Automatica 50 (2014) 2234–2244

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

A sparse collocation method for solving time-dependent HJB equations using multivariate B-splines☆

Nithin Govindarajan [a], Cornelis. C. de Visser [b,1], Kalmanje Krishnakumar [c]

[a] Delft Center for Systems and Control, Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands

[b] Control and Simulation division, Faculty of Aerospace Engineering, Delft University of Technology, 2600GB Delft, The Netherlands

[c] Intelligent Systems Division, NASA Ames Research center, Moffett Field, CA 94035, USA

ARTICLE INFO

Article history:
Received 13 May 2013
Received in revised form
27 February 2014
Accepted 30 April 2014
Available online 20 August 2014

Keywords:
Optimal feedback control
Hamilton–Jacobi–Bellman equation
Adaptive dynamic programming
Splines
Collocation method

ABSTRACT

This paper presents a sparse collocation method for solving the time-dependent Hamilton–Jacobi–Bellman (HJB) equation associated with the continuous-time optimal control problem on a fixed, finite time-horizon with integral cost functional. Through casting the problem in a recursive framework using the value-iteration procedure, the value functions of every iteration step is approximated with a time-varying multivariate simplex B-spline on a certain state domain of interest. In the collocation scheme, the time-dependent coefficients of the spline function are further approximated with ordinary univariate B-splines to yield a discretization for the value function fully in terms of piece-wise polynomials. The B-spline coefficients are determined by solving a sequence of highly sparse quadratic programming problems. The proposed algorithm is demonstrated on a pair of benchmark example problems. Simulation results indicate that the method can yield increasingly more accurate approximations of the value function by refinement of the triangulation.

1. Introduction

In the design of modern control systems for applications in aerospace, process, and automotive industry, ensuring optimality with respect to some specified cost index is highly desired. Optimal control theory is the field that deals with the problem of finding optimal controls for dynamical systems in either open-loop, or more preferably, closed-loop (i.e. feedback) form.

Within optimal control, the dynamic programming method is widely regarded as the most comprehensive approach in finding optimal feedback controllers for generic nonlinear systems. Over the years, the subject has evolved into a broad field. However, a limiting factor for the widespread application of dynamic programming algorithms today is the inherent computational intractability of the method. This curse of dimensionality, as it was coined by Bellman (1957), has forced researchers to explore techniques that can overcome the computational drawbacks. The common approach is to approximate the value function and optimal feedback control with generic function approximators such as Neural Networks (NNs). In the literature, such approaches have come under a variety of names: Adaptive Dynamic programming (Wang, Zhang, & Liu, 2009), Approximate Dynamic Programming (ADP) (Powell, 2007), Neuro-Dynamic Programming (Bertsekas & Tsitsiklis, 1996), and reinforcement learning (Sutton & Barto, 1998). Nevertheless, there are several variations to the dynamic programming problem, which broadly can be classified according to the type of dynamics (i.e. discrete vs. continuous time, linear vs. nonlinear systems, deterministic vs. stochastic systems) and optimization criteria (i.e. finite vs. infinite horizon, quadratic vs. general nonlinear cost functionals) the problems address. Much of the literature has been mainly devoted to the discrete-time case with often infinite horizon cost functions. Work in the continuous-time domain remains fairly limited to a few publications (Abu-Khalaf & Lewis, 2005; Beard, 1995; Cheng, Lewis, & Abu-Khalaf, 2007; Hanselmann, Noakes, & Zaknich, 2007; Vamvoudakis & Lewis, 2009, 2010). Central in dynamic programming is solving the Bellman equation of optimality. In the continuous-time setting, this equation comes in the form of a first-order hyperbolic PDE known as the Hamilton–Jacobi–Bellman (HJB) equation.

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Kok Lay Teo under the direction of Editor Ian R. Petersen.

E-mail addresses: N.Govindarajan@student.tudelft.nl (N. Govindarajan), C.C.deVisser@tudelft.nl (C.C. de Visser), k.krishnakumar@nasa.gov (K. Krishnakumar).

[1] Tel.: +31 527851246; fax: +31 527851246.

http://dx.doi.org/10.1016/j.automatica.2014.07.012
0005-1098/

This paper focuses on the optimal control problem for a nonlinear, continuous-time, deterministic system with cost criteria defined on a finite-horizon, also known generically as the *Bolza problem* (Bardi & Capuzzo-Dolcetta, 2008). In contrast to the infinite-horizon, the class of problems considered here give rise to a time-dependent value function of which the associated HJB equation contains an additional partial derivative term with respect to the time variable. The HJB equation can be solved using the upwind finite volume method in Wang, Jennings, and Teo (2003). In Huang, Wang, Chen, and Li (2006), a collocation method was developed to solve the same HJB equation using Radial Basis Functions (RBFs). In Alwardi, Wang, Jennings, and Richardson (2012), this method was extended with an adaptive scheme that refines the distribution of the RBF centers by means of feeding back the approximation error. In parallel, Cheng et al. (2007) presented a nearly identical scheme for a special case of the problem in Huang et al. (2006) where the system is input affine, the cost functional is positive definite on the state and input, and the input space is unbounded. In the collocation method, the aim is to determine appropriate weights for a given set of trial functions, such that the PDE is satisfied exactly at a certain carefully selected points spanning the computational domain. As opposed to finite difference methods which approximate the solution of a PDE only at a discrete set of points, the collocation method finds an approximation for the entire computational domain.

Regardless, a limitation of previous work is that only the state-dependent part of the value function is approximated with a collocation-like procedure. The temporal variable on the other hand, is discretized in the ordinary fashion using finite difference approximations, resulting in a scheme which is only semi-meshless in nature. In the interest of designing optimal feedback controllers, a full state and time parametrization of the value function in terms of smooth functions is desirable. In such a parametrization, the gradient of value function can be evaluated analytically for any time instance, and if the argument that optimizes the Hamiltonian can be expressed explicitly in terms of this gradient, closed-form expressions can be obtained for the (time-dependent) optimal feedback control-law.

There are two major complications that accompany a full state and time discretization of the time-dependent HJB equation in terms of smooth functions: (i) selection of an effective approximation structure which complies with the underlying physics of the PDE and handles the PDE boundary conditions effectively, (ii) computing values for the approximation coefficients by solving the resulting nonlinear optimization problem, which typically is also non-smooth if the optimal feedback control-law consist of a switching structure. In this paper, we show that by: (i) introducing a specific hybrid between a tensor and simplex *B*-spline, (ii) applying a value-iteration like procedure, we can effectively deal with both of these complications, respectively.

The contribution of this paper is a new collocation method that fully discretizes both the state and time components of the HJB equation in terms of smooth functions. Given the evolutionary nature of the underlying PDE, we present a scheme that approximates the value function with a time-varying simplex *B*-spline. By successively approximating the time-dependent *B*-coefficients with univariate *B*-spline functions, we end up with a discretization for the value function in terms of piecewise polynomials. By means of employing a value-iteration procedure, the unknown *B*-spline coefficients are determined by solving a sequence of least-squares problems with equality constraints. The validity of using this procedure in the overall approximation scheme is supported with a theorem that states convergence for the corresponding theoretical procedure. The entire method is illustrated on two benchmark examples. Simulation results indicate that high accuracy approximations of the value function can be obtained for relatively simple spline function configurations.

The proposed method makes explicit use of the multivariate simplex *B*-spline as a function approximator. The simplex *B*-spline can be considered as the natural generalization of the univariate *B*-spline to the multivariate case (de Boor, 1987; Lai & Schumaker, 2007). From a numerical standpoint, we establish a sound argument for using this specific spline function in the proposed approximation scheme. We show by means of a complexity analysis that the *B*-spline method has favorable properties in terms of both space and time complexity. These favorable properties arise from domain decomposition and the resulting sparse structure in the quadratic program. Domain decomposition techniques were used in the RBF based method of Alwardi, Wang, and Jennings (2013) to reduce complexity of the problem. It suffices that the use of *B*-splines naturally decomposes the domain of the specified problem, where every sub-problem associated with every sub-domain is combined into one consistent solution by means of linear equality constraints on the *B*-coefficients. Furthermore, the inclusion of some trivial additional equality constraints allows us to effectively deal with the PDE boundary conditions, which represents another choice for the *B*-spline based architecture. The multivariate simplex *B*-spline presented in the form given in this paper has been used in the past to find numerical solutions for elliptic PDEs (Awanou & Lai, 2004), and also specific cases of the Navier–Stokes equations (Awanou, Lai, & Wenston, 2005; Hu, Han, & Lai, 2007). Yet, these are all problems which are static in nature, and therefore do not involve an evolving time parameter. In any case, to the best of the authors knowledge, the versatility of simplex *B*-splines in solving HJB equations has never been formally explored in the literature, and hence is considered a novelty by itself.

The paper is organized as follows. Section 2 provides background on the optimal feedback control problem, and reflects on several aspects concerning the HJB equation, and value-iteration. Section 3 introduces readers to the multivariate simplex *B*-spline. Section 4 presents the main contributions of this paper in which the details of the collocation method are discussed. Section 5 discusses the computational complexity of the *B*-spline method. Section 6 presented numerical results obtained for two benchmark example problems. Section 7 states the conclusions of the paper.

2. Problem statement and background on optimal control

This section discusses the details of the optimal control problem. We introduce here also the value-iteration procedure which will be used in the overall scheme to approximate the solution of the HJB equation.

