SYNTHESIS OF SAFE CONTROLLERS FOR NONLINEAR SYSTEMS USING DYNAMIC PROGRAMMING TECHNIQUES

Jorge Estrela da Silva
School of Engineering
Polytechnic of Porto
Portugal
jes@isep.ipp.pt

João Borges de Sousa
Faculty of Engineering
University of Porto
Portugal
jtasso@fe.up.pt

Fernando Lobo Pereira
Faculty of Engineering
University of Porto
Portugal
flp@fe.up.pt

Abstract

A dynamic programming based procedure for the synthesis of near-optimal controllers for sampled data systems is proposed. The main features of the proposed procedure are the approximation of nonlinear systems by piecewise affine systems, the synthesis of control strategy in the form of a piecewise constant map, and handling of state constraints and exogenous inputs. In order to increase accuracy, the dynamic programming equations are evaluated for an horizon consisting of several sampling periods instead of the typical single time step approach. The dynamic programming equations are decomposed into set-valued operations and approximations, such that the resulting closed loop system is ensured to be stable and safe, even though the control strategy is computed by numerical methods. Based on that decomposition, two different implementations, of varying complexity and accuracy, are compared. Results are illustrated with the solution of a pursuit-evasion problem.

Key words

Dynamic programming, nonlinear systems, numerical methods, safety, set invariance.

1 Introduction

This paper focuses on the problem of optimal feedback control synthesis. Exact solutions can be obtained analytically for some dynamic optimization problems, most notably the Linear Quadratic Regulator (LQR). However, there is a vast range of problems for which analytical solutions are hard of impossible to find. Even for linear systems, dynamic optimization problems become more complex as soon as input or state constraints are considered, hence the popularity of the Model Predictive Control (MPC) technique (Camacho and Bordons, 2004). The LQR, while of theoretical and practical interest, does not take into account either state or input constraints. Model Predictive control and its nonlinear variation, Nonlinear MPC (NMPC) (Grüne and Pannek, 2011), while applicable to a wider range of problems, in general involve the on-line solution of a quadratic (in the MPC case) or nonlinear (in the NMPC case) programming problem on each control instant. Although, given the constant technological progress, such computation is feasible for an increasing number applications, there still remain scenarios for which minimization of computational requirements is of prime importance.

Value function based methods, such as dynamic programming (DP), provide, in theory, a more efficient approach for feedback control: given the value function and the current system state, the optimal control can be computed as a simple static optimization over the domain of the control input and, eventually, disturbances (e.g., in the framework of differential games (Krasovskii and Subbotin, 1988)). However, for many dynamic optimization problems, it is impossible to obtain the value function in analytical form. Frequently, numerical methods are employed to obtain an approximate solution. Moreover, in certain problems, both the value function and the optimal feedback control law may be non-differentiable or even discontinuous, presenting an additional challenge for the derivation of effective approximation methods. Numerical approaches based on the iterative evaluation of the Bellman equation or some temporal discretization of the Hamilton-Jacobi-Bellman/Isaacs equations can be found in the literature (see, e.g., (Mitchell, 2007; Cristiani and Falcone, 2009; Cardaliaguet et al., 1999)). Those methods compute the solution for a discrete set of the state space, i.e., they are grid based methods. Approximations based on a combination of some set of basis functions – such as polynomials (Beard et al., 1997; Hu and Shu, 1999), radial basis functions (Junge and Schreiber, 2015; Cecil et al., 2004) and spectral methods (Dehghan and Salehi, 2010) – avoid the state space discretization, but are less suitable to describe discontinuous phenomena.

For simulation studies and scenario analysis, the type of approximation is usually not critical. On the other hand, for feedback control synthesis, it is essential that the value function approximation and associated control law ensure stability of the closed-loop system. Set-valued methods (Cardaliaguet et al., 1999) provide such type of solution. Besides the typical space and time discretization, these methods rely on this particular procedure: instead of evaluating the dynamic programming equation only at the centroid of each grid cell, each grid cell takes the worst case value for all states belonging to that cell.

This work is focused on sampled data systems (Ragazzini and Franklin, 1958; Franklin et al., 1997; Chen and Francis, 1995). A sampled data system is a system whose state evolves in continuous-time but is observed only at discrete instants, the sampling instants. Additionally, control is assumed to change only at the sampling instants.

The set-valued methods described in (Cardaliaguet et al., 1999) are tailored specifically for pure continuous-time systems, i.e., both the plant and the controller are continuous-time. Moreover, very few details are given concerning computational implementation. Related work can also be found in (Habets et al., 2006), in the context of continuous-time piecewise-affine systems. The authors employ dynamic programming and computational geometry techniques to approach the reach-avoid problem. However, only control inputs are considered, and the feedback control law is also continuous-time.

This paper proposes an algorithm specifically tailored for sampled data systems. A method for improving the accuracy of the solution, loosely inspired on the receding horizon techniques, is described. This method offers an efficient way, both computationally and in terms of storage requirements, of increasing solution accuracy without resorting to increasing grid resolutions. A concept of generalized value function, which allows for best effort control action even if the system state is out of guaranteed safe region (the maximal robustly controlled invariant set (Blanchini and Miani, 2008)), is also proposed. Finally, two different implementations of the numerical solver for arbitrary dimensions are compared: one based on over-approximation of reachable sets by general convex polytopes and other based on over-approximation of reachable sets by axis-aligned hyper-rectangles (AAHR).

