

AN EFFICIENT DP ALGORITHM ON A TREE-STRUCTURE FOR FINITE HORIZON OPTIMAL CONTROL PROBLEMS*

ALESSANDRO ALLA[†] MAURIZIO FALCONE[‡] LUCA SALUZZI[§]

Abstract. The classical Dynamic Programming (DP) approach to optimal control problems is based on the characterization of the value function as the unique viscosity solution of a Hamilton-Jacobi-Bellman (HJB) equation. The DP scheme for the numerical approximation of viscosity solutions of Bellman equations is typically based on a time discretization which is projected on a fixed state-space grid. The time discretization can be done by a one-step scheme for the dynamics and the projection on the grid typically uses a local interpolation. Clearly the use of a grid is a limitation with respect to possible applications in high-dimensional problems due to the *curse of dimensionality*.

Here, we present a new approach for finite horizon optimal control problems where the value function is computed using a DP algorithm with a tree structure algorithm (TSA) constructed by the time discrete dynamics. In this way there is no need to build a fixed space triangulation and to project on it. The tree will guarantee a perfect matching with the discrete dynamics and drop off the cost of the space interpolation allowing for the solution of very high-dimensional problems. Numerical tests will show the effectiveness of the proposed method.

Key words. dynamic programming, Hamilton-Jacobi-Bellman equation, optimal control, tree structure

AMS subject classifications. 49L20, 49J15, 49J20, 93B52

1. Introduction. The Dynamic Programming (DP) approach has been introduced and developed by Richard Bellman in the '50s in a series of pioneering papers (see e.g. [5]). Since then it has been applied to many problems in deterministic and stochastic optimal control although its real application has been up to now limited to low dimensional problems. Via the Dynamic Programming Principle (DPP) one can obtain a characterization of the value function as the unique viscosity solution of a nonlinear partial differential equation (the Hamilton-Jacobi-Bellman (HJB) equation) and then use the value function to get a synthesis of a feedback control law. This is the major advantage over the approach based on the Pontryagin Maximum Principle (PMP) [6,28] that gives necessary conditions for the characterization of the open-loop optimal control and of the corresponding optimal trajectory. As it is well known, the DP approach suffers from the *curse of dimensionality* since one has to solve a nonlinear partial differential equation (PDE) whose dimension is the same of the dynamical system. This has always been the main obstacle to apply that theory to real industrial applications despite the large number of theoretical results established for many classical control problems via the DP approach (see e.g. the monographies by Bardi and Capuzzo-Dolcetta [4] on deterministic control problems and by Fleming and Soner [17] on stochastic control problems). Even in low dimension this is a challenging problem since the value function associated to the control problem (i.e. the viscosity solution of the HJB equation) is known to be only Lipschitz continuous also when the dynamics and the running costs are regular functions. The numerical analysis of

[†]PUC-RIO, RUA MARQUES DE SAO VICENTE, 225, GÁVEA - RIO DE JANEIRO, RJ - BRASIL - 22451-900, ALLA@MAT.PUC-RIO.BR

[‡]SAPIENZA UNIVERSIT DI ROMA - PIAZZALE ALDO MORO 5, 00185 ROMA, FALCONE@MAT.UNIROMA1.IT

[§]GRAN SASSO SCIENCE INSTITUTE - VIALE F. CRISPI 7, 67100 L'AQUILA, LUCA.SALUZZI@GSSI.IT

* MF and LS would like to thank the support obtained by the 2018 INDAM -GNCS research project *Metodi numerici per problemi di controllo multiscala e applicazioni*.

low order numerical methods is now rather complete even for a state space in \mathbb{R}^d and several methods have been proposed to solve the HJB equation using a number of different techniques including finite differences, semi-Lagrangian, finite volumes and finite elements. We refer the interested reader to the monographies by Sethian [31], Osher and Fedkiw [26], Falcone and Ferretti [12] for an extensive discussion of some of these methods and for an extended list of references on numerical methods. All the above mentioned methods are based on a space discretization which requires the construction of a space grid (or triangulation). For higher dimensional problems the method needs a huge amount of memory allocations and makes the problem unfeasible for a dimension $d > 5$ on a standard computer. Several efforts have been made to mitigate the *curse of dimensionality*. Although a detailed description of these contributions goes beyond the scopes of this paper, we want to mention [14] for a domain decomposition method with overlapping between the subdomains and [10] for similar results without overlapping. It is important to note that in these papers the method is applied to subdomains with a rather simple geometry (see the book by Quarteroni and Valli [29] for a general introduction to this technique) to pass down conditions to the boundaries. More recently another way to decompose the problem has been proposed in [25] who have used a patchy decomposition based on Al’brekht method (see e.g. [1]). Later in [8] the patchy idea has been implemented taking into account an approximation of the underlying optimal dynamics to obtain subdomains which are almost invariant with respect to the optimal dynamics, clearly in this case the geometry of the subdomains can be rather complex but the transmission conditions at the internal boundaries can be eliminated saving on the overall complexity of the algorithm. More recently other decomposition techniques for optimal control problems and games have been proposed in [15] where the parallel algorithm is based on the construction of independent sub-domains and in [16] where a parallel version of the Howards algorithm is proposed and analyzed. In general, domain decomposition methods reduce a huge problem into subproblems of manageable size and allow to mitigate the storage limitation distributing the computation over several processors. However, the approximation schemes used in every subdomain are rather standard.

Another improvement can be obtained using efficient acceleration methods for the computation of the value function in every subdomain. In the framework of optimal control problems an efficient acceleration technique based on the coupling between value and policy iterations has been recently proposed and studied in [2]. The construction of a DP algorithm for time dependent problems has been addressed in [13] where also a-priori error estimates have been studied. An adaptation of similar methods for high-dimensional problems has been proposed later in [9].

However, we also mention that high-dimensional problems often imply a huge amount of data and are too complex to be solved even by a direct approach based on domain decomposition (this approach is typically feasible below dimension 10). A reasonable solution to attack high-dimensional problems is to apply first model order reduction techniques (e.g. Proper Orthogonal Decomposition [33]) to have a low dimensional version of the dynamics. Thus, if the reduced system of coordinates for the dynamics has a low number of dimension (e.g. $d \approx 5$) the problem can be solved via the DP approach. Model reduction techniques are based on orthogonal projections where the choice of the basis functions is non trivial, e.g. it requires to compute some reference trajectories corresponding to a priori given control strategies and compute the basis via an SVD. At the end of this step, the set of controlled trajectories will be represented as a linear combination of the basis functions. Whenever we are able to compute accurate projectors we drastically reduce the dimension of the control prob-

lem, say $\ell \ll d$ but we lose the physical meaning of the projected dynamical system. This makes it difficult to define a reasonable choice of the numerical domain Ω and the easiest solution is to choose Ω as a rather large box in \mathbb{R}^ℓ . We refer, among others, to the pioneer work on the coupling between model reduction and HJB approach [24] and the recent [3] work which provides a-priori error estimates for the aforementioned coupling method. We also mention a sparse grid approach in [18] where the authors apply HJB to the control of the wave equation and a spectral elements approximation in [23] which allows to solve the HJB equation up to dimension 12.

Despite these efforts and the mathematical elegance of the DP approach, its impact in industrial applications is limited by this bottleneck and the solution of many optimal control problems has been accomplished instead via open-loop control. More information on the topic can be found in the monographies by Hinze, Pinnau, Ulbrich Ulbrich [20] and by Tröltzsch [32].

The aim of this paper is to eliminate the space discretization and the construction of a grid to reduce the memory allocations and improve the applicability of the DP approach. This can be done for the finite horizon problem via the construction of a tree-structure that will account for the controlled dynamics. For numerical purposes, we will assume that the system has a finite number of controls at every time step t_n and, to simplify the presentation, we are keeping this number M constant during the evolution although the extension to a variable number M_n is straightforward. Under these hypotheses starting from a point x we can reach M points in the state space according to the discrete time dynamics. So a single starting point will produce a tree \mathcal{T} of order $O(M^{\bar{N}+1})$ points in \bar{N} time steps and the number of points is exponentially increasing as expected. Note that we will not compute the value function by the DP algorithm on that tree: exploiting the Lipschitz continuity of the value function in the space variable (see Section 2) we are going to prune the tree identifying the nodes that are "very close". The *pruning step* of the algorithm will be governed by a pruning parameter $\varepsilon_{\mathcal{T}}$ and at every step many branches will be cut away so the final complexity will be drastically reduced.

Working on the tree has several *advantages*:

- (i) we do not need to define a priori a numerical domain Ω where we want to solve the problem, the original tree is constructed by the controlled dynamics;
- (ii) we do not need to build a space grid and to make a space interpolation on the grid nodes, therefore we do not introduce the interpolation error;
- (iii) the pruned tree allows to deal with high-dimensional problems.

In conclusion, with respect to the standard space discretization we can drop the interpolation step that is rather expensive in high-dimension and we do not need the classical assumptions at the boundary of Ω which classically requires to have an invariant dynamics or to impose boundary conditions (Dirichlet, Neumann or state constraint).

Via the tree structure algorithm (TSA) we eliminate these difficulties at least for the finite horizon problem and we can directly solve the discrete time HJB equation for $d = 1000$ without any particular assumption on the structure of the problem as in model reduction context. This will be shown in Section 5.

The paper is organized as follows: in Section 2 we recall some basic facts about the time approximation of the finite horizon problem via the DP approach, we introduce our notation and prove that the discrete time value function is Lipschitz continuous in space. Section 3 is devoted to present the construction of the tree-structure related to the controlled dynamics. In Section 4, we present some hints on the actual im-

plementation of the method, in particular the pruning technique used to cut off the branches of the tree in order to reduce the global complexity of the algorithm. Some numerical tests are presented and analyzed in Section 5. We give our conclusions and perspectives in Section 6.