2.1. Problem formulation

Consider a generic nonlinear dynamical system

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

with $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathrm{U} \subseteq \mathbb{R}^m$, and $\boldsymbol{f} : \mathbb{R}^n \times \mathrm{U} \mapsto \mathbb{R}^n$ a Lipschitz continuous function. Let $\boldsymbol{\psi}(\tau; \boldsymbol{x}, t, \boldsymbol{u}(\cdot))$ denote the unique state trajectory of the system (1) at the time interval $\tau \in [t, T]$ for a given initial state $\boldsymbol{x}$ at initial time t, and a control signal $\boldsymbol{u}(\cdot) \in \mathcal{U}_{[t,T]}$, where

$$\mathcal{U}_{[t,T]} := \{\boldsymbol{u}(\cdot) : [t, T] \mapsto \mathrm{U} \mid \boldsymbol{u}(\cdot) \text{ is measurable}\}.$$

The objective in the finite-horizon optimal control problem is to find a control $\boldsymbol{u}(\cdot) \in \mathcal{U}_{[t,T]}$ that results in a system trajectory $\boldsymbol{\psi}(\cdot) : [t, T] \mapsto \mathbb{R}^n$ which minimizes a specific cost functional over a fixed, finite horizon. The goal is to find

$$V(t, \boldsymbol{x}) := \inf_{\boldsymbol{u}(\cdot) \in \mathcal{U}_{[t,T]}} \int_t^T r[\boldsymbol{\psi}(\tau; \boldsymbol{x}, t, \boldsymbol{u}(\cdot)), \boldsymbol{u}(\tau)] \, \mathrm{d}\tau + l[\boldsymbol{\psi}(T; \boldsymbol{x}, t, \boldsymbol{u}(\cdot))] \tag{2}$$

where $r : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ and $l : \mathbb{R}^n \mapsto \mathbb{R}$ are continuous functions that denote respectively the *running* and *terminal* cost. The function $V : [0, T] \times \mathbb{R}^n \mapsto \mathbb{R}$ in (2) is known in the literature as the value function, and it plays an essential role in the *synthesis* of optimal feedback controllers.

2.2. The Hamilton–Jacobi–Bellman equation

The value function (2) has an optimal substructure as it satisfies Bellman's principle of optimality (Bardi & Capuzzo-Dolcetta, 2008). In infinitesimal form, this optimality condition is expressed in the form of a PDE

$$\frac{\partial V(t,\boldsymbol{x})}{\partial t} + \inf_{\boldsymbol{u}\in \mathrm{U}} \mathcal{H}\left(\boldsymbol{x}, \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{u}\right) = 0 \tag{3}$$

with $V(T, \boldsymbol{x}) = l(\boldsymbol{x})$ as the boundary condition, and $\mathcal{H} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ the Hamiltonian defined by

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{u}) := \langle \boldsymbol{p}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})\rangle + r(\boldsymbol{x}, \boldsymbol{u}) \tag{4}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product. The first-order hyperbolic PDE in (3) is termed the Hamilton–Jacobi–Bellman (HJB) equation and is essentially nonlinear because of the optimization of the Hamiltonian (4). Classical solutions rarely exist for this class of PDEs. In Bardi and Capuzzo-Dolcetta (2008) it has been shown that, although (2) is continuous, it may not be differentiable everywhere. This holds true even for simple cases where (1) is linear, and r, l are smooth.[2] Consequently, for the general case, (2) can satisfy (3) only in an appropriate weak sense. It has been shown by Bardi and Capuzzo-Dolcetta (2008) that this weak solution comes in the form of viscosity solutions. The viscosity solution is unique to (3) and satisfies the PDE in the classical sense wherever (2) is differentiable. On the other hand, wherever (2) is not differentiable, it suffices that the super- and sub-differentials of the viscosity solution meet certain inequalities at that point (Crandall, Evans, & Lions, 1984).

In the synthesis of optimal feedback controllers, wherein optimality is encoded in terms of r and l, solving (3) is the main issue. This fact is brought to light by rewriting (3) into the form

$$\frac{\partial V(t,\boldsymbol{x})}{\partial t} + \mathcal{H}\left(\boldsymbol{x}, \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{g}^*(t,\boldsymbol{x})\right) = 0 \tag{5}$$

where $\boldsymbol{g}^* : [0, T] \times \mathbb{R}^n \mapsto \mathrm{U}$ is a time-dependent feedback control that optimizes the Hamiltonian

$$\boldsymbol{g}^*(t,\boldsymbol{x}) := \arg\inf_{\boldsymbol{u}\in \mathrm{U}} \mathcal{H}\left(\boldsymbol{x}, \frac{\partial V(t,\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{u}\right). \tag{6}$$

The function $\boldsymbol{g}^*$ can be seen as an optimal feedback control-law for the system (1). Note that this control-law may not be unique, as there might exist multiple controllers that optimize (4). A trivial example of an optimal control problem with non-unique optimal feedback controllers is the case where: $f(x, u) = -x + u$ with $u \in [-1, 1]$, $r(x, u) = 0$, and $l(x) = -x^2$. Indeed, it can be verified that optimal control at $x = 0$ is multivalued, i.e. $g^*(t, 0) \in \{1, -1\}$.

For most problems, it is very difficult to construct an analytic solution for (5). In most cases, one must resort to numerical techniques that approximate the solution, such as the finite difference methods that go under the name of level set methods (Osher & Fedkiw, 2003). Furthermore, note that when (1) is linear, r is quadratic with respect to $\boldsymbol{x}$ and $\boldsymbol{u}$, $l(\boldsymbol{x}) = 0$, and $\mathrm{U} = \mathbb{R}^m$, (3) can be reduced to the well-known Riccati differential equation of the Linear Quadratic Regulator (LQR) problem.

[2] See also example I in Section 6.1 for a trivial optimal control problem with non-smooth solution.

2.3. Value-iteration

The task of finding an optimal feedback law (6) and solving the terminal value problem (3) are two interrelated problems. The interdependency between these two problems can be decoupled by introducing a successive approximation algorithm.

Procedure 1 (*Value-Iteration*). *Initialize* $V^{(0)}(t, \boldsymbol{x}) = l(\boldsymbol{x})$. *Perform the following iteration:*

1. *Define*

$$\boldsymbol{g}^{(i)}(t,\boldsymbol{x}) := \arg\inf_{\boldsymbol{u}\in \mathrm{U}} \mathcal{H}\left(\boldsymbol{x}, \frac{\partial V^{(i-1)}(t,\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{u}\right). \tag{7}$$

2. *Find* $V^{(i)}$, *which is the solution of*

$$\frac{\partial V^{(i)}(t,\boldsymbol{x})}{\partial t} + \mathcal{H}\left(\boldsymbol{x}, \frac{\partial V^{(i)}(t,\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{g}^{(i)}(t,\boldsymbol{x})\right) = 0 \tag{8}$$

with $V^{(i)}(T, \boldsymbol{x}) = l(\boldsymbol{x})$.

In the form presented, the above recursion essentially describes a value-iteration procedure for the finite-horizon, continuous-time and continuous-state optimal control problem. At every iteration cycle, an updated value function is obtained by (8), which is then used to define a new feedback control-law (7) for the system. The expectation is that the recursive updating of the value function will converge to (2). In Saridis and Lee (1979) an extensive proof was developed for the convergence of Procedure 1 when: $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f_1}(\boldsymbol{x}) + B\boldsymbol{u}$, $r(\boldsymbol{x}, \boldsymbol{u}) = L(\boldsymbol{x}) + \|\boldsymbol{u}\|^2$ with $L(\boldsymbol{x}) \geq 0$, and $l(\boldsymbol{x}) \geq 0$. Although not proven explicitly in the paper, Saridis and Lee (1979) stated that the convergence results are generalizable for the class of problems where: $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ and $r(\boldsymbol{x}, \boldsymbol{u})$ form a convex set for $\boldsymbol{u} \in \mathrm{U}$, and $l(\boldsymbol{x})$ is nonexistent. We will show in the following theorem that, under certain conditions, we can extend this convergence statement for the more general case considered in this paper, wherein the cost functional also contains a terminal cost, i.e. $l(\boldsymbol{x}) \neq 0$.

Theorem 1 (*Convergence Value-Iteration Proc.*). *Consider the optimal control problem defined in Section 2.1. Assume that:*

(i) *The input set* U *is a convex set;*
(ii) *For all* $\boldsymbol{x} \in \mathbb{R}^n$, *the function* $\boldsymbol{f}(\boldsymbol{x}, \cdot) : \mathrm{U} \mapsto \mathbb{R}^n$ *is a convex function, componentwise;*
(iii) *For all* $\boldsymbol{x} \in \mathbb{R}^n$, *the function* $r(\boldsymbol{x}, \cdot) : \mathrm{U} \mapsto \mathbb{R}$ *is a convex function;*
(iv) *The function* $l : \mathbb{R}^n \mapsto \mathbb{R}$ *is continuously differentiable, and the function* $\left\langle \frac{\partial l(\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x}, \cdot)\right\rangle : \mathrm{U} \mapsto \mathbb{R}$ *is convex.*

Then the sequence $\left\{V^{(i)},\ i \in 1, 2, \ldots\right\}$ *defined in Procedure 1 is monotonically decreasing for every* $(t, \boldsymbol{x}) \in [0, T] \times \mathbb{R}^n$, *and satisfies*

$$\lim_{i\to\infty} V^{(i)}(t,\boldsymbol{x}) = V(t,\boldsymbol{x}), \quad \forall (t,\boldsymbol{x}) \in [0, T] \times \mathbb{R}^n.$$