The paper is organized as follows. Section 2 presents the problem targeted by the proposed algorithm, along with main assumptions and some background. Section 3 discusses the approximation of the original problem by one with a fully discrete-time system. Section 4 presents the set-valued formulation. Section 5 discusses proposed extensions to the standard DP method. Section 6 describes the main aspects of the implementation of the numerical solver. An illustrative problem is solved on section 7 and some conclusions are drawn on section 8.

2 Problem Formulation

2.1 System model

The model of the system to be controlled is of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the system state, $\boldsymbol{u}(t) \in U_u$ is the control input, $\boldsymbol{v}(t) \in U_v$ is a disturbance input and $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ is a Lipschitz continuous function in a given domain of interest $\mathcal{D} \subset \mathbb{R}^n$, i.e.,

$$\begin{aligned} &\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) - \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{v})\| \\ &\leq \lambda \|\boldsymbol{x} - \boldsymbol{y}\|, \, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{D}, \, \boldsymbol{u} \in U_u, \, \boldsymbol{v} \in U_v. \end{aligned} \tag{2}$$

The set U_u is discrete and finite, with cardinality n_u, as typically happens for computer controlled systems. The system dynamics are bounded as follows:

$$\begin{aligned} &|f_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})| \leq f_{i,\max} \\ &\forall \boldsymbol{x} \in \mathcal{D}, \boldsymbol{u} \in U_u, \boldsymbol{v} \in U_v \end{aligned} \tag{3}$$

where $f_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$, with $i \in \{1, \ldots, n\}$, is the ith component of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$. Define $\boldsymbol{f}_{\max} := (f_{1,\max}, \ldots, f_{n,\max})$.

As usual for this type of systems, the control value is assumed to be constant between sampling instant, in a process denoted zero-order hold (ZOH). In this sense, define $\mathcal{U}_{SD}(U, \Delta_t)$ as the set of all possible control sequences generated from a discrete-time controller subject to a ZOH conversion, sampling interval Δ_t, and amplitude constrained to the finite discrete set U:

$$\begin{aligned} \mathcal{U}_{SD}(U, \Delta_t) := \{\boldsymbol{u} : \mathbb{R}_{\geq 0} \to U : \boldsymbol{u}(k\Delta_t + s) = \\ \boldsymbol{u}(k\Delta_t), \, \forall (k, s) \in \mathbb{N}_0 \times [0, \Delta_t)\}. \end{aligned} \tag{4}$$

It is assumed that the computer based control system has a base clock with a period of Δ_t, with k defined as the number of clock cycles since the initial time $t = 0$. The input sequence $a(.)$ is the output of a zero-order-hold converter, with minimal hold time Δ_t (one clock cycle):

$$\boldsymbol{u}(.) \in \mathcal{U}_{u,\Delta} := \mathcal{U}_{SD}(U_u, \Delta_t) \tag{5}$$

The set of admissible input sequences for $b(t)$ is

$$\mathcal{U}_v := \{b : \mathbb{R}_{\geq 0} \to U_v, \text{measurable}\}. \tag{6}$$

Define $\boldsymbol{x}(s; \boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{v})$ as the system state s units of time after departing from state $\boldsymbol{x}_0$ with inputs $\boldsymbol{u}(.) \in \mathcal{U}_{u,CT}$ and $\boldsymbol{v}(.) \in \mathcal{U}_v$.

In what follows, the unit step function is denoted by $h(t)$. Additionally, the same symbol will be used for constants (e.g., $\boldsymbol{u} \in U_u$) and functions (e.g., $\boldsymbol{u} \in \mathcal{U}_{u,CT}$) whenever the distinction is clear from the indicated domain.

2.2 Dynamic programming for sampled data systems

Dynamic programming can be employed to tackle a wide range of dynamic optimization problems. In what follows, for simplicity, the formulation is limited to problems with boundary conditions. Define the cost functional

$$J(t_f, \boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{v}) = \int_0^{t_f} L(\boldsymbol{x}(\tau; \boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{v}), \boldsymbol{u}(\tau), \boldsymbol{v}(\tau))d\tau. \quad (7)$$

and consider the following dynamic optimization problem:

$$V(\boldsymbol{x}_0) := \inf_{\boldsymbol{f}_c:\mathbb{R}^n \to U_u} \sup_{\boldsymbol{v}\in\mathcal{U}_v} J(k_f\Delta_t, \boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{v}) \quad (8)$$

subject to:

$$\boldsymbol{u}(k\Delta_t) = \boldsymbol{f}_c(\boldsymbol{x}(k\Delta_t; \boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{v})) \quad (9)$$

$$\boldsymbol{x}(k_f\Delta_t; \boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{v}) \in \mathcal{T} \quad (10)$$

$$\boldsymbol{x}(t) \in \mathcal{K} \quad (11)$$

as also (1) and (5), where $\mathcal{T}$ is a given target set for the controlled system. Assume, for the time being, that the target is reachable and, therefore, the problem has a solution. Note that the final time is considered only at the sampling instants. This is suitable for application where the control system must be able to detect that the target was reached, in order, for instance, to enter a different mode of operation. Nevertheless, target reachability between sampling instants can also be considered, with a slightly more complex computational implementation.