2. Finite horizon optimal control problems via dynamic programming principle. In this section we will summarize the basic results that will constitute the building blocks for our new algorithm. The essential features will be briefly sketched, and more details can be found in [4,12] and the references therein. Let us present the method for the classical *finite horizon problem*. Let the system be driven by

$$(2.1) \quad \begin{cases} \dot{y}(s) = f(y(s), u(s), s), & s \in (t, T], \\ y(t) = x \in \mathbb{R}^d. \end{cases}$$

We will denote by $y : [t, T] \rightarrow \mathbb{R}^d$ the solution, by u the control $u : [t, T] \rightarrow \mathbb{R}^m$, by $f : \mathbb{R}^d \times \mathbb{R}^m \times [t, T] \rightarrow \mathbb{R}^d$ the dynamics and by

$$\mathcal{U} = \{u : [t, T] \rightarrow U, \text{measurable}\}$$

the set of admissible controls where $U \subset \mathbb{R}^m$ is a compact set. We assume that there exists a unique solution for (2.1) for each $u \in \mathcal{U}$.

The cost functional for the finite horizon optimal control problem will be given by

$$(2.2) \quad J_{x,t}(u) := \int_t^T L(y(s, u), u(s), s) e^{-\lambda(s-t)} ds + g(y(T)) e^{-\lambda(T-t)},$$

where $L : \mathbb{R}^d \times \mathbb{R}^m \times [t, T] \rightarrow \mathbb{R}$ is the running cost, $g : \mathbb{R}^d \rightarrow \mathbb{R}$ is the final cost and $\lambda \geq 0$ is the discount factor.

The goal is to find a state-feedback control law $u(t) = \Phi(y(t), t)$, in terms of the state variable $y(t)$, where Φ is the feedback map. To derive optimality conditions we use the well-known DPP due to Bellman. We first define the value function for an initial condition $(x, t) \in \mathbb{R}^d \times [t, T]$:

$$(2.3) \quad v(x, t) := \inf_{u \in \mathcal{U}} J_{x,t}(u)$$

which satisfies the DPP, i.e. for every $\tau \in [t, T]$:

$$(2.4) \quad v(x, t) = \inf_{u \in \mathcal{U}} \left\{ \int_t^\tau L(y(s), u(s), s) e^{-\lambda(s-t)} ds + v(y(\tau), \tau) e^{-\lambda(\tau-t)} \right\}.$$

Due to (2.4) we can derive the HJB for every $x \in \mathbb{R}^d$, $s \in [t, T]$:

$$(2.5) \quad \begin{cases} -\frac{\partial v}{\partial s}(x, s) + \lambda v(x, s) + \max_{u \in U} \{-L(x, u, s) - \nabla v(x, s) \cdot f(x, u, s)\} = 0, \\ v(x, T) = g(x). \end{cases}$$

Suppose that the value function is known, by e.g. (2.5), then it is possible to compute the optimal feedback control as:

$$(2.6) \quad u^*(t) := \arg \max_{u \in U} \{-L(x, u, t) - \nabla v(x, t) \cdot f(x, u, t)\}.$$

Equation (2.5) is a nonlinear PDE of the first order which is hard to solve analytically although a general theory of weak solutions is available in e.g. [4]. Rather, we can solve equation (2.5) numerically by means of finite difference or semi-Lagrangian methods. In the current work we recall the semi-Lagrangian method. One usually starts the numerical method by discretizing in time the underlying control problem with a time step $\Delta t := [(T-t)/\bar{N}]$ where \bar{N} is the number of temporal time steps and then projects the semi-discrete scheme on a grid obtaining the fully discrete scheme:

$$(2.7) \quad \begin{cases} V_i^n = \min_{u \in U} [\Delta t L(x_i, u, t_n) + e^{-\lambda \Delta t} I[V^{n+1}](x_i + \Delta t f(x_i, u, t_n))], \\ V_i^{\bar{N}} = g(x_i) \end{cases} \quad \begin{matrix} n = \bar{N} - 1, \dots, 0, \\ x_i \in \Omega, \end{matrix}$$

where $t_n = t + n\Delta t$, $t_{\bar{N}} = T$, Ω is the numerical domain and x_i is an element of its discretization, $V_i^n := V(x_i, t_n)$ and $I[\cdot]$ is an interpolation operator which is necessary to compute the value of V^n at the point $x_i + \Delta t f(x_i, u, t_n)$ (in general, this point will not be a node of the grid). The interested reader will find in [13] a detailed presentation of the scheme and a priori error estimates for its numerical approximation. We note that it is possible to show that the value function $v(x, t)$ is Lipschitz continuous on compact sets provided that f, L and g are Lipschitz continuous with constant $L_f, L_L, L_g > 0$ respectively. It is possible to extend the result for the numerical value function $V(x, t)$ as explained in the following proposition. The proof follows closely from the continuous version in [4, Prop. 3.1].

PROPOSITION 2.1. *Let us suppose the functions $f(\cdot, u, t), L(\cdot, u, t)$ and $g(\cdot)$ are Lipschitz continuous uniformly with respect to the other variables. Then, the numerical value function $V^n(x)$ is Lipschitz in x*

$$(2.8) \quad |V^n(x) - V^n(y)| \leq \begin{cases} |x - y| \left(\frac{L_L}{L_f - \lambda} (e^{(T-t_n)(L_f - \lambda)} - 1) + L_g e^{(T-t_n)(L_f - \lambda)} \right), & \text{for } L_f > \lambda, \\ |x - y| (L_L(T - t_n) + L_g e^{(T-t_n)(L_f - \lambda)}), & \text{for } L_f \leq \lambda, \end{cases}$$

$\forall x, y \in \mathbb{R}^d$ and $n = 0, \dots, \bar{N}$.

Proof. In the case $n = \bar{N}$, we have that $V^{\bar{N}}(x) = g(x)$, then the estimate follows directly from the hypothesis on g .

In the case $n < \bar{N}$, we fix $\bar{x}, \bar{y} \in \mathbb{R}^d$ and consider the following quantity $V^n(\bar{x}) - V^n(\bar{y})$:

$$(2.9) \quad \begin{aligned} V^n(\bar{x}) - V^n(\bar{y}) &\leq e^{-\lambda \Delta t} V^{n+1}(\bar{x} + \Delta t f(\bar{x}, u_*^n, t_n)) + \Delta t L(\bar{x}, u_*^n, t_n) \\ &\quad - e^{-\lambda \Delta t} V^{n+1}(\bar{y} + \Delta t f(\bar{y}, u_*^n, t_n)) - \Delta t L(\bar{y}, u_*^n, t_n) \\ &\leq e^{-\lambda \Delta t} (V^{n+1}(\bar{x} + \Delta t f(\bar{x}, u_*^n, t_n)) - V^{n+1}(\bar{y} + \Delta t f(\bar{y}, u_*^n, t_n))) \\ &\quad + \Delta t L_L |\bar{x} - \bar{y}|, \end{aligned}$$

provided that

$$u_*^n = \arg \min_{u \in U} \{ e^{-\lambda \Delta t} V^{n+1}(\bar{y} + \Delta t f(\bar{y}, u, t_n)) + \Delta t L(\bar{y}, u, t_n) \}.$$

To achieve the desired estimate (2.8), we need to iterate (2.9) starting from \bar{x} and \bar{y}

at time t_n . Let us first define the whole tree paths $\{x^m\}_m$ and $\{y^m\}_m$ as

$$x^m := x^n + \Delta t \sum_{j=n}^{m-1} f(x^j, u_*^j, t_j), \quad y^m := y^n + \Delta t \sum_{j=n}^{m-1} f(y^j, u_*^j, t_j),$$

where

$$u_*^j = \arg \min_{u \in U} \{e^{-\lambda \Delta t} V^{j+1}(y^j + \Delta t f(y^j, u, t_j)) + \Delta t L(y^j, u, t_j)\}, \quad j = n, \dots, m-1.$$

By the discrete Grönwall's lemma, it is easy to prove the following estimate for Euler schemes starting from $x^n = \bar{x}$ and $y^n = \bar{y}$

$$(2.10) \quad |x^{n+k} - y^{n+k}| \leq |x^n - y^n| e^{k \Delta t L_f} = |\bar{x} - \bar{y}| e^{k \Delta t L_f}, \quad k = 0, \dots, \bar{N} - n.$$

Then, iterating (2.9) we obtain

$$(2.11) \quad \begin{aligned} V^n(\bar{x}) - V^n(\bar{y}) &\leq \Delta t L_L \sum_{k=0}^{\bar{N}-n-1} e^{-\lambda k \Delta t} |x^{n+k} - y^{n+k}| + e^{-\lambda(T-t_n)} |g(x^{\bar{N}}) - g(y^{\bar{N}})| \\ &\leq \Delta t L_L \sum_{k=0}^{\bar{N}-n-1} e^{-\lambda k \Delta t} |x^{n+k} - y^{n+k}| + L_g e^{-\lambda(T-t_n)} |\bar{x} - \bar{y}| \\ &\leq |\bar{x} - \bar{y}| \left(\Delta t L_L \sum_{k=0}^{\bar{N}-n-1} e^{k \Delta t (L_f - \lambda)} + L_g e^{(T-t_n)(L_f - \lambda)} \right), \end{aligned}$$

where we used (2.10) and the Lipschitz continuity of g .