Proof. To prove the theorem, let us introduce the augmented running cost

$$\tilde{r}(\boldsymbol{x}, \boldsymbol{u}) := r(\boldsymbol{x}, \boldsymbol{u}) + \left\langle \frac{\partial l(\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})\right\rangle$$

so that we can define an augmented value function

$$\tilde{V}(t,\boldsymbol{x}) := \inf_{\boldsymbol{u}(\cdot)\in \mathcal{U}_{[t,T]}} \int_t^T \tilde{r}\left[\boldsymbol{\psi}(\tau; \boldsymbol{x}, t, \boldsymbol{u}(\cdot)), \boldsymbol{u}(\tau)\right] \mathrm{d}\tau. \tag{9}$$

By assump. (iv), note that $\tilde{r}(\boldsymbol{x}, \cdot) : \mathrm{U} \mapsto \mathbb{R}$ is a convex function, and hence, according to Saridis and Lee (1979)[3] we can claim for

[3] Refer to Theorem 4 and comments provided at the end of Section 2.

every $(t, \boldsymbol{x}) \in [0, T] \times \mathbb{R}^n$ that

$$\tilde{V}^{(i)}(t, \boldsymbol{x}) \leq \tilde{V}^{(i-1)}(t, \boldsymbol{x}) \tag{10}$$

and

$$\lim_{i \to \infty} \tilde{V}^{(i)}(t, \boldsymbol{x}) = \tilde{V}(t, \boldsymbol{x}). \tag{11}$$

The key idea behind the proof is to show that $\tilde{V}(t, \boldsymbol{x}) = V(t, \boldsymbol{x}) - l(\boldsymbol{x})$, and that $\tilde{V}^{(i)}(t, \boldsymbol{x}) = V^{(i)}(t, \boldsymbol{x}) - l(\boldsymbol{x})$. Then the claims made in the theorem directly follow from substitution of the results into (10) and (11).

We proceed by showing that these relations hold through examining the difference between (2) and (9), i.e. we define $S := V - \tilde{V}$. By the definition of the value function, we have that

$$\begin{aligned} S(t, \boldsymbol{x}) := \inf_{\boldsymbol{u}(\cdot) \in \mathcal{U}_{[t,T]}} \int_t^T & - \left\langle \frac{\partial l[\boldsymbol{\psi}(\tau; \boldsymbol{x}, t, \boldsymbol{u}(\cdot))]}{\partial \boldsymbol{x}}, \right. \\ & \left. \times \boldsymbol{f}[\boldsymbol{\psi}(\tau; \boldsymbol{x}, t, \boldsymbol{u}(\cdot)), \boldsymbol{u}(\cdot)] \right\rangle \mathrm{d}\tau \\ & + l[\boldsymbol{\psi}(T; \boldsymbol{x}, t, \boldsymbol{u}(\cdot))]. \end{aligned} \tag{12}$$

By evaluating the Hamiltonian (4) of (12) at the boundary condition: $S(T, \boldsymbol{x}) = l(\boldsymbol{x})$, we obtain that

$$\begin{aligned} \mathcal{H}\left(\boldsymbol{x}, \frac{\partial S(T, \boldsymbol{x})}{\partial x}, \boldsymbol{u}\right) &= \left\langle \frac{\partial l(\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \right\rangle \\ &\quad - \left\langle \frac{\partial l(\boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \right\rangle = 0. \end{aligned}$$

Furthermore, by (3), it follows that

$$\frac{\partial S(T, \boldsymbol{x})}{\partial t} = - \inf_{\boldsymbol{u} \in \mathrm{U}} \mathcal{H}\left(\boldsymbol{x}, \frac{\partial S(T, \boldsymbol{x})}{\partial \boldsymbol{x}}, \boldsymbol{u}\right) = 0.$$

Because the Hamiltonian vanishes at the boundary condition, $\frac{\partial S(t, \boldsymbol{x})}{\partial t} = 0$ for all $t \in [0, T]$. This implies that $S(t, \boldsymbol{x}) = S(T, \boldsymbol{x}) = l(\boldsymbol{x})$, and therefore, $\tilde{V}(t, \boldsymbol{x}) = V(t, \boldsymbol{x}) - l(\boldsymbol{x})$.

Next, let us consider the sequence $\left\{S^{(i)},\ i \in 1, 2, \ldots\right\}$, as defined by Procedure 1 where $S^{(i)} := V^{(i)} - \tilde{V}^{(i)}$. On a similar note, we can verify that $\frac{\partial S^{(i)}(t, \boldsymbol{x})}{\partial t} = 0$. Consequently, $S^{(i)}(t, \boldsymbol{x}) = l(\boldsymbol{x})$, which implies that $\tilde{V}^{(i)}(t, \boldsymbol{x}) = V^{(i)}(t, \boldsymbol{x}) - l(\boldsymbol{x})$. □

The value function (2) can be computed by either directly solving the nonlinear PDE (3), or approximating it in the limit by solving the sequence of PDEs in Procedure 1. In the proposed approximation scheme, we follow the latter approach for certain solid reasons. Through introduction of Procedure 1, the optimization of the Hamiltonian term is eliminated from the HJB equation, conversely turning (5) into a linear PDE in (8). Later in Section 4, we show that this 'linearization' will allow us to introduce a collocation scheme that uses B-spline functions to approximate both the state and time dependent parts of the value function. The introduction of the value-iteration procedure circumvents the need to solve a complex nonlinear and non-smooth optimization problem, by turning it respectively into a sequence of quadratic programs. Additionally, the procedure also introduces an adaptive critic framework (Wang et al., 2009) for the finite-horizon optimal control problem, wherein the value-iteration cycle can be used as an update-step for the optimal feedback control law for systems with parametric uncertainties or slowly changing dynamics.

3. Preliminaries on multivariate simplex *B*-splines

This section provides an introduction into the theory of multivariate simplex B-splines. The machinery developed in this section will be necessary for the treatment of the collocation method in Section 4. Only the bare essentials of the theory is discussed, a more complete account on the subject can be found in Lai and Schumaker (2007).

3.1. Simplices, triangulations, and the barycentric coordinate system

A simplex B-spline is a piecewise polynomial function defined over a special geometric structure called a triangulation. A triangulation $\mathcal{T}$ is a special partitioning of a polytopic domain in $\mathbb{R}^n$ into a set of J simplices which are either disjoint, or share a common facet. Hence, if we let Δ denote an n-simplex formed by the convex hull of $n+1$ non-degenerate vertices $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^n$, then a triangulation can be defined as

$$\mathcal{T} := \bigcup_{i=1}^{J} \Delta_i, \quad \Delta_i \cap \Delta_j \in \left\{\varnothing, \tilde{\Delta}\right\},\ \forall i \neq j \tag{13}$$

where $\tilde{\Delta}$ is a k-simplex with $0 \leq k \leq n-1$.

For every simplex in the triangulation, a separate local coordinate system can be defined in terms of barycentric weights. In this barycentric coordinate system, every point $\boldsymbol{x} \in \mathbb{R}^n$ is uniquely represented by a vector $\boldsymbol{b} := (b_0, b_1, \ldots, b_n) \in \mathbb{R}^{n+1}$, whose elements denote the normalized weights that decompose $\boldsymbol{x}$ in terms of the simplex vertices, i.e.

$$\boldsymbol{x} = \sum_{i=0}^{n} b_i \boldsymbol{v}_i, \qquad \sum_{i=0}^{n} b_i = 1.$$

In the remainder of this paper, we denote $\boldsymbol{\beta}_{\Delta_j} : \mathbb{R}^n \mapsto \mathbb{R}^{n+1}$ as the function describing the mapping from Cartesian to barycentric coordinates for a specific simplex $\Delta_j \subseteq \mathcal{T}$.

3.2. The B-form polynomial

In the simplex B-spline, the polynomials in every simplex of the triangulation are expressed in B-form (de Boor, 1987). The B-form polynomial of degree d is defined as

$$p^{\Delta_j}(\boldsymbol{x}) := \sum_{|\kappa|=d} c_\kappa^{\Delta_j} B_\kappa^d(\boldsymbol{\beta}_{\Delta_j}(\boldsymbol{x})) \tag{14}$$

where $\kappa = (\kappa_0, \kappa_1, \ldots, \kappa_n) \in \mathbb{N}^{n+1}$ is a *multi-index* with $|\kappa| = \kappa_0 + \kappa_1 + \cdots + \kappa_n$. In (14), the term $B_\kappa^d(\boldsymbol{\beta}_{\Delta_j}(\boldsymbol{x}))$ denotes the individual basis polynomial (de Visser & Verhaegen, 2013)

$$B_\kappa^d(\boldsymbol{\beta}_{\Delta_j}(\boldsymbol{x})) := \begin{cases} \dfrac{d!}{\kappa_0!\kappa_1! \cdots \kappa_n!} b_0^{\kappa_0} b_1^{\kappa_1} \cdots b_n^{\kappa_n}, & \boldsymbol{x} \in \Delta_j \\ 0, & \text{otherwise.} \end{cases}$$

Note that the polynomial terms $B_\kappa^d(\boldsymbol{\beta}_{\Delta_j}(\boldsymbol{x}))$ can attain a non-zero value only when $\boldsymbol{x} \in \Delta_j$. Consequently, the B-form polynomial (14) has a local support, which is defined only for the simplex $\Delta_j \subseteq \mathcal{T}$. Outside of this simplex, the function is cut-off and becomes zero. The total number of individual basis polynomials in the summation of (14), denoted by $\hat{d}$, equals the number of valid permutations for $|\kappa| = d$

$$\hat{d} = \frac{(d+n)!}{n!d!}. \tag{15}$$

For convenience of notation, we express (14) in the vectorized notation (de Visser, Chu, & Mulder, 2009)

$$p^{\Delta_j}(\boldsymbol{x}) := \mathbf{B}_{\Delta_j}^d(\boldsymbol{x})\, \boldsymbol{c}^{\Delta_j}. \tag{16}$$

In the vectorized notation, the B-coefficients $\boldsymbol{c}^{\Delta_j} \in \mathbb{R}^{\hat{d} \times 1}$ and row vector of the basis polynomials $\mathbf{B}_{\Delta_j}^d \in \mathbb{R}^{1 \times \hat{d}}$ are constructed by sorting the summation terms in (14) lexicographically. Furthermore, de Boor (1987) showed that (14) represents a unique basis for the polynomial space $\mathbb{P}_d^n$.