This is a zero-sum two person deterministic differential game (Isaacs, 1965; Krasovskii and Subbotin, 1988), also denoted target problem (Cardaliaguet et al., 1999). The objective of Player A (the controller) is to reach the target while minimizing J, for any possible sequence of Player B's actions. Moreover, it is assumed that, at each instant, Player B knows the current control action of Player A. This corresponds to the upper value formulation of the differential game. In this setting, the dynamic programming principle (DPP) can be cast as

$$V(\boldsymbol{x}_0) = \inf_{\boldsymbol{u}\in U_u} \sup_{\boldsymbol{v}\in\mathcal{U}_v} \Big\{ V(\boldsymbol{x}(\Delta_t; \boldsymbol{x}_0, \boldsymbol{u}h, \boldsymbol{v})) + \int_0^{\Delta_t} L(\boldsymbol{x}(\tau; \boldsymbol{x}_0, \boldsymbol{u}h, \boldsymbol{v}), \boldsymbol{u}, \boldsymbol{v}(\tau))d\tau \Big\} \quad (12)$$

$$V(\boldsymbol{x}_0) = 0,\ \boldsymbol{x}_0 \in \mathcal{T} \quad (13)$$

$$V(\boldsymbol{x}_0) = \infty,\ \boldsymbol{x}_0 \notin \mathcal{K} \quad (14)$$

The sup search over the space of disturbance sequences in (12) is an infinite dimensional problem. Infinite dimensional problems, in general, are not amenable for numerical approaches. In order to allow an efficient numerical solution, a suitable approximation of the system model will be considered, as described in the next section. Moreover, in what follows, only the case of constant $L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ is considered.

3 Simulation of sampled data system with continuous-time disturbances by a fully discrete-time system

3.1 General approach

Consider a function $\boldsymbol{F}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ and a set $U_{v,\Delta}$ such that

$$\{\boldsymbol{x}(t; \boldsymbol{x}_0, \boldsymbol{u}h, \boldsymbol{v}) : \boldsymbol{v} \in \mathcal{U}_v\} \subseteq \{\boldsymbol{F}(t, \boldsymbol{x}_0, \boldsymbol{u}, \boldsymbol{v}_\Delta) : \boldsymbol{v}_\Delta \in U_{v,\Delta}\}, \quad \forall \boldsymbol{x}_0 \in \mathcal{D}, \boldsymbol{u} \in U_u. \quad (15)$$

Moreover, define $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) := \boldsymbol{F}(\Delta_t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$. This means that system

$$\boldsymbol{x}((k+1)\Delta_t) = \boldsymbol{F}(\boldsymbol{x}(k\Delta_t), \boldsymbol{u}(k\Delta_t), \boldsymbol{v}_\Delta(k\Delta_t)) \quad (16)$$

with $\boldsymbol{u}(.) \in \mathcal{U}_{u,\Delta}$ and $\boldsymbol{v}_\Delta(.) \in \mathcal{U}_{SD}(U_{v,\Delta}, \Delta_t)$, is able to replicate the sequence of states at the sampling instants of any trajectory of the original sampled data system, i.e., (16) simulates the original sampled data system (1) at the sampling instants. In order to achieve the exact solution, it would be necessary to have an equality in condition (15). In general, it is not trivial to ensure equivalency. Therefore, the method settles for an approximation of the system model, leading to a sub-optimal solution.

System model (16) is suitable for a pure discrete-time formulation, as employed in the numerical schemes described in (Cristiani and Falcone, 2009) and (Cardaliaguet et al., 1999). The key difference is that, in this work, Δ_t is a parameter with physical meaning, as opposed to a time step that one wants to make as small as possible in order to improve the solution accuracy.

Thus, in the context of sampled data systems with running cost $L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = 1$, the following dynamic optimization problem is considered:

$$\tilde{V}(\boldsymbol{x}_0) = \inf_{\boldsymbol{f}_c:\mathbb{R}^n \to U_u} \sup_{\substack{\boldsymbol{v}_\Delta \in \\ \mathcal{U}_{SD}(U_{v,\Delta}, \Delta_t)}} k_f\Delta_t \quad (17)$$

subject to:

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \quad (18)$$

$$\boldsymbol{x}((k+1)\Delta_t) = \boldsymbol{F}(\boldsymbol{x}(k\Delta_t), \boldsymbol{u}(k\Delta_t), \boldsymbol{v}_\Delta(k\Delta_t)) \quad (19)$$

$$\boldsymbol{u}(k\Delta_t) = \boldsymbol{f}_c(\boldsymbol{x}(k\Delta_t)) \quad (20)$$

$$\boldsymbol{x}(k_f\Delta_t) \in \mathcal{T} \quad (21)$$

In this setting, the DP equation is cast as

$$\tilde{V}(\boldsymbol{x}) = \Delta_t + \inf_{\boldsymbol{u}\in U_u} \sup_{\boldsymbol{v}_\Delta \in U_{v,\Delta}} \tilde{V}(\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}_\Delta)). \quad (22)$$

The corresponding optimal feedback control law is

$$\boldsymbol{f}_c^*(\boldsymbol{x}) \in \arg\min_{\boldsymbol{u}\in U_u} \sup_{\boldsymbol{v}_\Delta \in U_{v,\Delta}} \tilde{V}(\boldsymbol{F}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}_\Delta)). \quad (23)$$

In general, (23) will be a sub-optimal control law for the original system, because, due to the over-approximation in (15), the DP equations may have to take into account adversarial actions, leading the trajectory to states of higher value, that will never take place in the original system. Nevertheless, given that $\tilde{V}(x) \geq V(x)$, this approach ensures that the performance of the closed loop system composed of the obtained control policy $\boldsymbol{f}_c^*$ and the original system, will be at least as good as indicated by $\tilde{V}(x)$.