If $L_f > \lambda$, then by (2.11) and the equality $(\bar{N} - n)\Delta t = T - t_n$, we get

$$(2.12) \quad \begin{aligned} V^n(\bar{x}) - V^n(\bar{y}) &\leq |\bar{x} - \bar{y}| \left(\Delta t L_L \frac{e^{(T-t_n)(L_f - \lambda)} - 1}{e^{\Delta t (L_f - \lambda)} - 1} + L_g e^{(T-t_n)(L_f - \lambda)} \right) \\ &\leq |\bar{x} - \bar{y}| \left(\frac{L_L}{L_f - \lambda} (e^{(T-t_n)(L_f - \lambda)} - 1) + L_g e^{(T-t_n)(L_f - \lambda)} \right), \end{aligned}$$

whereas if $L_f \leq \lambda$, noticing that $e^{k \Delta t (L_f - \lambda)} \leq 1$, we directly obtain

$$(2.13) \quad V^n(\bar{x}) - V^n(\bar{y}) \leq |\bar{x} - \bar{y}| \left(L_L (T - t_n) + L_g e^{(T-t_n)(L_f - \lambda)} \right).$$

Analogously, it is possible to obtain the same estimate for $V^n(\bar{y}) - V^n(\bar{x})$ which leads to the desired result. \square

In the next section we will take advantage of the estimate (2.9) to guarantee the feasibility of our proposed method. The numerical approximation of the feedback control (2.6) follows directly from the SL-scheme (2.7) and reads

$$u_*^n(x) = \arg \min_{u \in U} [\Delta t L(x, u, t_n) + e^{-\lambda \Delta t} I[V^{n+1}](x + \Delta t f(x, u, t_n))].$$

3. HJB on a tree structure. The DP approach for the numerical approximation of viscosity solutions of the HJB equation is typically based on a time discretization which is projected on a fixed state-space grid of the numerical domain. The choice of the numerical domain is already one bottleneck of the method. In fact, although the theory is valid in the whole space \mathbb{R}^d for computational reasons we need to restrict to a compact set in \mathbb{R}^d which should be large enough to include all the possible trajectories. That also yields the selection of some boundary conditions which are not trivial.

In this section we will provide a novel algorithm which does not require a state-space grid and therefore avoids (i) the choice of the numerical domain, (ii) the computation of polynomial interpolation, (iii) the selection of boundary conditions and finally (iv) we can solve the problem for larger dimension, such as $d \gg 5$ (in Section 5 we provide an example in dimension 1000). Note that dimension 5 was the maximum dimension for SL-schemes based on a grid on a standard computer (see e.g. [3]).

Construction of the tree data structure. We build the nodes tree \mathcal{T} starting from a given initial condition x and following directly the dynamics in (2.1) discretized by e.g. Euler method. Since we only discretize in time, we set a temporal step Δt which divides the interval $[t, T]$ into \bar{N} subintervals. We note that $\mathcal{T} := \cup_{j=0}^{\bar{N}} \mathcal{T}^j$, where each \mathcal{T}^j contains the nodes of the tree correspondent to time t_j . The first level $\mathcal{T}^0 = \{x\}$ is simply given by the initial condition x . To compute the second level, and the other levels we suppose to discretize the control domain U with step-size Δu . We denote that the control set U is a subset in \mathbb{R}^m , in particular we will consider U as a hypercube, discretized in all directions with constant step-size Δu , obtaining $U^{\Delta u} = \{u_1, \dots, u_M\}$. To ease the notation in the sequel we continue to denote by U the discrete set of controls. Then, starting from the initial condition x , we consider all the nodes obtained following the dynamics (2.1) discretized using e.g. an explicit Euler scheme with different discrete controls $u_j \in U$

$$\zeta_j^1 = x + \Delta t f(x, u_j, t_0), \quad j = 1, \dots, M.$$

Therefore, we have $\mathcal{T}^1 = \{\zeta_1^1, \dots, \zeta_M^1\}$. We note that all the nodes can be characterized by their n -th *time level*, as in the following definition.

DEFINITION 3.1. *The general n -th level of the tree will be composed by M^n nodes denoted by*

$$\mathcal{T}^n = \{\zeta_i^{n-1} + \Delta t f(\zeta_i^{n-1}, u_j, t_{n-1})\}_{j=1}^M \quad i = 1, \dots, M^{n-1}.$$

We show in the left panel of Figure 1 the structure of the whole tree \mathcal{T} . All the nodes of the tree can be shortly defined as

$$\mathcal{T} := \{\zeta_j^n\}_{j=1}^{M^n}, \quad n = 0, \dots, \bar{N},$$

where the nodes ζ_i^n are the result of the dynamics at time t_n with the controls $\{u_{j_k}\}_{k=0}^{n-1}$:

$$\begin{aligned} \zeta_{i_n}^n &= \zeta_{i_{n-1}}^{n-1} + \Delta t f(\zeta_{i_{n-1}}^{n-1}, u_{j_{n-1}}, t_{n-1}) \\ &= x + \Delta t \sum_{k=0}^{n-1} f(\zeta_{i_k}^k, u_{j_k}, t_k), \end{aligned}$$

with $\zeta^0 = x$, $i_k = \left\lceil \frac{i_{k+1}}{M} \right\rceil$ and $j_k \equiv i_{k+1} \bmod M$, where $\lceil \cdot \rceil$ is the ceiling function. We note that $\zeta_i^k \in \mathbb{R}^d$, $i = 1, \dots, M^k$. On the right panel of Figure 1 we show the path to

reach for instance ζ_{26}^4 if the control set contains only three elements. We, again, would like to emphasize that the domain is not chosen a priori, but constructed following the dynamics.

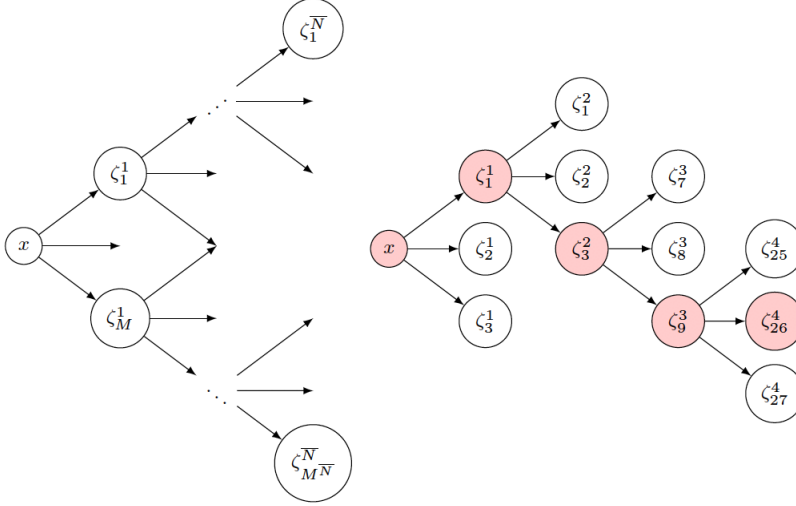


FIG. 1. Example of the tree \mathcal{T} (left), path to reach ζ_{26}^4 starting from the initial condition x with $U = \{u_1, u_2, u_3\}$ (right).

In what follows we provide two remarks about the properties of the tree \mathcal{T} under some particular assumptions on the dynamics f .

REMARK 3.1. Let us suppose that the dynamics is affine with respect to u and that $u \in [u_{min}, u_{max}] \subset \mathbb{R}$, e.g. the following decomposition holds true

$$f(x, u, t) = f_1(x, t)u + f_2(x, t).$$

Then, all the nodes in \mathcal{T}^n will lie on the segment with extremal points given by the controls at the boundary $\partial U = \{u_1 = u_{min}, u_M = u_{max}\}$. Specifically,

$$\text{if } z \in \mathcal{T}^n \text{ this implies } z \in [\zeta_{i_1}^n, \zeta_{i_M}^n]$$

where $\zeta_{i_1}^n$ and $\zeta_{i_M}^n$ are obtained by using the control u_1 and u_M respectively.

REMARK 3.2. Let us suppose that the dynamics is monotone with respect to $u \in [u_{min}, u_{max}] \subset \mathbb{R}$:

$$\begin{aligned} \min_{\tilde{u} \in \{u_{min}, u_{max}\}} f_j(x, \tilde{u}, t) \leq f_j(x, u, t) \leq \max_{\tilde{u} \in \{u_{min}, u_{max}\}} f_j(x, \tilde{u}, t), \\ \forall u \in [u_{min}, u_{max}], j = 1, \dots, d. \end{aligned}$$

Then the nodes of the tree will belong to a box with vertices given by the coordinates of the nodes obtained with the extremal controls u_{min} and u_{max} as follows:

$$\min_{\bar{i} \in \{i_1, i_M\}} \zeta_{\bar{i}}^n \leq \zeta_i^n \leq \max_{\bar{i} \in \{i_1, i_M\}} \zeta_{\bar{i}}^n, \quad i \in \{i_1, \dots, i_M\},$$

where the last inequality holds component-wise.

Approximation of the value function. The numerical value function $V(x, t)$ will be computed on the tree nodes in space, whereas in time it will be approximated as a piecewise constant function, *i.e.*

$$V(x, t) = V^n(x) \quad \forall x, \text{ and } t \in [t_n, t_{n+1}),$$

where $t_n = t + n\Delta t$.

We note that we start to approximate the value function once the tree \mathcal{T} has been already built. Then, we will be able to approximate the value function $V^n(x_i + \Delta t f(x_i, u, t_n))$ in (2.7) without the use of an interpolation operator on a grid. The reason is that we build our domain according to all the possible directions of the dynamics for a discrete set of controls and, as a consequence, all the nodes $x_i + \Delta t f(x_i, u, t_n)$ will belong to the grid. It is now straightforward to evaluate the value function. The TSA defines a grid $\mathcal{T}^n = \{\zeta_j^n\}_{j=1}^{M^n}$ for $n = 0, \dots, \bar{N}$, we can approximate (2.5) as follows:

$$(3.1) \quad \begin{cases} V^n(\zeta_i^n) = \min_{u \in U} \{e^{-\lambda \Delta t} V^{n+1}(\zeta_i^n + \Delta t f(\zeta_i^n, u, t_n)) + \Delta t L(\zeta_i^n, u, t_n)\}, \\ \zeta_i^n \in \mathcal{T}^n, n = \bar{N} - 1, \dots, 0, \\ V^{\bar{N}}(\zeta_i^{\bar{N}}) = g(\zeta_i^{\bar{N}}), \\ \zeta_i^{\bar{N}} \in \mathcal{T}^{\bar{N}}. \end{cases}$$

We note that the minimization is computed by comparison on the discretized set of controls U . We refer to [7,22] for a more sophisticated approach to compute the minimum in (2.7).