3.3. The simplex B-spline

Approximating functions by pure polynomial interpolation is not a recommended practice. Apart from the susceptibility to unwanted oscillations (i.e. Runge's phenomenon), one often needs to deal with dense regression matrices. These drawbacks of polynomials can be overcome by using piecewise polynomials, or spline functions, as a replacement. In the multivariate simplex *B*-spline, the local polynomials function are expressed in *B*-form (14), and are defined over the simplices of the triangulation. To enforce a certain order of smoothness m among all sub-domains of the triangulation, the *B*-coefficients of every local polynomial functions need to satisfy a certain number of R^* unique, inter-simplex, homogeneous equality constraints. These constraints are given by

$$\mathrm{H}\boldsymbol{c} = 0 \tag{17}$$

where $\mathrm{H} \in \mathbb{R}^{R^* \times J\cdot\hat{d}}$ is the smoothness matrix defined in de Visser et al. (2009) and Lai and Schumaker (2007), and $\boldsymbol{c} \in \mathbb{R}^{J\cdot\hat{d}\times 1}$ is the global vector of *B*-coefficients

$$\boldsymbol{c} := \begin{bmatrix} \boldsymbol{c}^{\Delta_1\top} & \cdots & \boldsymbol{c}^{\Delta_J\top} \end{bmatrix}^\top. \tag{18}$$

We formulate the following definition for the simplex *B*-spline.

Definition 1 (*Simplex B-Spline*)**.** Let $\mathcal{T}$ be a triangulation of the polytopic domain $\Omega \subset \mathbb{R}^n$ into J simplices. A simplex *B*-spline of degree $d \geq 1$, and continuity order $m \geq 0$, is a function $s_d^{\mathcal{T},m} : \Omega \mapsto \mathbb{R}$, such that

$$s_d^{\mathcal{T},m}(\boldsymbol{x}) := \mathbf{B}_{gl}^d(\boldsymbol{x})\boldsymbol{c} \tag{19}$$

where $\boldsymbol{c} \in \mathbb{R}^{J\hat{d}\times 1}$ satisfies (17), and $\mathbf{B}_{gl}^d(\boldsymbol{x}) \in \mathbb{R}^{1\times J\hat{d}}$ is the global vector of basis polynomials given by

$$\mathbf{B}_{gl}^d(\boldsymbol{x}) := \begin{bmatrix} 0^{1\times(j-1)\hat{d}} & \mathbf{B}_{\Delta_j}^d(\boldsymbol{x}) & 0^{1\times(J-j)\hat{d}} \end{bmatrix}, \quad \boldsymbol{x} \in \Delta_j.$$

The spline function $s_d^{\mathcal{T},m}$ is uniquely defined in terms of the *B*-coefficients vector (18). The *B*-coefficients have a geometric interpretation in terms of the so-called B-net. Associated with every *B*-coefficient is a spatial location in which the corresponding *B*-form polynomial term has a maximum influence. Small perturbations of individual *B*-coefficients lead to only local changes in the function in the neighborhood around the associated spatial position. The spatial position correspond to the domain points of the *B*-form polynomial (de Boor, 1987). In barycentric coordinates, these points are found by

$$\boldsymbol{b}_\kappa = \frac{\kappa}{d}, \qquad |\kappa| = d. \tag{20}$$

In Fig. 1, the domain points are shown for a bivariate spline function of degree 3.

Lai and Schumaker (2007) have shown that the B-form polynomial evaluated at a domain point positioned on a simplex edge equals the value of the B-coefficient of that domain point. Later in Section 4, it will become evident that this specific property will allow us to deal with the PDE boundary conditions in a very straightforward manner.

3.4. B-spline derivatives

The function $s_d^{\mathcal{T},m} \in C^m(\Omega)$ is guaranteed to be m-times differentiable on the domain Ω. In this paper, we are interested in determining the first-order partial derivatives of (19). Using the formulation in de Visser, Chu, and Mulder (2011) for the directional derivatives, the partial derivative of the B-form polynomial $p^{\Delta_j}(\boldsymbol{x})$ with respect to x_k ($k = 1, \ldots, n$) at a given $\boldsymbol{x} \in \Delta_j$, can be determined by evaluating

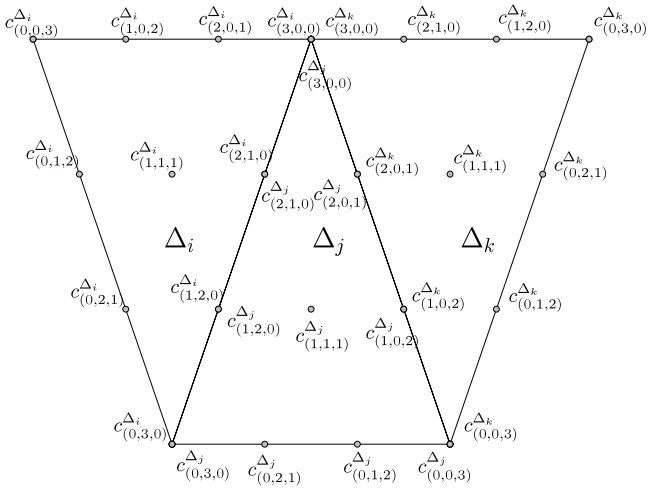


Fig. 1. The domain points for a 3rd degree spline function on a triangulation consisting of three triangles: Δ_i, Δ_j and Δ_k.

$$\frac{\partial p^{\Delta_j}(\boldsymbol{x})}{\partial x_k} = \frac{\mathbf{B}_{\Delta_j}^d(\boldsymbol{x})}{\partial x_k}\boldsymbol{c}^{\Delta_j} \tag{21}$$

where

$$\frac{\mathbf{B}_{\Delta_j}^d(\boldsymbol{x})}{\partial x_k} := \frac{d!}{(d-1)!}\mathbf{B}_{\Delta_j}^{d-1}(\boldsymbol{x})\,\mathbf{P}^{d,d-1}(\mathbf{a}_k) \in \mathbb{R}^{1\times\hat{d}}$$

with $\mathbf{P}^{d,d-1}(\mathbf{a}) \in \mathbb{R}^{\frac{(d-1+2)!}{2(d-1)!}\times\hat{d}}$ the de-Casteljau matrix of degree d to $d-m$ from de Visser et al. (2011), and $\mathbf{a}_k$ the directional coordinate of the unit vector $\vec{\boldsymbol{e}}_k$ in barycentric coordinates, i.e. $\mathbf{a}_k := \boldsymbol{\beta}_{\Delta_j}(\vec{\boldsymbol{e}}_k) - \boldsymbol{\beta}_{\Delta_j}(0)$. We denote the spline partial derivatives by

$$\frac{\partial s_d^{\mathcal{T},m}(\boldsymbol{x})}{\partial x_k} = \frac{\partial \mathbf{B}_{gl}^d(\boldsymbol{x})}{\partial x_k}\boldsymbol{c}, \quad k = 1, \ldots, n \tag{22}$$

with

$$\frac{\partial \mathbf{B}_{gl}^d(\boldsymbol{x})}{\partial x_k} := \begin{bmatrix} 0^{(j-1)\times\hat{d}} & \dfrac{\mathbf{B}_{\Delta_j}^d(\boldsymbol{x})}{\partial x_k} & 0^{(J-j)\times\hat{d}} \end{bmatrix} \in \mathbb{R}^{1\times J\hat{d}}.$$

4. The collocation method

In this section, we present a collocation method that approximates the solution of (8) fully in terms of spline functions. The method that we lay out in this section is presented in a very general framework that directly can be applied on any bounded, n-dimensional polytopic domain $\Omega \subset \mathbb{R}^n$ which is decomposable into a triangulation.