In the context of a solution by numerical methods, it is not possible to compute $V(\boldsymbol{x})$ for the infinite number of points of the domain. In some numerical approaches (e.g., (Cristiani and Falcone, 2009; Mitchell, 2007)), the dynamic programming equations are computed only for a finite set of states. In those cases, even for a exact simulation (bisimulation) of the system dynamics, the right hand side of (44) will be computed based on an approximation of $V(\boldsymbol{x})$ (e.g., interpolation). Therefore, the solution obtained by those approaches is not guaranteed to ensure the desired reachability and safety (in the case of problems with state constraints) properties. A more suitable approach will be discussed in section 4.

3.2 Problems with state constraints

Consider the state constraint $\boldsymbol{x}(t) \in \mathcal{K}$. In order to be considered safe, the control law must ensure that the system state is in $\mathcal{K}$ at all times. The formulation presented in the previous sections gives guarantees concerning the system state at the sampling instants. However, in the presence of state constraints, it is necessary to ensure that, between sampling instants, the system trajectory does not cross the forbidden regions $\mathbb{R}^n \setminus \mathcal{K}$.

One way to achieve this is by considering a safety margin along the boundary of $\mathcal{K}$. Since $\|\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v})\|$ is finite, it is possible to estimate the maximum displacement of the system trajectory over a given period of time. Define the set

$$\mathcal{K}_\Delta := \{\boldsymbol{x} \in \mathcal{K} : \|\boldsymbol{x}-\boldsymbol{y}\| \geq \delta_c, \forall \boldsymbol{y} \in \mathbb{R}^n \setminus \mathcal{K}\}. \quad (24)$$

with $\delta_c = \frac{\|\boldsymbol{f}_{\max}\|}{2\Delta_t}$. Therefore, the safety margin is $\mathcal{K} \setminus \mathcal{K}_\Delta$. Then, if the system stays at $\mathcal{K}_\Delta$ at every sampling instant, it is impossible for it to leave $\mathcal{K}$ between those instants.

In this context, let us define the maximal robustly controlled invariant subset of $\mathcal{K}_\Delta$ as

$$\mathcal{D}_{\text{safe}} := \{\boldsymbol{x} : \tilde{V}(\boldsymbol{x}) \text{ is finite}\} \quad (25)$$

Therefore, as desired for a conservative approach, $\mathcal{D}_{\text{safe}}$ is an under-approximation of the maximal robustly controlled invariant subset of $\mathcal{K}$. Similarly, the over-approximation of the set of states that, in the worst case conditions, will be attracted to the forbidden set independently of the choice of controls (*unsafe set*), is defined as:

$$\mathcal{D}_{\text{unsafe}} := \mathcal{K} \setminus \mathcal{D}_{\text{safe}} \quad (26)$$

4 Set-valued computation of the dynamic programming equation

Consider an arbitrary partition of $\mathcal{D}$, $\mathcal{P} := \{C_1, \ldots, C_d\}$, composed of d subsets, and define $\tilde{V}_s : \mathcal{P} \to \mathbb{R}$ as

$$\tilde{V}_s(C_i) = \Delta_t + \min_{\boldsymbol{u}\in U_u} \max_{\substack{\boldsymbol{v}_\Delta \in U_{v,\Delta} \\ \boldsymbol{x}\in C_i}} \tilde{V}(\boldsymbol{F}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}_\Delta)) \quad (27)$$

where

$$\tilde{V}(\boldsymbol{x}) := \tilde{V}_s(C_i) : \boldsymbol{x} \in C_i,\ i \in \{1,\ldots,d\} \quad (28)$$

In general, the obtained solution will be sub-optimal with respect to original problem, since the same value and optimal actuation is assigned to all states in each subset C_i. However, the problem now becomes finite dimensional. Therefore, the exact solution for this problem can be computed, ensuring that the corresponding controller, albeit sub-optimal, will be safe and will ensure target reachability.

This formulation has many similarities with the one proposed in (Cardaliaguet et al., 1999). However, it must be remarked that, that in (Cardaliaguet et al., 1999), whose formulation is targeted to purely continuous-time systems, Δ_t is simply an adjustable parameter whose value, along with the grid resolution, can be changed arbitrarily to improve the solution accuracy. In the current case Δ_t is a constant with physical meaning.

Moreover, (27) can be cast in a more constructive fashion:

$$\tilde{V}_s(C_i) = \min_{\boldsymbol{u}\in U_u} M(C_i,\boldsymbol{u}) \quad (29)$$

where

$$M(C,\boldsymbol{u}) = \Delta_t + \max\{\tilde{V}_s(C_j) : R(C,\boldsymbol{u}) \bigcap C_j \neq \emptyset,\ j \in \{1,\ldots,d\}\} \quad (30)$$

and $R(C,\boldsymbol{u})$ is a reachable set, namely the set of states the disturbance can impose on the system when it departs from C with constant control $\boldsymbol{u}$:

$$R(C,\boldsymbol{u}) = \{\boldsymbol{F}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}_\Delta) : \boldsymbol{x} \in C,\ \boldsymbol{v}_\Delta \in U_{v,\Delta}\} \quad (31)$$

5 Extensions to the standard dynamic programming method

5.1 Trajectory evaluation over multiple control periods

The proposed set based approach computation of $\tilde{V}$ implies approximation errors analogous to using a 0th order interpolation in a traditional grid based numerical solver. Moreover, the approximation error is specially penalizing near the forbidden and unsafe sets. In some cases (specially for low d), the computed unsafe set may artificially "absorb" all states, leading to an useless solution. Notice that D_{unsafe} is an over-approximation of the exact unsafe set, in the sense that if some part of C_i belongs to the exact unsafe set, then the whole C_i is marked as belonging to D_{unsafe}.