REMARK 3.3. *If the dynamics (2.1) is autonomous the evolution of the dynamics will not depend explicitly on t_n and the problem can be simplified since the argument of the minimization in (3.1) will be*

$$e^{-\lambda \Delta t} V^{n+1}(\zeta + \Delta t f(\zeta, u)) + \Delta t L(\zeta, u, t_n).$$

At the time t_n we have n levels of the tree on the left and $\bar{N} - n$ levels on the right (till $t_{\bar{N}}$). Since the computation is going backward, to compute the value function at time t_n , we need to do $\bar{N} - n$ steps in time starting from the final condition at time T . Once we know V^n this information can also be interpreted as a final condition for the sub-tree $\cup_{k=0}^n \mathcal{T}^k$ and, since the dynamics is autonomous, we can proceed backward computing V^{n-1} for the nodes belonging to all the k -th time levels, for $k \leq n - 1$. Indeed the nodes $\zeta + \Delta t f(\zeta, u)$ do not depend explicitly on the time and they can be involved in the computation of the value function at different time steps (this is not the case for a non-autonomous dynamics). Thus, we will proceed as follows: first we impose the final cost g on the whole tree, then we start computing the value function backward. This procedure leads to a more extensive knowledge of the value function on the tree.

4. Hints on the algorithm. In this section we will provide further details on the implementation of the method proposed in Section 3. We will explain how to reduce the number of tree nodes to make the problem feasible, compute the feedback control and recall the whole procedure.

Pruning the tree. The proposed method mitigates the curse of dimensionality and it allows to deal with problems in \mathbb{R}^d with $d \gg 5$, which is absolutely not feasible

with the classical approach. However, we still have dimensionality problem related to the amount of nodes in the tree \mathcal{T} . In fact, given $M > 1$ controls and \bar{N} time steps, the cardinality of the tree is

$$|\mathcal{T}| = \sum_{i=0}^{\bar{N}} M^i = \frac{M^{\bar{N}+1} - 1}{M - 1},$$

which is infeasible due to the huge amount of memory allocations, if M or \bar{N} are too large. Therefore, we suggest to select the nodes of the TSA neglecting those very close to each other, assuming that the value function will not be completely different on those nodes, e.g.

$$\zeta_i^n \approx \zeta_j^n \implies V(\zeta_i^n) \approx V(\zeta_j^n).$$

This is a realistic assumption since the numerical value function is Lipschitz continuous as explained in Proposition 2.1. We can introduce the *pruning rule*.

DEFINITION 4.1 (Pruning rule). *Two given nodes ζ_i^n and ζ_j^n can be merged if*

$$(4.1) \quad \|\zeta_i^n - \zeta_j^n\| \leq \varepsilon_{\mathcal{T}}, \quad \text{with } n = 0, \dots, \bar{N},$$

for a given threshold $\varepsilon_{\mathcal{T}} > 0$.

Specifically, if during the construction of the tree, a node ζ^{n-1} has as a son a new node ζ_j^n which verifies (4.1) with a certain ζ_i^n , then we will not add the new node to the tree and we will connect the node ζ^{n-1} with ζ_i^n . We cut the node which verifies the criteria before going on with the construction of the tree, in this way we avoid the sub-tree coming out from this node, saving a huge amount of memory.

The cut of the tree works as follows: during the construction of the n -th level, the new node will be compared with the previous nodes already computed at the same level n . If the new node ζ_j^n , whose father is ζ^{n-1} , satisfies the condition (4.1) with a node ζ_i^n , the new node will not be added to the tree and the adjacency matrix will be updated, connecting the node ζ^{n-1} to the node ζ_i^n . Figure 2 provides a graphic idea about the application of the pruning criteria. The choice of the tolerance plays an

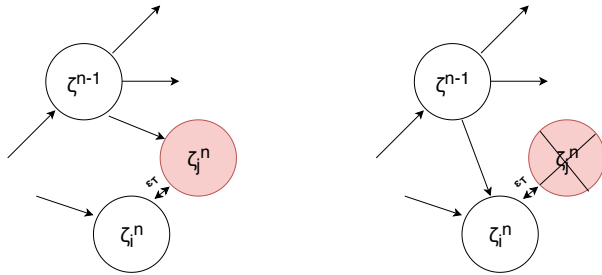


FIG. 2. *Pruning technique throughout the construction of the tree it might happen the two nodes are very close (left) and we link those node in order to prune the tree.*

important role: if $\varepsilon_{\mathcal{T}}$ is very small, the algorithm will be very slow, whereas if it is too large, we will not obtain an accurate approximation. A reasonable choice turns out to be $\varepsilon_{\mathcal{T}} = \Delta t^2$, as shown in Section 5. The interested reader will find a rigorous proof of this heuristic statement in [30] together with convergence results of the proposed method.

REMARK 4.1 (Pruning rule in the autonomous case). *If the dynamics is autonomous, as explained in Remark 3.3, we can extend the computation of the value function at time t_n even for nodes belonging to the subtree $\cup_{k=0}^n \mathcal{T}^k$. Therefore, we can extend the pruning criteria (4.1) as follows. Two given nodes ζ_i^n and ζ_j^m can be merged if*

$$(4.2) \quad \|\zeta_i^n - \zeta_j^m\| \leq \varepsilon_{\mathcal{T}}, \quad \text{with } n, m = 0, \dots, \bar{N},$$

for a given threshold $\varepsilon_{\mathcal{T}} > 0$.

REMARK 4.2 (Efficient Pruning). *The computation of the distances among all the nodes would be very expensive, especially for high dimensional problems. Hence, we need an efficient algorithm to compute the distances quickly. One possible strategy is the Principal Analysis Component ([27],[21]). Our aim is to project the data onto a lower dimensional linear space such that the variance of the projected data is maximized. This can be done e.g. computing the Singular Value Decomposition of the data matrix and taking the first basis. Once we project the data, the distances will be computed in a lower dimension space and this turns out to accelerate the algorithm.*

Feedback reconstruction and closed-loop control. During the computation of the value function, we store the control indices corresponding to the argmin in (3.1). Then starting from $\zeta_*^0 = x$, we follow the path of the tree to build the optimal trajectory $\{\zeta_*^n\}_{n=0}^{\bar{N}}$ in the following way

$$(4.3) \quad u_n^* := \arg \min_{u \in U} \{e^{-\lambda \Delta t} V^{n+1}(\zeta_*^n + \Delta t f(\zeta_*^n, u, t_n)) + \Delta t L(\zeta_*^n, u, t_n)\},$$

$$\zeta_*^{n+1} \in \mathcal{T}^{n+1} \text{ s.t. } \zeta_*^n \rightarrow^{u_n^*} \zeta_*^{n+1},$$

for $n = 0, \dots, \bar{N} - 1$, where the symbol \rightarrow^u stands for the connection of two nodes by the control u . We note that this is possible because in the current work we assume to consider the same discrete control set U for both HJB equation (3.1) and feedback reconstruction (4.3).

Algorithm. In what follows we summarize the whole algorithm including the construction of the tree, the selection of the nodes and, finally, the approximation of the value function.

Algorithm 1 TSA algorithm with pruning

- 1: $\mathcal{T}^0 \leftarrow x$
 - 2: **for** $n = 1, \dots, \bar{N}$ **do**
 - 3: **for** $u_j \in U, \zeta^{n-1} \in \mathcal{T}^{n-1}$ **do**
 - 4: $\zeta_{new} = \zeta^{n-1} + \Delta t f(\zeta^{n-1}, u_j, t_{n-1})$
 - 5: **if** $\|\zeta_{new} - \zeta\| > \varepsilon_{\mathcal{T}}, \forall \zeta \in \mathcal{T}$ **then**
 - 6: $\mathcal{T}^n \leftarrow \zeta_{new}$
 - 7: $\zeta^{n-1} \rightarrow^u \zeta_{new}$
 - 8: **else**
 - 9: $\bar{\zeta} = \arg \min_{\zeta \in \mathcal{T}} \|\zeta_{new} - \zeta\|$
 - 10: $\zeta^{n-1} \rightarrow^u \bar{\zeta}$
 - 11: $V^{\bar{N}}(\zeta) = g(\zeta), \forall \zeta \in \mathcal{T}^{\bar{N}}$
 - 12: **for** $n = \bar{N} - 1, \dots, 0$ **do**
 - 13: $V^n(\zeta^n) = \min_{\zeta^{n+1}: \zeta^n \rightarrow^u \zeta^{n+1}} \{e^{-\lambda \Delta t} V^{n+1}(\zeta^{n+1}) + \Delta t L(\zeta^n, u, t_n)\}, \quad \zeta^n \in \mathcal{T}^n.$
-

As one can see in Algorithm 1, we first start the construction of the tree \mathcal{T} from 1 to step 10. We note that the pruning criteria is involved in the steps 5-10 of Algorithm 1. Clearly, a very small tolerance will not allow any selection of the nodes and we will work with a full tree. Finally, in step 11-12-13 we compute the approximation of the value function. In the last step, the computation of the value function $V^n(\zeta^n)$ can be extended to the nodes in the tree $\cup_{k=0}^n \mathcal{T}^k$ in the case of autonomous dynamics.

5. Numerical tests. In this section we are going to apply the proposed algorithm to show the effectiveness of the method.

We will present five test cases. In the first we are able to compute the analytical solution of the HJB equation and, therefore, to compute the error with our method compared to the classical approach, see e.g. [13]. The second test concerns the well-known Van der Pol equation and the third is about non-autonomous dynamics. Finally we present the results for two different linear PDEs which shows the power of the method even for large-scale problems.