4.1. A full state and time parametrization of the value function

Our motivation to use polynomial-type basis functions in the discretization is supported by the well-known Weierstrass approximation theorem. Since (8) describes a continuous surface that evolves continuously over time, there exists a complete set of polynomials $\{\phi_j(\boldsymbol{x})\}_{j=1}^\infty$ such that

$$V^{(i)}(t, \boldsymbol{x}) = l(\boldsymbol{x}) + \sum_{j=1}^{\infty} \phi_j(\boldsymbol{x})\, c_j^{(i)}(t) \tag{23}$$

with $c_j^{(i)}(t) : [0, T] \mapsto \mathbb{R}$ as *time-dependent* coefficients. To find a suitable parametrization for the value function in terms of only

constant coefficients, we further express the continuous functions $c_j^{(i)}(t)$ by

$$c_j^{(i)}(t) = \sum_{k=1}^{\infty} \varphi_k(t)\, c_{jk}^{(i)} \tag{24}$$

with $\{\varphi_k(t)\}_{k=1}^{\infty}$ a complete polynomial basis w.r.t. the time variable. By substitution of (24) into (23), we obtain a parametrization for the value function which complies with the evolutionary nature of the underlying PDE. This parametrization of the value function in terms of only constant coefficients takes the form

$$V^{(i)}(t, \boldsymbol{x}) = \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \phi_j(\boldsymbol{x})\, \varphi_k(t)\, c_{jk}^{(i)}. \tag{25}$$

4.2. The spline-based approximate solution

Given the computational and numerical benefits of spline functions over regular polynomials, we suggest to approximate $V^{(i)}$ along the same lines as (23) using

$$V_{\text{apprx}}^{(i)}\left(t, \boldsymbol{x}; \boldsymbol{c}^{(i)}(t)\right) = l(\boldsymbol{x}) + \mathbf{B}_{gl}^{d_{\boldsymbol{x}}}(\boldsymbol{x}) \boldsymbol{c}^{(i)}(t) \tag{26}$$

where $\mathbf{B}_{gl}^{d_{\boldsymbol{x}}}(\boldsymbol{x}) \in \mathbb{R}^{1 \times J_{\boldsymbol{x}} \hat{d}_{\boldsymbol{x}}}$ is the global vector of basis functions associated to the (spatial) simplex B-spline $s_{d_{\boldsymbol{x}}}^{\mathcal{T}_{\boldsymbol{x}}, m_{\boldsymbol{x}}} : \Omega \mapsto \mathbb{R}$ of degree $d_{\boldsymbol{x}}$, triangulation $\mathcal{T}_{\boldsymbol{x}}$ consisting of $J_{\boldsymbol{x}}$ simplices, and continuity order $m_{\boldsymbol{x}}$. Note that we deliberately add the terminal cost $l(\boldsymbol{x})$ to the approximation in (26), so that the boundary condition of (8) is satisfied by *construction* when setting $\boldsymbol{c}^{(i)}(T) = 0$.

Substituting (26) into (8) yields the differential equation

$$\begin{aligned} e^{(i)}(t, \boldsymbol{x}) = {} & \mathbf{B}_{gl}^{d_{\boldsymbol{x}}}(\boldsymbol{x}) \dot{\boldsymbol{c}}^{(i)}(t) + \left\{ \sum_{k=1}^{n} \left(\frac{\partial \mathbf{B}_{gl}^{d_{\boldsymbol{x}}}(\boldsymbol{x})}{\partial x_k} \boldsymbol{c}^{(i)}(t) + \frac{\partial l(\boldsymbol{x})}{\partial x_k} \right) \right. \\ & \left. \times f_k\left(\boldsymbol{x}, \boldsymbol{g}_{\text{apprx}}^{(i)}\left(t, \boldsymbol{x}; \boldsymbol{c}^{(i-1)}(t)\right)\right) \right\} \\ & + r\left(\boldsymbol{x}, \boldsymbol{g}_{\text{apprx}}^{(i)}\left(t, \boldsymbol{x}; \boldsymbol{c}^{(i-1)}(t)\right)\right) \end{aligned} \tag{27}$$

where $\boldsymbol{g}_{\text{apprx}}^{(i)}\left(t, \boldsymbol{x}; \boldsymbol{c}^{(i-1)}(t)\right)$ is given by

$$\boldsymbol{g}_{\text{apprx}}^{(i)}\left(t, \boldsymbol{x}; \boldsymbol{c}^{(i-1)}(t)\right) = \arg \inf_{\boldsymbol{u} \in \mathrm{U}} \mathcal{H} \times \left(\boldsymbol{x}, \frac{\partial V_{\text{apprx}}^{(i-1)}\left(t, \boldsymbol{x}; \boldsymbol{c}^{(i-1)}(t)\right)}{\partial \boldsymbol{x}}, \boldsymbol{u} \right). \tag{28}$$

In addition to (27), the following smoothness conditions (17) need to be satisfied for all $t \in [0, T]$

$$\mathrm{H}_{\boldsymbol{x}} \boldsymbol{c}^{(i)}(t) = 0, \quad \mathrm{H}_{\boldsymbol{x}} \in \mathbb{R}^{R_{\boldsymbol{x}}^{*} \times J_{\boldsymbol{x}} \cdot \hat{d}_{\boldsymbol{x}}}. \tag{29}$$

Previous work (Cheng et al., 2007; Huang et al., 2006) focused on integrating (27) backwards in time using ordinary finite difference approximations. In this paper, we propose to discretize (27) further among the lines of (24) and (25) by replacing the coefficients with univariate B-spline functions. Proceeding in this direction results in an approximation of $V^{(i)}$ fully in terms of piecewise-polynomial functions. Using a tensor product notation, we express the B-coefficients by

$$\boldsymbol{c}^{(i)}(t) = \left(\mathbb{I}_{J_{\boldsymbol{x}} \hat{d}_{\boldsymbol{x}}} \otimes \mathbf{B}_{gl}^{d_t}(t) \right) \boldsymbol{C}^{(i)} \tag{30}$$

where $\mathbf{B}_{gl}^{d_t}(t) \in \mathbb{R}^{1 \times J_t \hat{d}_t}$ is the global vector of basis functions associated to the (temporal) simplex B-spline $s_{d_t}^{\mathcal{T}_t, m_t} : [0, T] \mapsto \mathbb{R}$ of degree d_t, triangulation $\mathcal{T}_t$ consisting of J_t simplices, and continuity order m_t. The following smoothness conditions need to also be satisfied

$$\left(\mathbb{I}_{J_{\boldsymbol{x}} \hat{d}_{\boldsymbol{x}}} \otimes \mathrm{H}_t \right) \boldsymbol{C}^{(i)} = 0, \quad \mathrm{H}_t \in \mathbb{R}^{R_t^{*} \times J_t \cdot \hat{d}_t}. \tag{31}$$

Substituting (30) into (26), we obtain

$$V_{\text{apprx}}^{(i)}\left(t, \boldsymbol{x}; \boldsymbol{C}^{(i)}\right) = l(\boldsymbol{x}) + \left(\mathbf{B}_{gl}^{d_{\boldsymbol{x}}}(\boldsymbol{x}) \otimes \mathbf{B}_{gl}^{d_t}(t) \right) \boldsymbol{C}^{(i)}. \tag{32}$$

The expanded coefficient vector: $\boldsymbol{C}^{(i)} \in \mathbb{R}^{J_{\boldsymbol{x}} \hat{d}_{\boldsymbol{x}} J_t \hat{d}_t \times 1}$ has to satisfy smoothness conditions that arise from both the spatial spline function $s_{d_{\boldsymbol{x}}}^{\mathcal{T}_{\boldsymbol{x}}, m_{\boldsymbol{x}}}$, and temporal spline functions $s_{d_t}^{\mathcal{T}_t, m_t}$. The smoothness conditions are

$$\mathrm{H}_{global} \boldsymbol{C}^{(i)} = 0, \qquad \mathrm{H}_{global} := \begin{bmatrix} \mathrm{H}_{\boldsymbol{x}} \otimes \mathbb{I}_{J_t \hat{d}_t} \\ \mathbb{I}_{J_{\boldsymbol{x}} \hat{d}_{\boldsymbol{x}}} \otimes \mathrm{H}_t \end{bmatrix} \tag{33}$$

with $\mathrm{H}_{\boldsymbol{x}}$ and H_t defined in (29) and (31), respectively.

Apart from the smoothness conditions, (32) must also satisfy the boundary condition of (8), which is enforced through setting $c_{\kappa_{\boldsymbol{x}}}^{\Delta_{j_{\boldsymbol{x}}}}(T) = 0$. Since the value of a B-form polynomial function at a vertex point equals the value of the B-coefficient positioned at that vertex point, the unique properties of the simplex B-spline requires us to simply set all the B-coefficients *positioned at the terminal time* equal to zero. We denote these conditions by the expression

$$\mathrm{G} \boldsymbol{C}^{(i)} = 0 \tag{34}$$

where $\mathrm{G} \in \mathbb{R}^{J_{\boldsymbol{x}} \hat{d}_{\boldsymbol{x}} \times J_{\boldsymbol{x}} \hat{d}_{\boldsymbol{x}} J_t \hat{d}_t}$ is highly sparse.

The second term in (32) can be interpreted as a hybrid between a tensor and simplex spline. Geometrically, this is a spline function defined on a partitioning of a domain $[0, T] \times \Omega$ into $(n+1)$-dimensional prismatic polytopes. The simplex spline and its B-coefficients are basically extruded onto an extra dimension representing the temporal nature of the function. For the bivariate case, this expansion will yield a structure of triangular prisms. In Fig. 2 we visualize this prismatic structure for the triangulation shown in Fig. 1.