In order to cope with this challenge, in the proposed approach, the dynamic programming equation are evaluated for state transitions over more than one single control period. The objective is twofold: on one hand, to make the "truncation" error incurred by considering $\tilde{V}(\boldsymbol{x})$ constant for each $\boldsymbol{x} \in C_i$ insignificant with respect to the running cost accumulated over several control periods; on the other hand, to allow propagation of the trajectory emanating from C_i to terminate in a region not marked as unsafe. Note that, since D_{unsafe} is an over-approximation, a trajectory crossing D_{unsafe} (specially its boundary) can still end up in a safe region, independently of the adversary actions.

In order to avoid exponential complexity, only trajectories with constant control are considered. Thus, the dynamic programmig equation becomes:

$$\tilde{V}_s(C_i) = \min_{\boldsymbol{u} \in U_u} MR(C_i, \boldsymbol{u}, 1) \tag{32}$$

where

$$MR(C, \boldsymbol{u}, l) = \min\{MR(R(C, \boldsymbol{u}), \boldsymbol{u}, l+1), M(C, \boldsymbol{u})\},\ l < N \tag{33}$$

$$MR(C, \boldsymbol{u}, N) = M(C, \boldsymbol{u}) \tag{34}$$

and N is an adjustable parameter.

5.2 Enabling escape from unsafe regions

In some situations, due to the presence of state constraints, it is impossible to ensure that the target is reachable from certain states, the unsafe states, without violation of those state constraints. However, it must be recalled that the solution is computed for the worst case scenario. In practice, even if the system reaches one of those states, it might be possible for it to return to a safe state, should the adversary conditions be not so extreme. Usually, the value function is defined as infinity for the unsafe states, thus preventing the definition of a "best effort" control law. In order to allow the definition of such control law, the proposed algorithm implicitly solves two problems in parallel: the main problem, e.g., the minimization of the cost to reach the target, and, for those states found to be unsafe, maximization of the time to reach the forbidden states. The underlying idea is that anytime the system reaches an unsafe region of the state space, it should try to escape it as fast as possible. The latter problem is defined as follows:

$$\begin{aligned} V_c(\boldsymbol{x}_0) &= \sup_{\boldsymbol{f}_c:\mathbb{R}^n \to U_u} \inf_{\boldsymbol{v} \in \mathcal{U}_v} k_f \Delta_t + c\Delta_c \\ &= - \inf_{\boldsymbol{f}_c:\mathbb{R}^n \to U_u} \sup_{\boldsymbol{v} \in \mathcal{U}_v} -k_f \Delta_t - c\Delta_c \end{aligned} \tag{35}$$

subject to:

$$\boldsymbol{x}(k_f \Delta_t + c\Delta_c) \in \mathbb{R}^n \setminus \mathcal{K}_\Delta \tag{36}$$

as also (19), (20) and (40). This is a differential game similar to (17), with $L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = -1$. Therefore, the same computational techniques can be applied for both problems.

The synthesis of the optimal control law is based on the following generalized value function:

$$V_g(\boldsymbol{x}_0) = \begin{cases} \tilde{V}(\boldsymbol{x}_0), & \boldsymbol{x} \in \mathcal{D}_{\text{safe}} \\ -V_c(\boldsymbol{x}_0) + V_\infty, & \boldsymbol{x} \in \mathcal{D}_{\text{unsafe}} \end{cases} \tag{37}$$

where

$$V_\infty = \max_{\boldsymbol{x} \in \mathcal{D}_{\text{safe}}} \tilde{V}(\boldsymbol{x}) + \max_{\boldsymbol{x} \in \mathcal{D}_{\text{unsafe}}} V_c(\boldsymbol{x}_0) + \Delta_t. \tag{38}$$

5.3 Safety margin as a design parameter

For large sampling periods, δ_c, as defined in section 3.2, can become unacceptably large. In general, it is desirable to leave δ_c as a design parameter. In order to allow that, an artificial sampling period, Δ_c, is used in the proposed approach. This faster sampling rate is used only at the design stage, and also just for the purpose of preventing state constraints violations. The control value cannot be changed at these intermediate steps. Thus, for design purposes, the considered state constraints become

$$\boldsymbol{x}(k\Delta_t + c\Delta_c) \in \mathcal{K}_\Delta \tag{39}$$

with $k \in \{0, \ldots, k_f\}$,

$$c \in \left\{0, \ldots, ceil\left(\frac{\Delta_t}{\Delta_c}\right) - 1\right\} \tag{40}$$

and

$$\Delta_c \leq 2\|\boldsymbol{f}_{\max}\|^{-1}\delta_c. \tag{41}$$

In the latter approach, between control instants, the system dynamics are modelled by

$$\boldsymbol{x}(k\Delta_t + (c+1)\Delta_c) = \boldsymbol{F}(\Delta_c, \boldsymbol{x}(k\Delta_t + c\Delta_c), \boldsymbol{u}(k\Delta_t), \boldsymbol{v}_\Delta(k\Delta_t + c\Delta_c)) \quad (42)$$