The numerical simulations reported in this paper are performed on a laptop with 1CPU Intel Core i5-3,1 GHz and 8GB RAM. The codes are written in C++.

5.1. Test 1: Comparison with exact solution of the value function. In the first example we consider the following dynamics in (2.1)

$$(5.1) \quad f(x, u) = \begin{pmatrix} u \\ x_1^2 \end{pmatrix}, \quad u \in U \equiv [-1, 1],$$

where $x = (x_1, x_2) \in \mathbb{R}^2$. The cost functional in (2.2) is:

$$(5.2) \quad L(x, u, t) = 0, \quad g(x) = -x_2, \quad \lambda = 0,$$

where we only consider the terminal cost g . The corresponding HJB equation is

$$(5.3) \quad \begin{cases} -V_t + |V_{x_1}| - x_1^2 V_{x_2} = 0 & (x, t) \in \mathbb{R}^2 \times [0, T], \\ V(x, T) = g(x), \end{cases}$$

where its unique viscosity solution reads

$$(5.4) \quad V(x, t) = -x_2 - x_1^2(T - t) - \frac{1}{3}(T - t)^3 - |x_1|(T - t)^2.$$

Furthermore, we set $T = 1$. Figure 3 shows the contour lines of the value function $V(x, t)$ in (5.4) for time instances $t = \{0, 0.5, 1\}$.

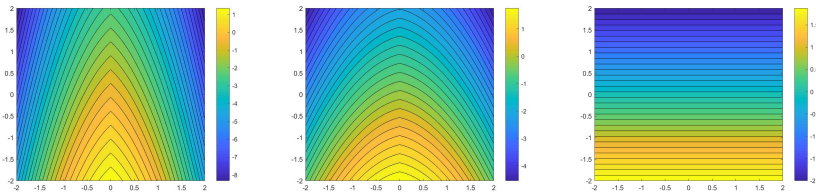


FIG. 3. Test 1: Contour lines for (5.4) with $t = 0$ (left), $t = 0.5$ (middle) and $t = 1$ (right).

In this example, we compare the classical approach with the TSA algorithm proposed in Algorithm 1 using both strategies: (i) no selection of the nodes and (ii)

applying criteria (4.1) to select the nodes as explained in Section 4. To perform a fair comparison we projected the value function computed with the classical method into the tree nodes. We note that it will not modify the accuracy of the classical approach since the interpolation has to be performed also on a structured grid. We compare the different approximations according to ℓ_2 -relative error with the exact solution on the tree nodes

$$\mathcal{E}_2(t_n) = \sqrt{\frac{\sum_{x_i \in \mathcal{T}^n} |v(x_i, t_n) - V^n(x_i)|^2}{\sum_{x_i \in \mathcal{T}^n} |v(x_i, t_n)|^2}},$$

where $v(x_i, t_n)$ represents the analytical solution and $V^n(x_i)$ its numerical approximation.

In Figure 4, we show all the nodes of the tree \mathcal{T} for the initial condition $x = (-0.5, 0.5)$, $\Delta t = 0.05$ and different choices of $\varepsilon_{\mathcal{T}} = \{0, \Delta t^2\}$. We note that there is a huge difference between the cardinality of the trees, that is $|\mathcal{T}| = 2097151$ when the tolerance is not applied whereas we have $|\mathcal{T}| = 3151$ for $\varepsilon_{\mathcal{T}} = \Delta t^2$.

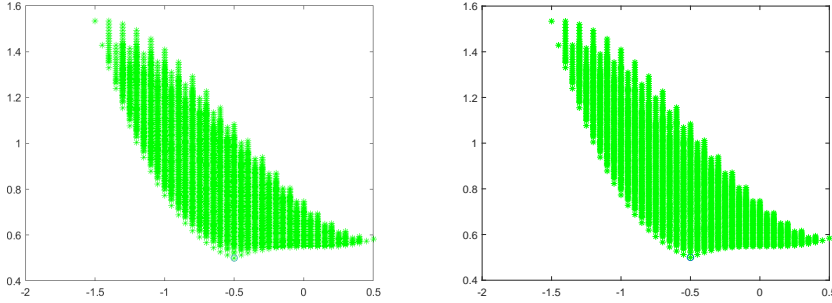


FIG. 4. *Test 1: Tree nodes without tolerance (left) and with tolerance equal to Δt^2 (right) for $x = (-0.5, 0.5)$.*

In Figure 5, we show the behaviour of the error \mathcal{E}_2 for two different initial conditions x . We note that its behaviour is very similar using both the classical approach and the TSA with or without the pruning criteria (4.1) for the nodes. As already mentioned, we would like to stress that the domain for the solution of the classical approach is chosen as large as possible to avoid that the boundary conditions are active, whereas with TSA we do not have this kind of problem, since the domain of the tree constructed according to the vector field. We note that to compute the value function in the classical approach we use the following step size: $\Delta x = \Delta t = 0.05$.

In Table 1 we show the error decay decreasing the temporal step size Δt for $x = (-0.5, 0.5)$ and $\varepsilon_{\mathcal{T}} = 0$ (i.e no pruning criteria has been applied). We compute the error as follows:

$$Err_{2,2} = \sqrt{\Delta t \sum_{n=0}^{\bar{N}} \mathcal{E}_2^2(t_n)}, \quad Err_{\infty,2} = \max_{n=0, \dots, \bar{N}} \mathcal{E}_2(t_n)$$

and the order

$$Order_{2,2} = \log_2 \left(\frac{Err_{2,2}(\Delta t)}{Err_{2,2}(\Delta t/2)} \right), \quad Order_{\infty,2} = \log_2 \left(\frac{Err_{\infty,2}(\Delta t)}{Err_{\infty,2}(\Delta t/2)} \right).$$

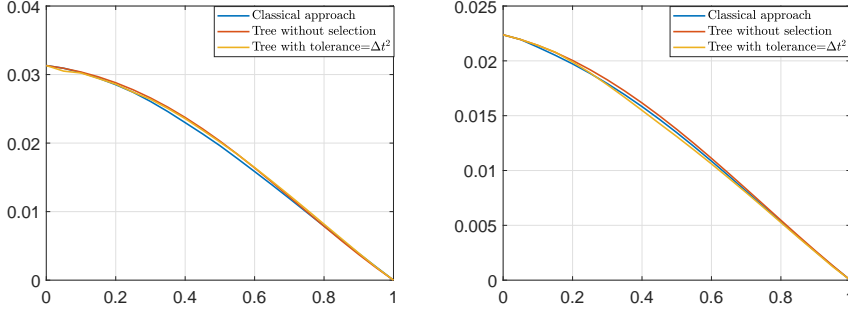


FIG. 5. Test 1: Comparison of the different methods with initial datum $(-0.5, 0.5)$ (left) and with initial datum $(1, 1)$ (right) for each time instance (x -axis).

We note that the order of convergence is linear as the order of the method used to discretize the dynamics (2.1), e.g. forward Euler scheme. This feature will be analyzed in a follow-up paper where we would like to provide error estimate for our proposed algorithm.

Δt	$ \mathcal{T} $	CPU	$Err_{2,2}$	$Err_{\infty,2}$	$Order_{2,2}$	$Order_{\infty,2}$
0.2	63	0.05s	0.090	0.122		
0.1	2047	0.35s	0.044	0.062	1.04	0.98
0.05	2097151	1.1s	0.022	0.031	1.02	0.99

TABLE 1

Test 1: Error analysis and order of convergence of the TSA without pruning rule.

However, the case without selection is quite unfeasible for more than 20 time steps since it requires to store a huge amount of nodes of order $O(M^{21})$, whereas with the selection we can obtain an impressive improvement. The results are shown in Table 2 where we can see, although the pruning of the nodes, we are still able to achieve an order of convergence close to 1.

Δt	$ \mathcal{T} $	CPU	$Err_{2,2}$	$Err_{\infty,2}$	$Order_{2,2}$	$Order_{\infty,2}$
0.2	42	0.05s	0.091	0.122		
0.1	324	0.08s	0.044	0.062	1.05	0.98
0.05	3151	0.1s	0.021	0.031	1.04	0.99
0.025	29248	0.5s	0.011	0.016	1.005	0.994
0.0125	252620	10s	0.005	0.008	1.004	0.997

TABLE 2

Test 1: Error analysis and order of convergence of the TSA with $\varepsilon_{\mathcal{T}} = \Delta t^2$ and $T = 1$.

The tolerance $\varepsilon_{\mathcal{T}}$ has been set equal to Δt^2 to keep the same order of convergence of the algorithm as the one without pruning. This is shown in Figure 6, where we compare the orders of the method with different tolerances. We note that we need to reduce the tolerance to Δt^2 to ensure linear convergence.

Furthermore, the pruned TSA allows to approximate HJB equation with rather small Δt and large horizon, e.g. $T = 3$ in a fast way, as shown in Table 3 keeping the order of convergence found in the previous case. Finally, for the sake of completeness

Δt	$ \mathcal{T} $	CPU	$Err_{2,2}$	$Err_{\infty,2}$	$Order_{2,2}$	$Order_{\infty,2}$
0.2	1420	0.2s	0.124	0.088		
0.1	15231	0.11s	0.061	0.045	1.02	0.98
0.05	141142	4s	0.030	0.022	1.03	1.01
0.025	1204637	147s	0.015	0.011	1.009	1.002
0.0125	10037898	7171s	0.007	0.006	1.009	1.004

TABLE 3

Test 1: Error analysis and order of convergence of the TSA with $\varepsilon_{\mathcal{T}} = \Delta t^2$ and $T = 3$.