The resulting hybrid architecture originates from the parametrization of the value function in (25). The suitability of (32) as

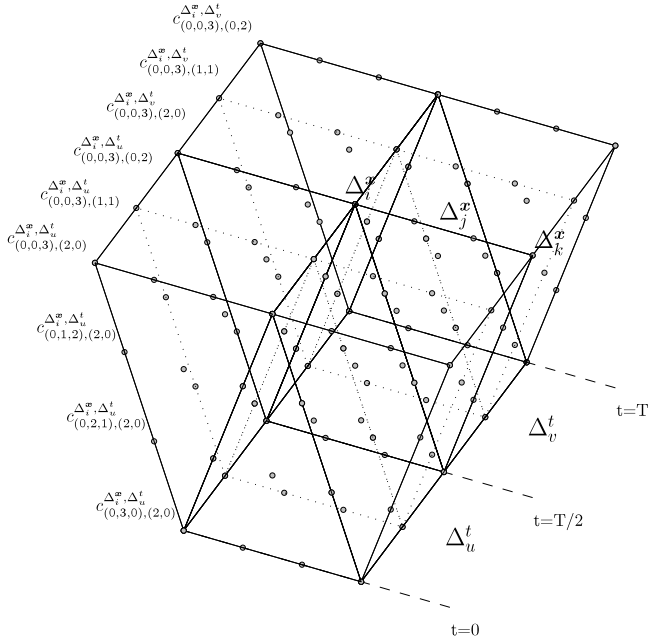


Fig. 2. Geometric interpretation of the time-varying spline for the triangulation of Fig. 1 with triangles: $\Delta_i^{\boldsymbol{x}}$, $\Delta_j^{\boldsymbol{x}}$ and $\Delta_k^{\boldsymbol{x}}$. The temporal variable is discretized using a 2nd degree spline function consisting of two sub-intervals: Δ_u^t, and Δ_v^t.

a candidate solution for (8) is hence supported with the argument that the chosen approximation structure complies with the underlying physics of the PDE.

4.3. Formulation as a quadratic program

Direct substitution of (32) into (8) yields the residual error

$$e^{(i)}(t,\boldsymbol{x}) = \boldsymbol{\alpha}\left(t,\boldsymbol{x};\boldsymbol{C}^{(i-1)}\right)\boldsymbol{C}^{(i)} + \beta\left(t,\boldsymbol{x};\boldsymbol{C}^{(i-1)}\right) \tag{35}$$

where

$$\begin{aligned}\boldsymbol{\alpha}\left(t,\boldsymbol{x};\boldsymbol{C}^{(i-1)}\right) &= \mathbf{B}_{gl}^{d_{\boldsymbol{x}}}(\boldsymbol{x}) \otimes \frac{\mathrm{d}\mathbf{B}_{gl}^{d_t}(t)}{\mathrm{d}t} \\ &\quad + \sum_{k=1}^{n} f_k\left(\boldsymbol{x}, \boldsymbol{g}_{\text{apprx}}^{(i)}\left(t,\boldsymbol{x};\boldsymbol{C}^{(i-1)}\right)\right) \\ &\quad \times \left(\frac{\partial \mathbf{B}_{gl}^{d_{\boldsymbol{x}}}(\boldsymbol{x})}{\partial x_k} \otimes \mathbf{B}_{gl}^{d_t}(t)\right) \end{aligned} \tag{36}$$

$$\begin{aligned}\beta\left(t,\boldsymbol{x};\boldsymbol{C}^{(i-1)}\right) &= r\left(\boldsymbol{x}, \boldsymbol{g}_{\text{apprx}}^{(i)}\left(t,\boldsymbol{x};\boldsymbol{C}^{(i-1)}\right)\right) \\ &\quad + \sum_{k=1}^{n} f_k\left(\boldsymbol{x}, \boldsymbol{g}_{\text{apprx}}^{(i)}\left(t,\boldsymbol{x};\boldsymbol{C}^{(i-1)}\right)\right) \frac{\partial l(\boldsymbol{x})}{\partial x_k}. \end{aligned} \tag{37}$$

The residual error of (35) is an affine function in terms of the coefficient vector: $\boldsymbol{C}^{(i)}$. The affine relationship is an immediate consequence of the application of Procedure 1. If value-iteration was not used in the overall approximation scheme, then $\boldsymbol{C}^{(i-1)} = \boldsymbol{C}^{(i)} = \boldsymbol{C}$, and the residual error will have a nonlinear relationship w.r.t. the approximation coefficients. Moreover, the relationship may also be non-smooth, since $\boldsymbol{g}_{\text{apprx}}^{(i)}$ is often a discontinuous function of the approximation coefficients.

In any case, we apply the *method of weighted residuals* (Finlayson, 1972) to determine an estimate for $\boldsymbol{C}^{(i)}$, such that the residual error (35) is forced to zero in some average sense, i.e.

$$\int_0^T \int_{\Omega} \boldsymbol{w}(t,\boldsymbol{x})\, e^{(i)}(t,\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}\mathrm{d}t = 0 \tag{38}$$

with $\boldsymbol{w}(t,\boldsymbol{x}) := \begin{bmatrix} w_1(t,\boldsymbol{x}) & \cdots & w_N(t,\boldsymbol{x}) \end{bmatrix}^T$ a vector of preselected weighting functions. Depending on the choice of $\boldsymbol{w}(t,\boldsymbol{x})$, various sub-methods can be formulated (e.g. Galerkin, least-squares, collocation, sub-domain).

In the collocation scheme, the weighting functions are chosen to be Dirac delta functions displaced to various 'collocation points' spanning the computational domain. Let $N = N_t N_{\boldsymbol{x}}$, and consider the functions

$$\boldsymbol{\delta}_{\boldsymbol{x}}(\boldsymbol{x}) := \begin{bmatrix} \delta(\boldsymbol{x}-\boldsymbol{x}_1) \\ \delta(\boldsymbol{x}-\boldsymbol{x}_2) \\ \vdots \\ \delta\left(\boldsymbol{x}-\boldsymbol{x}_{N_{\boldsymbol{x}}}\right) \end{bmatrix}, \qquad \boldsymbol{\delta}_t(t) := \begin{bmatrix} \delta(t-t_1) \\ \delta(t-t_2) \\ \vdots \\ \delta\left(t-t_{N_t}\right) \end{bmatrix}.$$

By setting $\boldsymbol{w}(t,\boldsymbol{x}) = \boldsymbol{\delta}_t(t) \otimes \boldsymbol{\delta}_{\boldsymbol{x}}(\boldsymbol{x})$, (38) can be re-written to

$$\int_0^T \boldsymbol{\delta}_t(t) \otimes \left(\int_{\Omega} \boldsymbol{\delta}_{\boldsymbol{x}}(\boldsymbol{x})\, e^{(i)}(t,\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}\right) \mathrm{d}t = 0. \tag{39}$$

By the property: $\int \delta(x-a) f(x)\mathrm{d}x = f(a)$, we have that

$$\int_{\Omega} \boldsymbol{\delta}_{\boldsymbol{x}}(\boldsymbol{x})\, e^{(i)}(t,\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = \begin{bmatrix} e^{(i)}(t,\boldsymbol{x}_1) \\ e^{(i)}(t,\boldsymbol{x}_2) \\ \vdots \\ e^{(i)}(t,\boldsymbol{x}_{N_{\boldsymbol{x}}}) \end{bmatrix}. \tag{40}$$

In effect, (39) breaks-down to setting the residual error (35) equal to zero at the collocation points

$$e^{(i)}(t_j,x_k) = 0, \quad \begin{matrix} j = 1,2,\ldots,N_t \\ k = 1,2,\ldots,N_{\boldsymbol{x}}. \end{matrix} \tag{41}$$

Based on (27), we may view (40) as a system of first-order, linear ODEs. Henceforth, (39) can be interpreted as solving a system of ODEs through collocation. Apart from the conditions (41), we also need to ensure that (32) satisfies the smoothness conditions (33) and boundary conditions (34). Collectively, the conditions (41), (33), and (34) are embedded into an optimization problem where the objective is to minimize the squared sum of the residuals. We have the quadratic program

$$\begin{aligned} &\text{minimize} \quad J(\boldsymbol{C}^{(i)}) := \|\mathrm{A}\boldsymbol{C}^{(i)} - \boldsymbol{\beta}\|_2^2 \\ &\text{subject to:} \quad \mathrm{H}_{global}\boldsymbol{C}^{(i)} = 0, \quad \mathrm{G}\boldsymbol{C}^{(i)} = 0 \end{aligned} \tag{42}$$

where

$$\mathrm{A} := [\boldsymbol{\alpha}(t_j,\boldsymbol{x}_k;\boldsymbol{C}^{(i-1)})]_{j=1,k=1}^{N_t,N_{\boldsymbol{x}}},$$

and

$$\boldsymbol{\beta} := [\beta(t_j,\boldsymbol{x}_k;\boldsymbol{C}^{(i-1)})]_{j=1,k=1}^{N_t,N_{\boldsymbol{x}}}.$$

4.4. Summary of the algorithm

To summarize the overall approximation scheme, we end up with the following algorithm.

Algorithm 1. Initialize $\hat{\boldsymbol{C}}^{(0)} = 0$, and set $\epsilon > 0$. Repeat the following steps:

1. Approximate (7) with $\boldsymbol{g}_{\text{apprx}}^{(i)}(t,\boldsymbol{x};\hat{\boldsymbol{C}}^{(i-1)})$.
2. Solve quadratic program (42) and set $\hat{\boldsymbol{C}}^{(i)} := \arg\min J(\boldsymbol{C}^{(i)})$.

Until $\| \hat{\boldsymbol{C}}^{(i)} - \hat{\boldsymbol{C}}^{(i-1)} \|_{\infty} < \epsilon$.

Note that Theorem 1 does not immediately imply convergence for Algorithm 1 as approximations errors can successively propagate into the next iteration cycle, and cause a divergence. A necessity is to thus have sufficient approximation power in the overall spline discretization. Lai and Schumaker (2007) have shown that the simplex *B*-spline can approximate any smooth function lying in a Sobolev space. Regardless, the ability to approximate (26) strongly depends also on the chosen weighted-residual method. In the collocation approach, convergent behavior is difficult to prove and depends strongly on the selection of the collocation points.