At the control instants, the system dynamics are modelled by

$$\boldsymbol{x}((k+1)\Delta_t) = \boldsymbol{F}(\Delta_c, \boldsymbol{x}((k+1)\Delta_t - \Delta_c), \boldsymbol{u}(k\Delta_t), \boldsymbol{v}_\Delta((k+1)\Delta_t - \Delta_c)) \quad (43)$$

The DP equation becomes

$$\tilde{V}(\boldsymbol{x}_0) = \Delta_t + \inf_{\boldsymbol{u}\in U_u} \sup_{\boldsymbol{v}_\Delta \in \mathcal{U}_{SD}(U_{v,\Delta},\Delta_c)} \tilde{V}(\boldsymbol{x}(\Delta_t)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0. \quad (44)$$

with $U_{v,\Delta}$ defined for Δ_c according to (15).

6 Numerical solver

Consider a partition of $\mathcal{D}$ composed of hyper-rectangles of equal size, with edges aligned with the principal axes, where Δ_{xi}, $i \in \{1, \ldots, n\}$, is the length of each hyper-rectangle along the i-th dimension. This partition defines a grid over the computational domain, where each hyper-rectangle of the above mentioned partition is a grid cell. More formally, a grid cell is defined as follows:

$$C_i := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{lb}_i \leq \boldsymbol{x} < \boldsymbol{ub}_i\} \quad (45)$$
$$\boldsymbol{lb}_i = \boldsymbol{c}_i - \left(\frac{\Delta_{x1}}{2}, \ldots, \frac{\Delta_{xn}}{2}\right)$$
$$\boldsymbol{ub}_i = \boldsymbol{c}_i + \left(\frac{\Delta_{x1}}{2}, \ldots, \frac{\Delta_{xn}}{2}\right)$$

$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}_\Delta)$ is assumed to be of the form $B(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{v}_\Delta$, which corresponds to the Euler method, or a linearization of the form $A(\boldsymbol{x}, \boldsymbol{u})\boldsymbol{x} + B(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{v}_\Delta$, if the derivatives of $\boldsymbol{f}$ are Lipschitz continuous.

The main operations to be implemented are the computation of the reachable set $R(C, \boldsymbol{u})$ and verification if two sets are disjoint. Given the assumed structure of $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}_\Delta)$, the reachable set can be defined as follows:

$$R(C, \boldsymbol{u}) = U_{v,\Delta} \oplus \{A(\boldsymbol{x}, \boldsymbol{u})\boldsymbol{x} + B(\boldsymbol{x}, \boldsymbol{u}) : \boldsymbol{x} \in C\} \quad (46)$$

where $S_1 \oplus S_2 = \{a + b : a \in S_1, b \in S_2\}$ is the Minkowski sum of sets S_1 and S_2, $\{A(\boldsymbol{x}, \boldsymbol{u})\boldsymbol{x}, \boldsymbol{x} \in C\}$ implies a rotation and scaling of set C, and term $B(\boldsymbol{x}, \boldsymbol{u})$ implies a translation of $\{A(\boldsymbol{x}, \boldsymbol{u})\boldsymbol{x}, \boldsymbol{x} \in C\}$ by $B(\boldsymbol{x}, \boldsymbol{u})$.

Minkowski sum is a non-trivial operation. The current implementation assumes that $U_{v,\Delta}$ is an AAHR or a truncated hyper-rectangle (e.g., an octagon, in the two-dimensional case).

The reach set $R(C, \boldsymbol{u})$, where C is a convex polytope, is a convex polytope itself. Moreover, define $RH(C, \boldsymbol{u})$ as the smallest AAHR containing $R(C, \boldsymbol{u})$. The next phase consists of determining which cells intersect with $R(C, \boldsymbol{u})$ (see (30)). In order to avoid verification of all d cells, the algorithm uses $RH(C, \boldsymbol{u})$ to make a more efficient pre-selection of the cells that can possibly be intersected by $R(C, \boldsymbol{u})$.

In order to check if $R(C, \boldsymbol{u})$ and a given C_i are disjoint, the algorithm takes into account that C_i is an AAHR. Still, the verification method implies substitution of the coordinates of the 2^n vertices of C_i on each of the linear inequalities describing $R(C, \boldsymbol{u})$.

A more efficient, albeit less accurate, algorithm is obtained by replacing $R(C, \boldsymbol{u})$ with $RH(C, \boldsymbol{u})$ in (30) (33). This is due to two factors. First, intersection between AAHR can be implemented with computational complexity $O(n)$. Second, the Minkowsky sum between AAHR is also simpler than between an AAHR and a general convex polytope.

On both cases, each cell is classified either as *target*, *forbidden*, *safe*, *unsafe* or *undefined*. The classification as *undefined* is used only at design stage, to indicate that the cell's value was still not computed.

In order to achieve better efficiency in the verification of state constraints, the forbidden region can be defined as a set of larger AAHR, instead of multiple cells.

The DP equations are evaluated for each grid cell until no change is detected between iterations. As soon as a cell is assigned a label different from *undefined*, the algorithm ceases to compute the corresponding DP equation. Therefore, algorithm termination is always ensured.