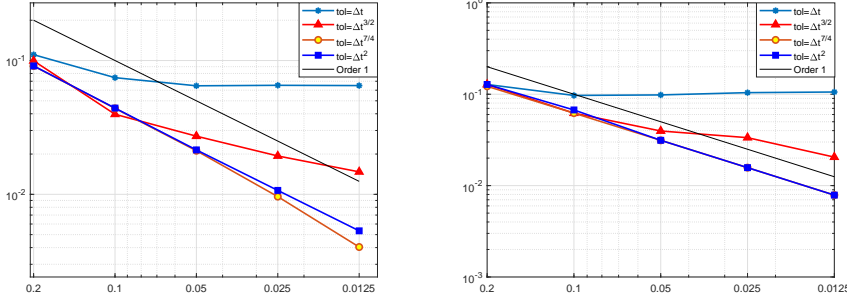


FIG. 6. Test 1: Comparison of the error $Err_{2,2}$ (left) and the error $Err_{\infty,2}$ (right) for the pruned TSA with different tolerances $\varepsilon_{\mathcal{T}}$ as a function of Δt

we would like to mention that similar convergence results have been achieved even for other initial conditions x .

5.2. Test 2: Van der Pol oscillator. In the second test case we consider the Van der Pol oscillator. The dynamics in (2.1) is given by

$$(5.5) \quad f(x, u) = \begin{pmatrix} x_2 \\ \omega(1 - x_1^2)x_2 - x_1 + u \end{pmatrix} \quad u \in U \equiv [-1, 1].$$

We note that the origin is a repulsive point for the uncontrolled dynamics in (5.5), e.g. $u = 0$, if $\omega \in (0, 2]$. For this example we consider $\omega = 0.15$ in (5.5). It is well-known that Van der Pol oscillator is characterized by its cycle limit as shown in Figure 7 with two different initial conditions.

In this example we want to minimize the following cost functional:

$$(5.6) \quad J_{x,t}(u) = \int_t^T (\delta_1 \|y(s)\|_2^2 + \gamma |u(s)|^2) ds + \delta_2 \|y(T)\|_2^2,$$

where $\delta_1, \delta_2, \gamma$ are positive constants.

Case 1. We consider the minimization of the terminal cost in (5.6), e.g. $\delta_1 = \gamma = 0$ and $\delta_2 = 1$. Let us consider $x = (-1, 1)$, $\Delta t = 0.05$ and $T = 1$. The error is computed with respect to the classical approach with a fine grid ($\Delta t = \Delta x = 0.002$).

We will consider Euler scheme with $U = \{-1, 1\}$ and the tolerance is set equal to $\varepsilon_{\mathcal{T}} = \Delta t^2$ with $|\mathcal{T}| = 37030$. In Figure 8 we compare the contour lines of the value function computed by the classical approach with a fine grid and the TSA. We note the approximations show the same behaviour. Furthermore, we mention that the contour line of the value functions are obtained by using MATLAB function

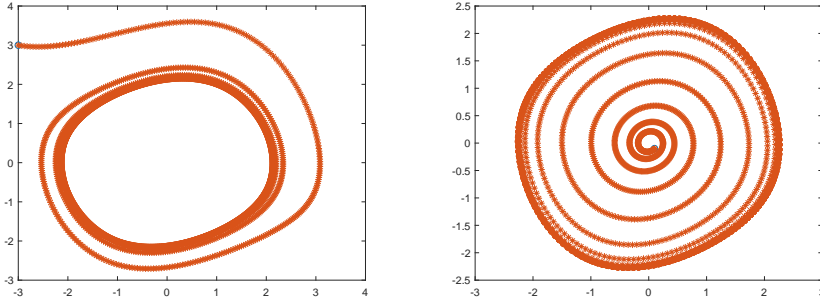


FIG. 7. Test 2: Cycle limit for Van der Pol oscillator with initial point $(-3,3)$ (left) and with initial point $(0.1,-0.1)$ (right)

tricontour, based on a Delaunay's triangulation of the scattered data. We remark that we can compute the value function $V^n(\zeta)$ for $\zeta \in \cup_{k=0}^n \mathcal{T}^k$ since the dynamics is autonomous.

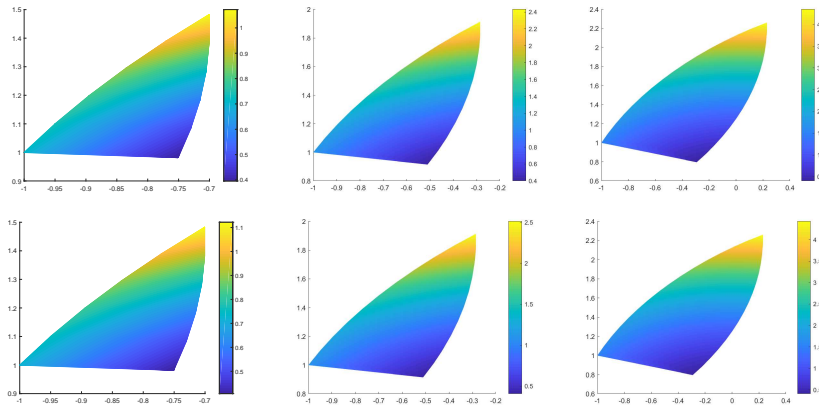


FIG. 8. Test 2: Value function with the classical approach (top) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right). Value function with the TSA (bottom) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right)

The quality of the numerical approximation is confirmed by the error shown in Figure 9. As we can see, pruning the nodes does not influence the error. For each time step the error is below to 0.05 which leads to an accurate approximation of the value function.

Case 2. We consider the minimization of the cost functional in (5.6) with $\delta_1 = \delta_2 = 1$ and $\gamma = 0.01$. Furthermore we set the same initial condition, discretization step and tolerance as in the previous case. The contour lines of the value function are shown in Figure 10.

We note that the results are very similar to the previous case. Our approach is robust with respect to different cost functionals and initial conditions. The right panel of Figure 9 shows the error for each time step considering the tree algorithm with and without nodal selection.

Case 3. In the last case we deal with a two dimensional control space, considering

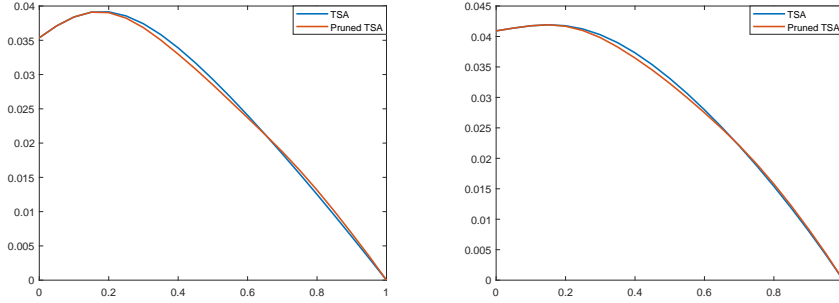


FIG. 9. *Test 2: Error in time with TSA without pruning and with pruning with tolerance $\varepsilon_{\mathcal{T}} = \Delta t^2$ for Case 1 (left) and Case 2 (right) with respect to a value function computed with the classical approach with a very fine grid.*

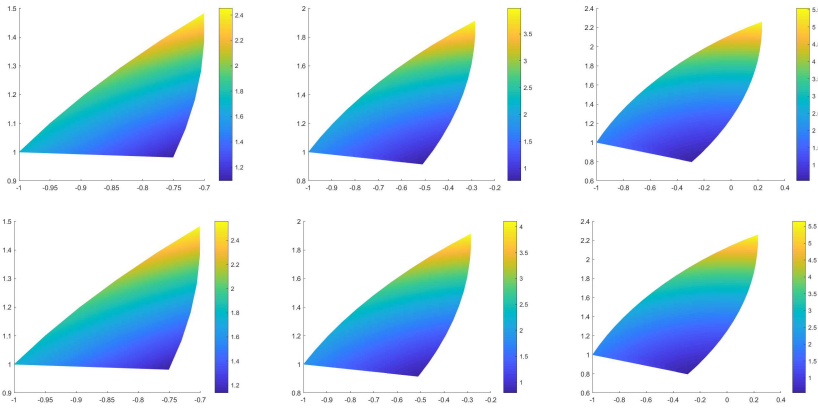


FIG. 10. *Test 2: Value function with the classical approach (top) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right). Value function with the TSA (bottom) on tree nodes at time $t = 0.25$ (left), $t = 0.5$ (middle) and $t = 0.75$ (right)*

the parameter ω in (5.5) as a control, e.g. $\omega \in U$. Therefore, we consider as control variables $(\omega, u) \in U \times U$ in (5.5). In the cost functional (5.6) we consider again $\delta_1 = \gamma = 0.1$ and $\delta_2 = 1$, with $x = (-0.5, 0.5)$, $\Delta t = 0.05$ and $T = 1$. We consider two different choices for the control set: $U = [-2, 0]$ and $U = [-1, 1]$. The control set is discretized with step-size $\Delta u = 0.2$, obtaining altogether 100 discrete controls for both examples. In Figure 11 we show the results in both situations. We can observe that the tree has a different shape due to the different control space. Here, we have set the pruning criteria with $\varepsilon_{\mathcal{T}} = \Delta t^2$. Finally, we note that in both situations we are able to steer the solution to the origin.