In the temporal dimension, a good choice is to take $\{t_j\}_{j=1}^{N_t} \in [0,T]$ as the *Gaussian points* (i.e. the roots of d_tth Legendre polynomial) relative to each sub-interval of $\mathcal{T}_t$. We support this choice with results in de Boor and Swartz (1973) which show that a mth order differential equation can be approximated under this distribution of points with order $\mathcal{O}(|\Delta_{\max}^t|^{m+d_t})$, where $|\Delta_{\max}^t|$ denotes the length of largest subinterval in $\mathcal{T}_t$. In the spatial dimension, the choice of collocation points is harder to justify. In the present work, we take $\{\boldsymbol{x}_k\}_{k=1}^{N_{\boldsymbol{x}}} \in \Omega$ to be the *domain points* (20).

Given the current scope of the paper, no rigorous convergence proofs are available for Algorithm 1. However, simulation result, as will be provided in Section 6, are promising and suggest that further development of such algorithms is worthwhile.

5. Complexity of the *B*-spline method

The computational complexity of the proposed method is dictated by the complexity of the quadratic program (42). In this section, we analyze the computational complexity of the proposed *B*-spline method. More specifically, we show that, when compared to a RBF based discretization, a *B*-spline based approximation structure leads to more favorable properties in both space-complexity and time-complexity.

5.1. Space-complexity

In a spline based discretization, the quadratic program (42) is highly sparse in both the objective function and constraints. The

sparsity in the constraints is caused by the fact that smoothness conditions (33) are shared only with neighboring simplices. Concerning the objective function, the following can be stated.

Proposition 2. *Let nz(A) denote the number of nonzero entries of* A *from (42), and take:*

(i) $\{t_j\}_{j=1}^{N_t}$ *as the Gaussian points relative to each sub-interval of the temporal triangulation* $\mathcal{T}_t$.
(ii) $\{\boldsymbol{x}_k\}_{k=1}^{N_x}$ *as the domain points of the spatial triangulation* $\mathcal{T}_{\boldsymbol{x}}$.

Then $nz(\mathrm{A}) \in \mathcal{O}(J_{\boldsymbol{x}} J_t)$, *where* $J_{\boldsymbol{x}}$ *denotes the number of simplices in the spatial discretization and* J_t *denotes the number of sub-intervals in the time discretization.*

Proof. To prove the statement, we derive an upper bound for $nz(\mathrm{A})$ in terms of the spline parameters: $J_{\boldsymbol{x}}, J_t, d_{\boldsymbol{x}}$ and d_t.

Let $(t, \boldsymbol{x}) \in (\Delta_j^t, \Delta_k^t)$, by Definition 1, we may rewrite (36) as

$$\boldsymbol{\alpha}(t, \boldsymbol{x}) = \begin{bmatrix} 0^{1\times(k-1)\hat{d}_{\boldsymbol{x}}(j-1)\hat{d}_t} & \boldsymbol{\alpha}^*(t, \boldsymbol{x}) & 0^{1\times(J_{\boldsymbol{x}}-k)\hat{d}_{\boldsymbol{x}}(J_t-j)\hat{d}_t} \end{bmatrix}$$

with

$$\boldsymbol{\alpha}^*(t, \boldsymbol{x}) = \mathbf{B}_{\Delta_k^{\boldsymbol{x}}}^{d_{\boldsymbol{x}}}(\boldsymbol{x}) \otimes \frac{\mathrm{d}\mathbf{B}_{\Delta_j^t}^{d_t}(t)}{\mathrm{d}t} + \sum_{k=1}^{n} f_k\left(\boldsymbol{x}, \boldsymbol{g}_{\mathrm{apprx}}^{(i)}(t, \boldsymbol{x})\right) \left(\frac{\partial \mathbf{B}_{\Delta_k^{\boldsymbol{x}}}^{d_{\boldsymbol{x}}}(\boldsymbol{x})}{\partial x_k} \otimes \mathbf{B}_{\Delta_j^t}^{d_t}(t) \right). \quad (43)$$

From (43) and (15), we deduce that

$$nz(\boldsymbol{\alpha}(t, \boldsymbol{x})) = nz(\boldsymbol{\alpha}^*(t, \boldsymbol{x})) \leq \hat{d}_{\boldsymbol{x}}\hat{d}_t = \left(\frac{(d_{\boldsymbol{x}}+n)!}{d_{\boldsymbol{x}}!n!}\right)(d_t+1).$$

The number of domain points in $\mathcal{T}_{\boldsymbol{x}}$ equals $J_{\boldsymbol{x}}\hat{d}_{\boldsymbol{x}}$ and the number of Gaussian points in $\mathcal{T}_t$ equals $J_t(\hat{d}_t - 1)$. We hence obtain that

$$nz(\mathrm{A}) = J_{\boldsymbol{x}}\hat{d}_{\boldsymbol{x}}J_t(\hat{d}_t - 1)nz(\boldsymbol{\alpha}(t, \boldsymbol{x})) \leq J_{\boldsymbol{x}}J_t\left(\frac{(d_{\boldsymbol{x}}+n)!}{d_{\boldsymbol{x}}!n!}\right)^2 (d_t+1)\, d_t$$

which proves our claim.

In terms of space complexity, Proposition 2 has the following implication. The approximation power of $V_{\mathrm{apprx}}^{(i)}$ can be improved by increasing the degrees of the polynomials and/or making the triangulations more dense. However, an increase in the number of prismatic polytopes (i.e. by increasing $J_{\boldsymbol{x}}$ and/or J_t) causes only a *linear* growth in $nz(\mathrm{A})$. Consequently, the approximation power of a *B*-spline based discretization can be improved with $\mathcal{O}(J_{\boldsymbol{x}}J_t)$ efficiency in storage requirement. In a RBF-based discretization, any improvement in the approximation power comes at the cost of a quadratic increase in storage requirement, since A would be dense.

5.2. Time-complexity

The time-complexity of the *B*-spline method is directly related to the efficiency in solving the KKT conditions associated with (42). The KKT conditions can efficiently be solved using the matrix iterative method in Awanou et al. (2005), which in turn is a specific case of the more general augmented Lagrangian and Method of Multipliers (Boyd, Parikh, Chu, Peleato, & Eckstein, 2011). The main time-consuming element of the matrix iterative method is computing the Cholesky factorization of $\mathrm{A}^T\mathrm{A} + \rho(\mathrm{H}_{global}^T\mathrm{H}_{global} + \mathrm{G}^T G)$. In a RBF based discretization, the time-complexity of computing the Cholesky factorization will be cubic in terms of number of centers used in the entire discretization, since A would be completely dense. In case of the *B*-spline method, a significant efficiency boost can already be obtained using sparse Cholesky factorization algorithms that aim to minimize the number of fill-in's. The exact gains time-complexity is difficult quantify, since it depends in complicated way on the sparsity pattern. However, the complexity would be no worse than $\mathcal{O}((J_{\boldsymbol{x}}J_t\hat{d}_{\boldsymbol{x}}\hat{d}_t)^3)$ flops, which is comparable to a RBF based method. Nonetheless, apart from the sparsity property, the matrix A satisfies the following property.

Proposition 3. *The matrix* A *is row permutable to a block-diagonal matrix consisting of* $J_{\boldsymbol{x}}J_t$ *blocks on the main diagonal.*

Proof. To show that A is permutable to a block diagonal matrix implies showing that the objective function (42) is separable, which can be proven straightforwardly from (43).

In terms of time-complexity, Proposition 3 has the following implication. Since A is of block-diagonal structure, one can resort to methods such as the Alternating Direction Method of Multipliers (ADDM), as in Boyd et al. (2011), to take advantage of this property. With the ADDM algorithm, one effectively needs to only compute the Cholesky factorization for a set of $J_{\boldsymbol{x}}J_t$ smaller matrices of size $\hat{d}_{\boldsymbol{x}}\hat{d}_t \times \hat{d}_{\boldsymbol{x}}\hat{d}_t$, which significantly reduces computational costs. More specifically, a time-complexity of $\mathcal{O}(J_{\boldsymbol{x}}J_t(\hat{d}_{\boldsymbol{x}}\hat{d}_t)^3)$ flops can be achieved, which is a factor of $(J_{\boldsymbol{x}}J_t)^2$ improvement overall.

The crux of our method lies in the properties discussed in Propositions 2 and 3. These properties are a immediate consequences of the domain decomposition into simplices and make the proposed method highly suitable for implementation under parallel computing architectures. Using techniques from distributed optimization, we believe that this parallelization of computations will allow us to tackle large-scale problems involving high density triangulations more effectively.

6. Simulation results

In this section, we demonstrate our method on two benchmark example problems. We provide numerical evidence that show a steady improvement of the approximation quality for increasingly more dense triangulations. Note that the optimization of the discretization parameters is a complex subject by itself, and hence is out of the scope for the present study. Consequently, we limit our analysis to equally sized type-I triangulation configurations (Lai & Schumaker, 2007), low degree polynomials (i.e. cubic, quartic), and continuity order of $m_{\boldsymbol{x}} = m_t = 1$.

