519.043	16.34
	12800	480.919	519.081	74.43
	25600	479.675	520.325	259.55

Table 7: (Example 4) Performance of the dichotomy combined with SOP for different values of N and $I_{\max}$ and from two different initial configurations.

6. Appendix A: Proofs of convergence results for OP and SOP algorithms

6.1. Proof of Theorem 4.3.

Consider a given node A_i of the tree Υ at some depth p_i. By Corollary 3.4, using the diameters $d_{i,k} = DM^{-s(k)}$, $\gamma := \frac{1}{L_{F,x}} < 1$, and estimate (4.1), we obtain the bound

$$\sigma_i \leq \frac{CD}{2}\Delta t \sum_{k=0}^{N-1} \gamma^k M^{-s(k)}$$

for some constant $C \geq 0$ independent of N (the same constant C as in Corollary 3.4). By using similar arguments in [9] (see also [10]), the following results hold:

- the split function $s(\cdot)$, indicating the number of splits at cell k, is decreasing, and $s(\cdot)$ decreases of at most 1 ($s(k) - 1 \leq s(k+1) \leq s(k)$, $\forall k$)
- denoting τ_0, τ_1,...,τ_n the lengths of the ranges where s is constant (here considering that $\tau_n = 0$ for p large enough), it holds $\tau_0 \leq \tau$, and $\tau_j \in \{\tau - 1, \tau\}$, $\forall j \geq 1$.

From this point, our analysis departs slightly from [9]. The previous results enable us to bound $s(k)$ as follows:

$$s(k) \geq \underline{s}(k) := r - 1 - \frac{(k - (N-1))}{\tau} \tag{32}$$

and

$$s(k) \leq \overline{s}(k) := r + 1 - \frac{(k - (N-1))}{\tau - 1}. \tag{33}$$

where $r = s(N-1)$ (see Fig. 9).

Hence

$$\begin{aligned} \sum_{k=0}^{N-1} \gamma^k M^{-s(k)} &\leq \sum_{k=0}^{N-1} \gamma^k M^{-(r-1+\frac{N-1}{\tau})+\frac{k}{\tau}} \\ &\leq M^{-(r-1+\frac{N-1}{\tau})} \frac{1}{1 - \gamma M^{\frac{1}{\tau}}} \end{aligned} \tag{34}$$

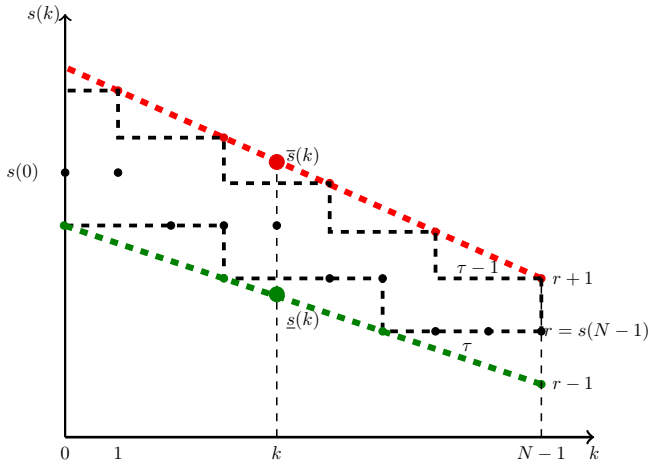

Figure 9: Example of split function $s(\cdot)$ (black dots) and the upper and lower bounds $\underline{s}(\cdot)$ and $\overline{s}(\cdot)$.

It remains to find a lower bound for r when p is large. By using (33), we have

$$p = \sum_{k=0}^{N-1} s(k) \leq \sum_{k=0}^{N-1} \overline{s}(k) \leq N(r+1) + \frac{N(N-1)}{2(\tau-1)}.$$

Therefore, we obtain that $r \geq \frac{p}{N} - 1 - \frac{N-1}{2(\tau-1)}$, and with the bound (34), we conclude that:

$$\frac{C}{2}\Delta t \sum_{k=0}^{N-1} \gamma^k M^{-s(k)} \leq \frac{C}{2}\Delta t \frac{1}{1-\gamma M^{\frac{1}{\tau}}} M^{q_N} M^{-\frac{p}{N}}$$

where $q_N = 2 + \frac{N-1}{2(\tau-1)} - \frac{N-1}{\tau} = 2 - (N-1)\frac{\tau-2}{2\tau(\tau-1)}$. The proof is then completed. □

6.2. Proof of Theorem 4.7.

Let $n \leq I_{max}$ be the number of iterations and let $p = p(n)$ be the corresponding depth. Suppose $m > 1$, where m is the asympototic branching factor. We have

$$n = |\Upsilon^*| = |\bigcup_{q=0}^{p} \Upsilon_q^*| = \sum_{q=0}^{p} |\Upsilon_q^*| \leq \sum_{q=0}^{p} Rm^q \ \leq \ R\frac{m^{p+1}-1}{m-1} \ \leq \ R\frac{m^{p+1}}{m-1}$$