5.3. Test 3: Damped harmonic oscillator with sinusoidal driving force.

In this third example we consider a non-autonomous dynamical system: a damped oscillator driven by a sinusoidal external force. The dynamics in (2.1) is given by

$$(5.7) \quad f(x, u, t) = \begin{pmatrix} x_2 \\ -\omega x_2 - \omega^2 x_1 + \sin(\omega t) + u \end{pmatrix} \quad u \in U \equiv [-1, 1].$$

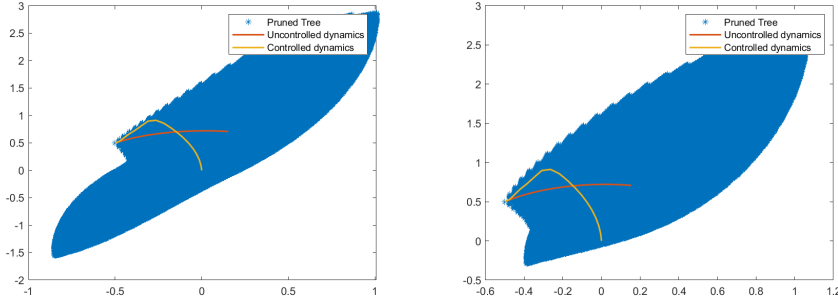


FIG. 11. *Test 2: Pruned tree with the uncontrolled and controlled dynamics with $U = [-2, 0]$ (left) and with $U = [-1, 1]$ (right)*

for $x = (x_1, x_2) \in \mathbb{R}^2$. In this example, we aim to show that our approach works also with non-autonomous dynamics. In this case we can not compute the value function $V^n(\zeta)$ on the sub-tree $\cup_{k=0}^n \mathcal{T}^k$, but only at the n -th time level \mathcal{T}^n and we will apply the pruning rule (4.1). The uncontrolled dynamics (e.g. $u = 0$) converges asymptotically to the cycle limit:

$$\bar{x}_1(t) = \frac{1}{\omega^2} \sin(\omega t + \pi/2), \quad \bar{x}_2(t) = \frac{1}{\omega} \cos(\omega t + \pi/2).$$

We used the same cost functional of the previous case with $\delta_1 = \gamma = 0.1$, $\delta_2 = 1$. The parameters are set as follows: $\omega = \pi/2$, $x = (-0.5, 0.5)$, $U = \{-1, 0, 1\}$, $\Delta t = 0.05$, $T = 1$, $\varepsilon_{\mathcal{T}} = \Delta t^2$. The cardinality of tree in this case is 32468. In the left

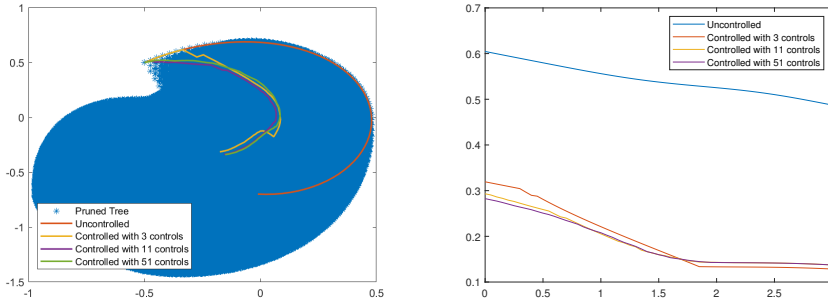


FIG. 12. *Test 3: Pruned tree with the uncontrolled and controlled dynamics (left) and comparison of the cost functional on time varying the number of discrete controls (right)*

panel of Figure 12 we show the tree nodes and the optimal trajectory computed with Algorithm 1 and the uncontrolled solution. To show the quality of the controlled solution we evaluate the cost functional for each time step as shown in the right panel of Figure 12. As expected the controlled trajectory is always below the uncontrolled one. In order to further show the effectiveness of the pruning criteria we have increased the number of controls up to $M = 51$ and the horizon up to $T = 3$. Again, this would not be possible without a pruning criteria due to the dimension of the tree.

5.4. Test 4: Heat equation. The fourth example concerns the control of a PDE. In the first three examples we showed the accuracy of our method with respect

to existing methods for low-dimensional problems. In what follows we would like to give an idea of how the proposed method can work in higher dimension.

We want to study the following heat equation:

$$(5.8) \quad \begin{cases} y_t = \sigma y_{xx} + y_0(x)u(t) & (x, t) \in \Omega \times [0, T], \\ y(x, t) = 0 & (x, t) \in \partial\Omega \times [0, T], \\ y(x, 0) = y_0(x) & x \in \Omega, \end{cases}$$

where the state lies in an infinite-dimensional Hilbert space (see e.g. [11]). Here, we consider the term $y_0(x)u(t)$ to provide a spatial dependence to the control input. This is a particular choice, but the algorithm has no restrictions on more general shape functions. To write equation (5.8) in the form (2.1) we use the centered finite difference method which leads to the following ODEs system

$$(5.9) \quad \dot{y}(t) = Ay(t) + Bu(t),$$

where the matrix $A \in \mathbb{R}^{d \times d}$ is the so called *stiffness* matrix whereas the vector $B \in \mathbb{R}^n$ is given by $(B)_i = y_0(x_i)$ for $i = 1, \dots, n$ and x_i is the spatial grid with constant step size Δx . The cost functional we want to minimize reads:

$$J_{y_0,t}(u) = \int_t^T (\delta_1 \|y(s)\|_2^2 dx + \gamma |u(s)|^2) ds + \|y(T)\|_2^2,$$

where $y(t)$ is the solution of (5.9), $u(t)$ is taken in the admissible set of controls $\mathcal{U} = \{u : [0, T] \rightarrow [-1, 1]\}$ and $\Omega = [0, 1]$. We set $\delta_1 = 1$ and $\gamma = 0.01$.

Smooth initial condition. In the numerical approximation of (5.8) we consider $y_0(x) = -x^2 + x$, $\Delta x = 10^{-3}$, $\Delta t = 0.05$, $T = 1$ and $\sigma = 0.1$. The dimension of the problem is $d = 1000$. We use an implicit Euler scheme to integrate the system (5.9) and guarantee its stability. We note that the use of a one step implicit is straightforward even if we have introduced an explicit scheme in the previous sections. We refer to [29] for more details about the method.

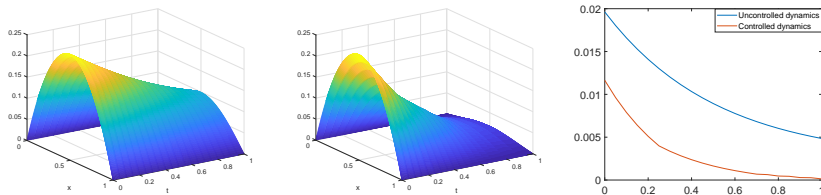


FIG. 13. Test 4 (smooth initial condition): Uncontrolled solution (left), optimal control solution (middle), time comparison of the cost functional of the uncontrolled solution and controlled solution (right).

The solution of the uncontrolled problem (5.8) with $u(t) \equiv 0$ is shown in the left panel of Figure 13. In the middle we show the solution of the controlled problem where the value function is computed with Algorithm 1 and the control is computed as explained in (2.6). We note that feedback control was computed with the discrete control set $U = \{-1, 0, 1\}$ as for the value function. A refinement for the control set would require further investigation that we will address in the near future. However, it is extremely interesting to show that we are able to compute the value function for (5.8) in dimension 1000. This approach might substitute recent advances where the

feedback for PDEs was computed by coupling the HJB equation with model order reduction techniques such as, e.g., Proper Orthogonal Decomposition [24]. Finally in the right panel of Figure 13 we show the time behaviour of the cost functional for the uncontrolled and the controlled solution. As expected, the cost functional of the latter is lower.

Non-smooth initial condition. In this example we consider the following non-smooth initial $y_0(x) = \chi_{[0.25,0.75]}(x)$, where $\chi_\omega(x)$ is the characteristic function in the domain ω , whereas the other parameters are set as in the previous case.

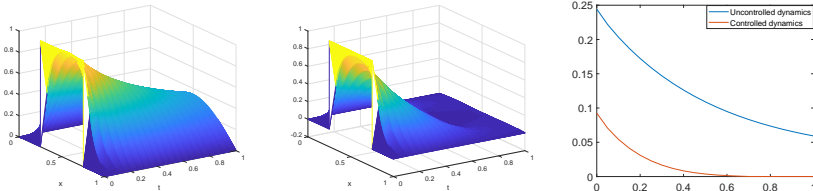


FIG. 14. *Test 4 (non-smooth initial condition): Uncontrolled solution (left), optimal control solution (middle), time comparison of the cost functional of the uncontrolled solution and controlled solution (right).*

As one can see from Figure 14, we are able to approximate the control problem even if the initial condition is non-smooth. We note that, although the simple diffusive properties of the problem, a model reduction approach will not be able to reconstruct such initial condition with a few number of basis functions. Therefore it will not be possible to solve this problem with a classical approach. This again shows the effectiveness of the method.

5.5. Test 5: Wave equation. In this last example we consider a hyperbolic PDE, the wave equation which reads:

$$(5.10) \quad \begin{cases} w_{tt} = c w_{xx} + \chi_\omega(x)u(t) & (x, t) \in \Omega \times [0, T], \\ w(x, t) = 0 & (x, t) \in \partial\Omega \times [0, T], \\ w(x, 0) = w_0(x), w_t(x, 0) = w_1(x) & x \in \Omega. \end{cases}$$

where ω is a subset of Ω . For all initial data $(w_0, w_1) \in H_0^1(\Omega) \times L^2(\Omega)$ and every $u(t) \in L^2(0, T)$, there exists a unique solution $w \in C^0(0, T; H_0^1(\Omega)) \cap C^1(0, T; L^2(\Omega)) \cap C^2(0, T; H^{-1}(\Omega))$ of the Cauchy problem (5.10). We refer to [11] for more details about this equation. We can rewrite the wave equation in the following compact form

$$\dot{y}(t) = Ay(t) + Bu(t),$$

defining

$$(5.11) \quad y(t) = \begin{pmatrix} w(t) \\ w_t(t) \end{pmatrix}, \quad A = \begin{pmatrix} 0 & I \\ c \partial_x^2 & 0 \end{pmatrix}, \quad Bu(t) = \begin{pmatrix} 0 \\ \chi_\omega(x)u(t) \end{pmatrix}.$$

Again we apply an implicit Euler scheme to avoid narrow CFL conditions. We want to minimize the following cost functional

$$J_{y_0, t}(u) = \int_0^T (\varphi(\|y(s)\|_2^2) + \gamma|u(s)|^2) ds + \varphi(\|y(T)\|_2^2),$$

with $w_0(x) = \sin(\pi x)$, $w_1(x) = 0$, $\gamma = 0.01$, $T = 1$, $c = 0.5$, $\Omega = (0, 1)$ and $\omega = (0.4, 0.6)$, $\Delta x = 10^{-3}$, $\Delta t = 0.05$. We note that the dimension of the semi-discrete problem is $d = 2000$.