7 Numerical example

Consider the problem of designing a collision avoidance controller for a vehicle constrained to planar motion. The objective of the controller is to avoid that any obstacle reaches a given neighbourhood of the vehicle, the forbidden zone. The vehicle dynamics can be modelled by the unicycle model (Dubins vehicle (Dubins, 1957)): the vehicle has a maximum angular velocity ω_{max} and it is travelling at its maximum speed $V_{\max}$. The obstacles can be either static or moving, with a maximum speed $V_{\max}$ in the latter case. No curvature constraints are assumed for the obstacles.

Additionally, given the position of an obstacle with respect to the vehicle, the controller must be able to tell whether the system is in a safe or unsafe state.

The system dynamics are modelled as follows:

$$\begin{pmatrix} \dot{x_r}(t) \\ \dot{y_r}(t) \end{pmatrix} = \begin{pmatrix} 0 & \omega(t) \\ -\omega(t) & 0 \end{pmatrix} \begin{pmatrix} x_r(t) \\ y_r(t) \end{pmatrix} + \begin{pmatrix} -V_{\max} \\ 0 \end{pmatrix} + \boldsymbol{v}(t) \quad (47)$$

with $\boldsymbol{v}(t) \in \{\boldsymbol{v} \in \mathbb{R}^2 : \|\boldsymbol{v}\| \leq V_{\max}\}$, where $(x_r(t), y_r(t))$ is the obstacle position with respect to the vehicle's body fixed frame, $\omega(t)$ is the vehicle angular velocity, and $\boldsymbol{v}(t)$ is the obstacle input, whose value may change instantaneously in time. On the other hand, the vehicle control input, $\omega(t)$, being the output of a ZOH digital controller, is constant during each sampling period. Therefore, during each sampling period, the system can be described by an affine time invariant model.

The forbidden set consists of a 20 m radius circle centred at the controlled vehicle. The numerical values for the model parameters are: $V_{\max} = 25$ m/s, $U_u \in \{-0.18, 0, 0.18\}$ (rad/s) and $\Delta_t = 1$ s.

The value function was computed for different partitions of the state space, with $\mathcal{D} = [600, -100] \times [-300, 300]$, and for different horizons N. The results for the case of reachable sets consisting of general convex polytopes are presented in Figure 1 and Table 1, while the results for the case of reachable sets over-approximated by AAHR are presented in Figure 2 and Table 2. Both pictures were obtained from the solutions for $d = 601 \times 401$ and $N = 2$. The union of regions C and D, represented in figures 1 and 2, corresponds to $\mathcal{D}_{\text{safe}}$, while region B corresponds to $\mathcal{D}_{\text{unsafe}}$. Region C, although safe, requires that the vehicle does not use controls that can drive it to the unsafe zone. For obstacles in region D, any control is safe.

As expected, for the same number of cells and N, the computation by the first method takes more time than by the second. However, the second method seems to scale more poorly with increasing values of N. This can be explained by the fact that, due to the over-approximation, more intersections must be verified than in the first method. It must be noted that the reach set tends to grow on each recursion of (33). The advantage of the multi-period DP is also clear in certain cases: for instance, the solutions for $(d, N) = (301 \times 201, 3)$ and $(d, N) = (601 \times 401, 2)$ have similar accuracy, but the former is computed almost 16 times faster while requiring one quarter of the memory to store the solution.

Finally, it can be seen that the rate of improvement of the solution accuracy for the second method becomes very low as the number of cells increases (i.e., as the size of the cells decreases).

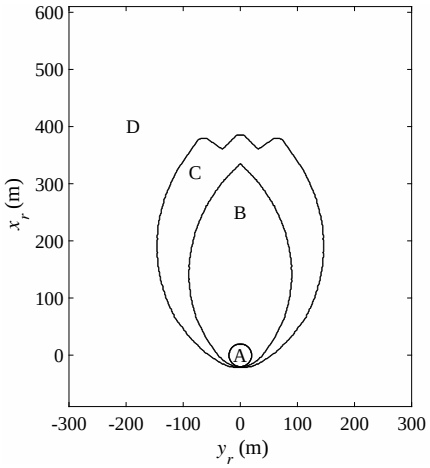

Figure 1. Hazard level as a function of the obstacle position, as computed with $R(C, \boldsymbol{u})$ based on general convex polytopes: region A - Forbidden zone; region B - Unsafe zone, obstacle avoidance depends on obstacle trajectory; region C - Alert zone, vehicle must follow evasive action to avoid entering unsafe zone; region D - vehicle may use any control action.

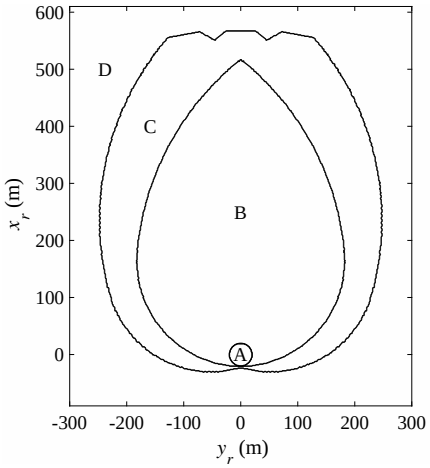

Figure 2. Hazard level as a function of the obstacle position, as computed with $R(C, \boldsymbol{u})$ over-approximated by axis-aligned hyper-rectangles (see Figure 1 for label description).