(recall from the definitions that Υ_q^* contains at most Rm^q cells), therefore

$$p + 1 \geq \frac{\log(\frac{(m-1)n}{R})}{\log(m)}.$$

By using the estimate of Theorem 4.3, we deduce that for some cell i,

$$\delta_i \leq c_1(N)\, M^{-\frac{p}{N}} \leq c_1(N)\, M^{\frac{1}{N} - \frac{\log(\frac{(m-1)n}{R})}{N\log(m)}} = c_2(N)\, n^{-\frac{\log(M)}{N\log(m)}}.$$

where $c_2(N) := c_1(N) M^{\frac{1}{N} - \frac{\log(\frac{(m-1)}{R})}{N\log(m)}}$ (for $m > 1$), which leads to the desired estimate.

For the case $m = 1$, we have $p + 1 \geq \frac{n}{R}$ and $\delta_i \leq c_2(N) M^{-\frac{n}{NR}}$ with $c_2(N) := c_1(N) M^{\frac{1}{N}}$. □

6.3. Sketch of proof for Lemma 4.9.

By [18, Lemma 4.1], we know that if $0 \leq p \leq p_{\max}(n)$ and

$$n \geq p_{\max}(n) \sum_{q=0}^{p} |\Upsilon_q^*|, \tag{35}$$

then $p \leq \bar{p}_n$ where $\bar{p}_n$ is the depth of the deepest expanded node in the branch containing an optimal sequence. (Roughly speaking, the r.h.s of (35) is an upper bound of the number of nodes, in the worst case situation, needed to be expanded by the algorithm in order to go through all nodes of $\bigcup_{q=0}^{p} \Upsilon_q^*$.)

Then we reproduce the same arguments as in the first part of the proof of [18, Theorem 4.2]: since $p(n)$ is the smallest depth verifying (25) (and using the bound $|\Upsilon_q^*| \leq Rm^q$), the reverse inequality is verified by $p = p(n) - 1$, with $n \geq p_{\max}(n) \sum_{q=0}^{p(n)-1} Rm^q$. If $p(n) - 1 \leq p_{\max}(n)$, this implies that the SOP algorithm has expanded at least one optimal node at depth $p(n) - 1 \leq \bar{p}_n$. Otherwise if $p(n) - 1 > p_{\max}(n)$ and since SOP algorithm does not expand nodes beyond depth $p_{\max}(n)$, this implies that $\bar{p}_n = p_{\max}(n)$. We deduce in all cases that $p_n^* = \min(p(n) - 1, p_{\max}(n)) \leq \bar{p}_n$, and, therefore, an error bound $\delta_{\bar{p}_n} \leq \delta_{p_n^*}$. □

6.4. Proof of Theorem 4.10.

Let us first consider the case $m > 1$. Let $p_{\max}(n) := n^\eta$, for some $\eta \in [0, 1[$. Recall that $p(n)$ satisfies (25), therefore $p(n) - 1$ satisfies the reversed inequality, that is:

$$R\,\frac{m^{p(n)} - 1}{m - 1} = \sum_{q=0}^{p(n)-1} Rm^q \leq \frac{n}{p_{\max}(n)} = n^{1-\eta},$$

which implies that:

$$p(n) \leq \frac{\log(1 + \frac{(m-1)}{R}\, n^{1-\eta})}{\log(m)} \overset{n\to\infty}{=} \frac{\log(\frac{(m-1)}{R}\, n^{1-\eta})}{\log(m)} + o(1) \overset{n\to\infty}{\sim} \frac{(1-\eta)}{\log(m)} \log(n). \tag{36}$$

Since $p_{\max}(n) = n^\eta$, we deduce from the last inequality that for n large enough, $p(n) \leq p_{\max}(n)$. Hence, in the algorithm, $p_n^* := \min(p(n)-1, p_{\max}(n)) = p(n)-1$ (p_n^* is the depth of the deepest expanded optimal node). On the other hand, by the definition of $p(n)$, we have also

$$R\,\frac{m^{p(n)+1}-1}{m-1} = \sum_{q=0}^{p(n)} Rm^q \geq \frac{n}{p_{\max}(n)} = n^{1-\eta},$$

and therefore

$$\begin{aligned} p(n) &\geq \frac{\log(\frac{(m-1)}{R} n^{1-\eta})}{\log(m)} - 1 \\ &\geq -\frac{\log(R)}{\log(m)} - 1 + \frac{(1-\eta)}{\log(m)} \log(n). \end{aligned}$$

Since $J_{i^*} - W(x,z) \leq \delta_{p^*} = c_1(N)\,\Delta t\, M^{-\frac{p^*}{N}}$, using the lower bound found for $p^* = p_n^* = p(n)-1$, we obtain

$$\delta_{p_n^*} \leq c_3(N)\,\Delta t\, M^{-\frac{(1-\eta)}{\log(m)}\frac{\log(n)}{N}} = c_3(N)\,\Delta t\, n^{-\frac{(1-\eta)}{\log(m)}\frac{\log(M)}{N}}$$

where $c_3(N) := c_1(N) M^{\frac{\log(R)}{N\log(m)} + \frac{2}{N}}$ which concludes to the desired estimate. Since $R \geq 1$ and $c_1(N)$ is bounded independently of N, we deduce that $c_3(N)$ is also bounded independently of N.