Quadratic cost functional. We first consider a standard tracking problem e.g. $\varphi(x) = x$ in the cost functional. In Figure 15 we show the uncontrolled solution in the left panel and the controlled solution in the middle. A comparison of the evaluations of the cost functional is given in the right panel. As expected the controlled solution is below the uncontrolled one for each time instance. This shows the capability of the method for high dimensional problem even for hyperbolic equations.

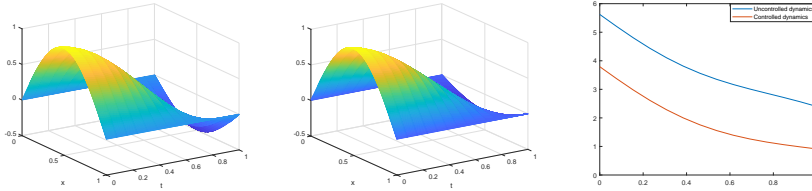


FIG. 15. Test 5: Uncontrolled solution (left), optimal control solution (middle), time comparison of the cost functional of the uncontrolled solution and controlled solution (right).

Non-quadratic cost functional. Now, we consider a more complicated example which deals with a non-quadratic cost functional. Let us consider for example the following cost functional where

$$\varphi(x) = \begin{cases} \sin(\pi|x|) & |x| \leq 0.5, \\ 1 & 0.5 < |x| \leq 1, \\ (|x| - 1)^2 + 1 & |x| > 1, \end{cases}$$

as shown in the left panel of Figure 16. We consider the same parameters as in the previous case, which lead to the same uncontrolled solution as shown in the left panel of Figure 15. In the middle of Figure 16 one can see the uncontrolled solution and

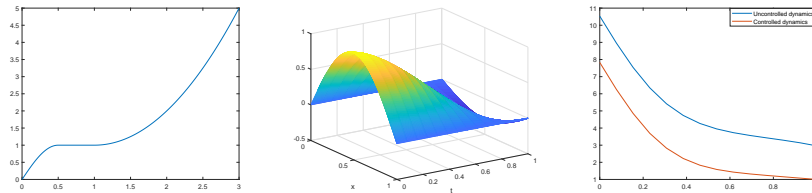


FIG. 16. Test 5 (Non-quadratic cost functional): Graphics of $\varphi(x)$ (left), optimal control solution (middle), time comparison of the cost functional of the uncontrolled solution and controlled solution (right).

in the right panel a comparison of the evaluation of the cost functional. Again, here we would like to stress the capability of the method to work with high dimensional problem and with non-smooth cost functionals.

6. Conclusions and future works. We have proposed a novel method to approximate time dependent HJB equations via DP scheme on a tree structure. The proposed algorithm creates the tree structure according to all the possible directions of the controlled dynamical system for a finite set of controls. This procedure has several advantages with respect to the DP algorithm based on the classical time and space discretization. The first advantage is that we do not have to build a space grid

and a local space interpolation. Furthermore, TSA does not require an a-priori choice of a numerical domain Ω to set the numerical scheme and, consequently, there is no need to impose boundary conditions. The construction of the tree is made step-by-step, via the pruning rule. Thus, the complexity of the problem is drastically reduced cutting all the branches laying in a small neighbourhood. After pruning the tree the efficiency of TSA is greatly improved in terms of CPU time. This approach allows to apply the DP method to high-dimensional problems as it has been shown in the numerical section for both ODEs and PDEs, in some test problems we solved an optimal control problem in dimension 2000.

We note that the method could be easily extended to second order approximation schemes using e.g. Heun method for the dynamics and it is also possible to reduce the CPU time via a parallel version of the method. Although the numerical results are promising, some issues are still open in the analysis of the method. The first is to derive error estimates in agreement with the order of convergence shown in Table 1. Furthermore, we would like to extend the method to the control of nonlinear PDEs coupling TSA with model order reduction methods as discussed in [3] and taking advantage of the theoretical results found there. These extensions could open the way to the application of DP techniques for real industrial problems.

REFERENCES

- [1] E. G. Al'brekht. *On the optimal stabilization of nonlinear systems*, J. Appl. Math. Mech., **25**, 1961, 254-266.
- [2] A. Alla, M. Falcone, and D. Kalise. *An efficient policy iteration algorithm for dynamic programming equations*, SIAM J. Sci. Comput., **37**, 2015, 181-200.
- [3] A. Alla, M. Falcone, S. Volkwein, *Error analysis for POD approximations of infinite horizon problems via the dynamic programming approach*, SIAM J. Control Optim. **55**, 5, 3091-3115
- [4] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhäuser, Basel, 1997.
- [5] R. Bellman, *Dynamic Programming*. Princeton university press, Princeton, NJ, 1957.
- [6] V.G. Boltyanskii, R.V. Gamkrelidze. L.S. Pontryagin, *Towards a theory of optimal processes*, (Russian), Reports Acad. Sci. USSR, vol.110(1), 1956.
- [7] R. P. Brent, *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.
- [8] S. Cacace, E. Cristiani. M. Falcone, and A. Picarelli. *A patchy dynamic programming scheme for a class of Hamilton-Jacobi-Bellman equations*, SIAM Journal on Scientific Computing, **34**, 2012, A2625-A2649.
- [9] E. Carlini, M. Falcone and R. Ferretti, *An efficient algorithm for Hamilton-Jacobi equations in high dimension*, Comput. Vis. Sc., **7**, (2004), 15-29.
- [10] F. Camilli, M. Falcone, P. Lanucara, and A. Seghini. *A domain decomposition method for Bellman equations*, in D. E. Keyes and J. Xu (eds.), *Domain Decomposition methods in Scientific and Engineering Computing*, Contemporary Mathematics n.180, AMS, 1994, 477-483.
- [11] C.L. Evans. *Partial Differential Equations*. American Mathematical Society, 2010.
- [12] M. Falcone and R. Ferretti. *Semi-Lagrangian Approximation Schemes for Linear and Hamilton-Jacobi equations*, SIAM, 2013.
- [13] M. Falcone, T. Giorgi, *An approximation scheme for evolutive Hamilton-Jacobi equations*, in W.M. McEneaney, G. Yin and Q. Zhang (eds.), "Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming", Birkhäuser, 1999, 289-303.
- [14] M. Falcone, P. Lanucara, and A. Seghini. *A splitting algorithm for Hamilton-Jacobi-Bellman equations* Applied Numerical Mathematics, **15**, 1994, 207-218.
- [15] A. Festa. *Reconstruction of independent sub-domains for a class of Hamilton-Jacobi equations and application to parallel computing*, ESAIM:M2AN, **4**, 2016, 1223-1240.
- [16] A. Festa, *Domain decomposition based parallel Howard's algorithm*, Math. Comput. Simulation, **147**, 2018, 121-139.
- [17] W.H. Fleming, H.M. Soner, *Controlled Markov processes and viscosity solutions*, Springer-

- Verlag, New York, 1993.
- [18] J. Garcke and A. Kröner. *Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids*, Journal of Scientific Computing, **70**, 2017, 1-28.
 - [19] R.A. Howard. Dynamic programming and Markov processes. Wiley, New York, 1960.
 - [20] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. Optimization with PDE Constraints. Mathematical Modelling: Theory and Applications, **23**, Springer Verlag, 2009.
 - [21] I.T. Jolliffe, Principal component analysis, Springer Series in Statistics, 2nd ed., Springer, 2002.
 - [22] D. Kalise, A. Kroener and K. Kunisch, *Local minimization algorithms for dynamic programming equations*, SIAM Journal on Scientific Computing, **38**, 2016, A1587 - A1615.
 - [23] D. Kalise, K. Kunisch, *Polynomial approximation of high-dimensional Hamilton-Jacobi-Bellman equations and applications to feedback control of semilinear parabolic PDEs*, SIAM Journal on Scientific Computing, **40**, 2018, A629-A652.
 - [24] K. Kunisch, S. Volkwein, and L. Xie. *HJB-POD based feedback design for the optimal control of evolution problems*, SIAM J. on Applied Dynamical Systems, **4**, 2004, 701-722.
 - [25] C. Navasca and A.J. Krener. *Patchy solutions of Hamilton-Jacobi-Bellman partial differential equations*, in A. Chiuso et al. (eds.), Modeling, Estimation and Control, Lecture Notes in Control and Information Sciences, **364**, 2007, 251-270.
 - [26] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2003.
 - [27] K. Pearson, *On Lines and Planes of Closest Fit to Systems of Points in Space*, Philosophical Magazine, **2**, 1901, 559-572.
 - [28] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, E.F. Mishchenko, The Mathematical Theory of Optimal Processes (Russian), English translation: Interscience 1962.
 - [29] A. Quarteroni and A. Valli. Domain Decomposition Methods for Partial Differential Equations, Oxford University Press, 1999.
 - [30] L. Saluzzi, A. Alla and M. Falcone. *Error estimates for a tree structure algorithm for dynamic programming equations*, submitted, 2018, <https://arxiv.org/abs/1812.11194>.
 - [31] J. A. Sethian. *Level set methods and fast marching methods*, Cambridge University Press, 1999.
 - [32] F. Tröltzsch. Optimal Control of Partial Differential Equations: Theory, Methods and Application, American Mathematical Society, 2010.
 - [33] S. Volkwein. *Model Reduction using Proper Orthogonal Decomposition*, Lecture Notes, University of Konstanz, 2013.