Table 1. Numerical solution with reach set based on general convex polytopes (a - number of unsafe cells and percentage relative to total number of cells)

Number of cells	Horizon (control cycles)	Processing time (s)	Unsafe Cells (a)
301×201	1	380	9079 (15.0%)
301×201	2	330	7044 (11.6%)
301×201	3	350	6672 (11.0%)
601×401	1	5482	30046 (12.5%)
601×401	2	5500	26383 (10.9%)

8 Conclusion

This paper extends existing results on set-valued methodologies for the solution dynamic optimization problems to the case of sampled data systems subjected to disturbances. The solution is obtained by numerical methods that, by design, ensure safety of the closed loop system at the expenses of optimality. In

Table 2. Numerical solution with reach set over-approximation by axes aligned hyper-rectangles (a - number of unsafe cells and percentage relative to total number of cells)

Number of cells	Horizon (control cycles)	Processing time (s)	Unsafe Cells (a)
151×101	1	1	5505 (36.1%)
151×101	2	1	4686 (30.7%)
151×101	3	1	4370 (28.7%)
151×101	4	1	4241 (27.8%)
151×101	5	16	4211 (27.6%)
151×101	6	223	4113 (27.0%)
301×201	1	14	17929 (29.6%)
301×201	2	16	16611 (27.5%)
301×201	3	18	16073 (26.6%)
301×201	4	46	15955 (26.4%)
601×401	1	200	65349 (27.1%)
601×401	2	1618	61239 (25.4%)
901×601	1	1052	142617 (26.3%)
1201×801	1	3073	248863 (25.9%)

what concerns the computation of the maximal robustly controlled invariant set, the numerical results illustrate that the proposed multi-period approach, with suitable choice of the horizon, can be a more efficient way of increasing accuracy, both in computation time and storage space, than just increasing the number of subsets in the state space partition $\{C_1, \ldots, C_d\}$.

One of the main building blocks of the numerical solvers is the intersection of reachable sets with the subsets C_i. Although introducing a larger error, over-approximation of the reachable sets by axis-aligned hyper-rectangles is more computationally efficient than the one based on general convex polytopes, since, in the former case, intersection cost is a function of the system dimension n, as opposed to latter case, where intersection cost is of order O(2^n). Nevertheless, in spite of the coarser approximation of the optimal solution, the faster method can still be preferable to choosing and tailoring a control function based on physical intuition and Lyapunov stability methods.

References

Beard, R. W., Saridis, G. N., and Wen, J. T. (1997). Galerkin approximations of the generalized hamilton-jacobi-bellman equation. *Automatica*, 33(12):2159 – 2177.

Blanchini, F. (1999). Set invariance in control. *Automatica*, 35(11):1747 – 1767.

Blanchini, F. and Miani, S. (2008). *Set-Theoretic Methods in Control*. Birkhauser, Boston.

Camacho, E. and Bordons, C. (2004). *Model predictive control*. Advanced textbooks in control and signal processing. Springer.

Cardaliaguet, P., Quincampoix, M., and Saint-Pierre, P. (1999). Set-valued numerical analysis for optimal control and differential games. In Bardi, M., Raghavan, T. E. S., and Parthasarathy, T., editors, *Stochastic and Differential Games: Theory and Numerical Methods*, volume 4 of *Annals of International Society of Dynamic Games*. Birkhäuser Boston.

Cecil, T., Qian, J., and Osher, S. (2004). Numerical methods for high dimensional hamilton-jacobi equations using radial basis functions. *Journal of Computational Physics*, 196(1):327 – 347.

Chen, T. and Francis, B. A. (1995). *Optimal Sampled-Data Control Systems*. 0178-5354. Springer-Verlag London.

Cristiani, E. and Falcone, M. (2009). Fully-discrete schemes for the value function of pursuit-evasion games with state constraint. In Bernhard, P., Gaitsgory, V., and Pourtallier, O., editors, *Advances in Dynamic Games and Their Applications: Analytical and Numerical Developments*, volume 10 of *Annals of International Society of Dynamic Games*, pages 178–205. Birkhäuser Boston.

Dehghan, M. and Salehi, R. (2010). The chebyshev spectral viscosity method for the time dependent eikonal equation. *Mathematical and Computer Modelling*, 52(1-2):70 – 86.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516.

Franklin, G. F., Workman, M. L., and Powell, D. (1997). *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition.

Grüne, L. and Pannek, J. (2011). *Nonlinear Model Predictive Control - Theory and Algorithms*. Communications and Control Engineering. Springer Verlag, London.

Habets, L. C. G. J. M., Collins, P. J., and van Schuppen, J. H. (2006). Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6):938–948.

Hu, C. and Shu, C.-W. (1999). A discontinuous galerkin finite element method for hamilton–jacobi equations. *SIAM Journal on Scientific Computing*, 21(2):666–690.

Isaacs, R. (1965). *Differential games; a mathematical theory with applications to warfare and pursuit, control and optimization*. John Wiley & Sons, New York.

Junge, O. and Schreiber, A. (2015). Dynamic programming using radial basis functions. *Discrete and Continuous Dynamical Systems*, 35(9):4439–4453.

Krasovskii, N. and Subbotin, A. (1988). *Game-theoretical control problems*. Springer-Verlag, New York.

Mitchell, I. M. (2007). A toolbox of level set methods. Technical Report TR-2007-11, UBC Department of Computer Science.

Ragazzini, J. R. and Franklin, G. F. (1958). *Sampled-Data Control Systems*. McGraw-Hill.