In the case $m = 1$ and $p_{\max}(n) = n^\eta$, by the definition (25),

$$Rp(n) = \sum_{q=0}^{p(n)-1} Rm^q < \frac{n}{p_{\max}(n)} = n^{1-\eta}, \tag{37}$$

and also

$$R(p(n)+1) = \sum_{q=0}^{p(n)} Rm^q \geq \frac{n}{p_{\max}(n)} = n^{1-\eta}. \tag{38}$$

Choosing $\eta \in [\frac{1}{2}, 1[$, by (37), we deduce $p_n^* \leq p_{\max}(n)$, hence $p_n^* = p(n)-1$. Therefore by (38),

$$\delta_{p_n^*} \leq c_1(N)\,\Delta t\, M^{-\frac{p(n)-1}{N}} \leq c_1(N)\,\Delta t\, M^{-\frac{2}{N}} M^{-\frac{n^{1-\eta}}{R}}.$$

□

References

[1] A. Alla, M. Falcone, and L. Saluzzi. An efficient dp algorithm on a tree-structure for finite horizon optimal control problems. *SIAM Journal on Scientific Computing*, 41(4):A2384–A2406, 2019.

[2] A. Altarovici, O. Bokanowski, and H. Zidani. A general Hamilton-Jacobi framework for non-linear state-constrained control problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 19(2):337–357, 2013.

[3] R. Apkarian, D. Noll, and A. Rondepierre. Mixed H_2/H_∞ control via nonsmooth optimization. *SIAM J. Control and Optimization*, 47(3):1516–1546, 2008.

[4] M. Assellaou, O. Bokanowski, A. Desilles, and H. Zidani. Value function and optimal trajectories for a maximum running cost control problem with state constraints. application to an abort landing problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 52(1):305–335, 2018.

[5] J.-P. Aubin and A. Cellina. *Differential inclusions: set-valued maps and viability theory*, volume 264. Springer Science & Business Media, 2012.

[6] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, part 1: Necessary conditions. *Journal of Optimization Theory and Applications*, 70(1):1–23, 1991.

[7] R. Bulirsch, F. Montrone, and H. J. Pesch. Abort landing in the presence of windshear as a minimax optimal control problem, part 2: Multiple shooting and homotopy. *Journal of Optimization Theory and Applications*, 70(2):223–254, 1991.

[8] L. Busoniu and R. Munos. Optimistic planning for markov decision processes. In N. D. Lawrence and M. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 182–189, La Palma, Canary Islands, 2012. PMLR.

[9] L. Buşoniu, E. Páll, and R. Munos. Discounted near-optimal control of general continuous-action nonlinear systems using optimistic planning. In *2016 American Control Conference (ACC)*, pages 203–208. IEEE, 2016.

[10] L. Buşoniu, E. Páll, and R. Munos. Continuous-action planning for discounted infinite-horizon nonlinear optimal control with lipschitz values. *Automatica*, 92:100–108, 2018.

[11] R. Elliott and N. Kalton. *The existence of value in differential games*, volume 126. American Mathematical Soc., 1972.

[12] C. Hermosilla, R. Vinter, and H. Zidani. Hamilton-jacobi-bellman equations for optimal control processes with convex state constraints. *Systems and Control Letters*, 109:3036, 2017.

[13] J.-F. Hren and R. Munos. Optimistic planning of deterministic systems. In *European Workshop on Reinforcement Learning*, pages 151–164. Springer, 2008.

[14] C. Mansley, A. Weinstein, and M. Littman. Sample-based planning for continuous action markov decision processes. In *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

[15] K. Máthé, L. Buçoniu, R. Munos, and B. De Schutter. Optimistic planning with a limited number of action switches for near-optimal nonlinear control. In *53rd IEEE Conference on Decision and Control*, pages 3518–3523. IEEE, 2014.

[16] A. Miele, T. Wang, C. Tzeng, and W. Melvin. Optimal abort landing trajectories in the presence of windshear. *Journal of Optimization Theory and Applications*, 55(2):165–202, 1987.

[17] R. Mifflin. An algorithm for constrained optimization with semismooth functions. *Math. Oper. Res*, 2:191207, 1977.

[18] R. Munos. From bandits to Monte-Carlo tree search: the optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014.

[19] C. Sagastizábal and M. Solodov. An infeasible bundle method for nonsmooth convex constrained optimization without a penalty function or a filter. *SIAM J. Optim.*, 16(1):146–169, 2005.

[20] H. M. Soner. Optimal control with state-space constraint I. *SIAM Journal on Control and Optimization*, 24(3):552–561, 1986.