6.1. Example I: a trivial example with analytic solution

The first example we discuss was also presented in Huang et al. (2006). Consider the scalar system

$$\dot{x} = xu$$

with $u \in \mathrm{U} := \{u \in \mathbb{R} \mid 0 \leq u \leq 1\}$, and define the cost criteria

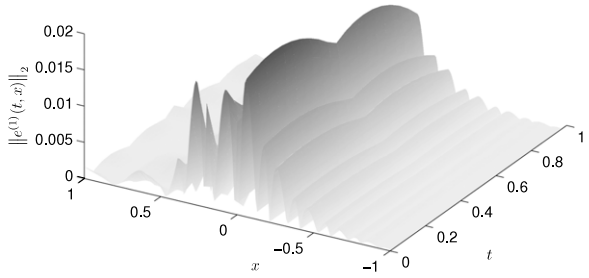
$$r(x, u) = 0, \quad l(x) = -x, \quad T = 1.$$

This yields the HJB equation

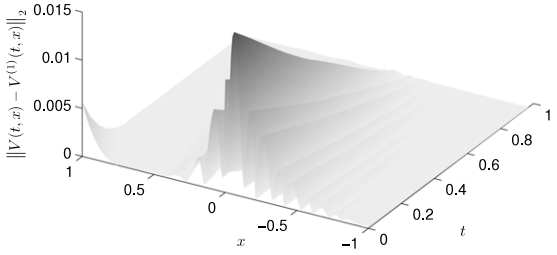
$$\frac{\partial V(t, x)}{\partial t} + \inf_{0\leq u\leq 1}\left[\frac{\partial V(t, x)}{\partial x} xu\right] = 0 \quad (44)$$

with boundary condition $V(1, x) = -x$. The exact viscosity solution for this PDE is known, and is given by the expression

$$V(t, x) = \begin{cases} -xe^{1-t}, & x > 0 \\ -x, & x \leq 0. \end{cases} \quad (45)$$

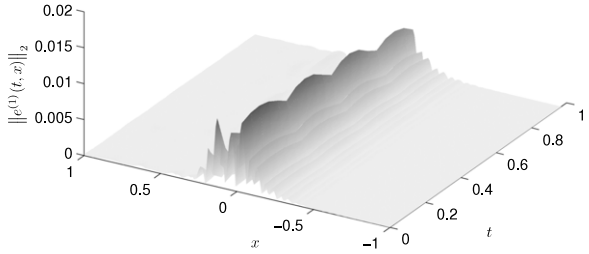


(a) The 2-norm of the residual error, i.e. $\left\|e^{(1)}(t,x)\right\|_2$.

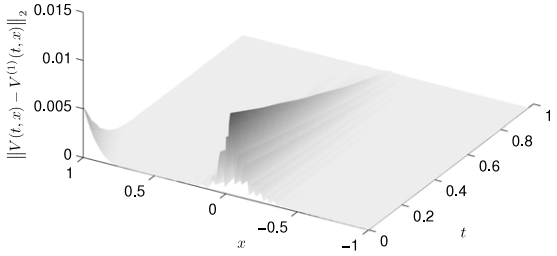


(b) The 2-norm of the approximation error, i.e. $\left\|V(t,x)-V^{(1)}(t,x)\right\|_2$.

Fig. 3. The 2-norm of the approximation error and residual error for the settings: $J_x = 40, J_t = 4, d_x = d_t = 3, m_x = m_t = 1$.



(a) The 2-norm of the residual error, i.e. $\left\|e^{(1)}(t,x)\right\|_2$.



(b) The 2-norm of the approximation error, i.e. $\left\|V(t,x)-V^{(1)}(t,x)\right\|_2$.

Fig. 4. The 2-norm of the approximation error and residual error for the settings: $J_x = 20, J_t = 2, d_x = d_t = 3, m_x = m_t = 1$.

Our aim is to approximate the solution of (44) for the state domain $\Omega = [-1, 1]$. Notice that the above considered optimal control problem satisfies all the conditions of Theorem 1, hence the value-iteration procedure is known to converge. Moreover, it suffices that this convergence happens in one step, i.e. $V^{(i)}(t,x) = V(t,x)$ and $g^{(i)}(t,x) = g^*(t,x)$ for $i = 1, 2$, etc. Therefore, in our analysis we only consider the first iteration of Procedure 1 wherein we solve

$$\frac{\partial V^{(1)}(t,x)}{\partial t} + \frac{\partial V^{(1)}(t,x)}{\partial x} x g^{(1)}(t,x) = 0$$

with $V^{(1)}(1,x) = -x$, and

$$g^{(1)}(t,x) = \begin{cases} 0, & x\dfrac{\partial V^{(0)}(t,x)}{\partial x} > 0 \\ 1, & x\dfrac{\partial V^{(0)}(t,x)}{\partial x} < 0 \end{cases}$$

with $V^{(0)}(t,x) = -x$.

In order to obtain quality approximations of $V^{(1)}$ within the domain of interest, we are required to apply our method on an extended domain $\bar{\Omega} \subset \Omega$ to allow for a proper transition layer. We set $\bar{\Omega} = [-2, 2]$. Table 1 shows the obtained results for different discretization settings. The table suggests that the approximations become more refined when the density of the spatial triangulation $\mathcal{T}_x$ is increased. This occurs however only when the resulting ODE (27) is also sufficiently discretized in time.

In Figs. 3(b) and 4(b) the approximation error and residual error, as in (35), is also plotted for two different discretization settings. We note here that the residual error can give a good indication on the quality of the approximation when the exact solution is unknown. For instance, Figs. 3(a) and 4(a) show that the residual error is particularly large at the area surrounding the non-smooth region of the exact solution (45), i.e. at $(t,x) \in |0,1\rangle \times \{0\}$. When referring back to Figs. 3(b) and 4(b), we notice a correlation in which the approximation error peaks at the same region. Although not demonstrated in this paper, we believe that the approximations can be improved locally by increasing the density of the triangulation at regions with large peaks in the residual.

6.2. Example II: an inverse pendulum problem

In the next example, consider a damped inverse pendulum, and let x_1, x_2 denote respectively the angular deviation and angular rate w.r.t. the vertical. We have the dynamics

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ a\sin x_1 + bx_2 + cu \end{bmatrix}$$

where $\boldsymbol{u} \in \mathrm{U} := [-1, 1]$ is a torque input. Suppose that $a = 9.81$, $b = 1$, and $c = 1$. The objective is to bring the pendulum mass to the vertical at the finite end time through the cost criteria

$$r(\boldsymbol{x}, \boldsymbol{u}) = w_1 {x_1}^2 + w_2 u^2, \qquad l(\boldsymbol{x}) = w_3 {x_2}^2, \qquad T = 1.$$

This yields the HJB equation

$$\frac{\partial V(t,\boldsymbol{x})}{\partial t} + \inf_{-1\le u\le 1}\left[\frac{\partial V(t,\boldsymbol{x})}{\partial x_1} x_2 + \frac{\partial V(t,\boldsymbol{x})}{\partial x_1}(a\sin x_1 + bx_2 + cu) + w_1 {x_1}^2 + w_2 u^2\right] = 0 \tag{46}$$

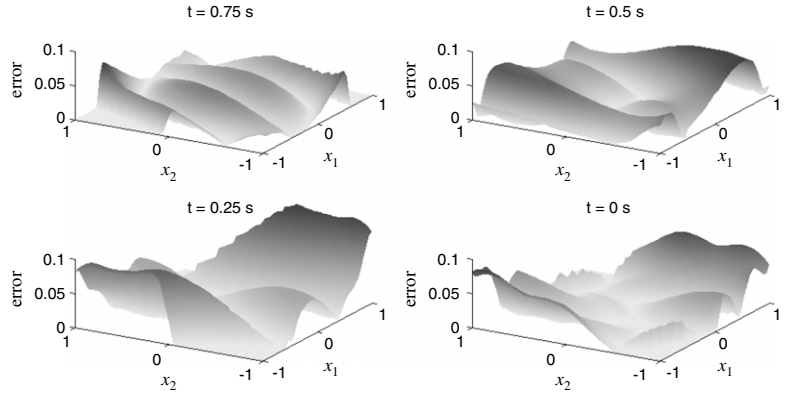
with boundary condition $V(1,\boldsymbol{x}) = w_3 {x_2}^2$. Our interest is to approximate the solution of (46) on $\Omega = [-1, 1] \times [-1, 1]$. We solve the PDE however on the extended domain $\bar{\Omega} = [-2, 2] \times [-4, 4]$. Since f is input affine, r is quadratic in u, and l is continuously differentiable, we deduce that Theorem 1 holds. Applying Procedure 1 yields

$$\frac{\partial V^{(i)}(t,\boldsymbol{x})}{\partial t} + \frac{\partial V^{(i)}(t,\boldsymbol{x})}{\partial x_1} x_2 + \frac{\partial V^{(i)}(t,\boldsymbol{x})}{\partial x_2}\Big(a\sin x_1 + bx_2 + cg^{(i)}(t,\boldsymbol{x})\Big) + w_1 {x_1}^2 + w_2 {g^{(i)}(t,\boldsymbol{x})}^2 = 0$$

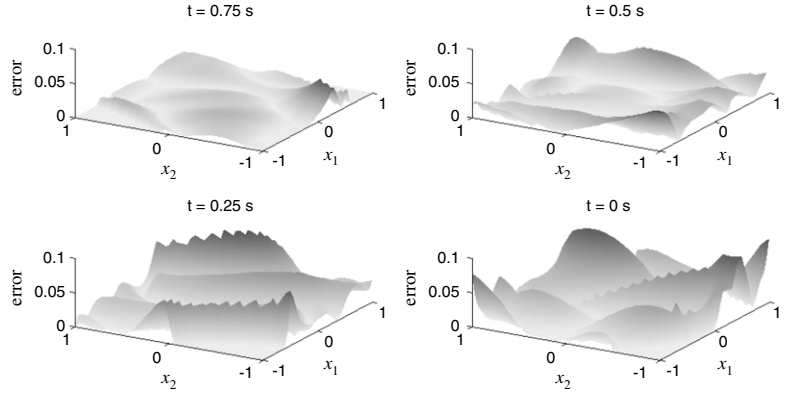
and

$$g^{(i)}(t,\boldsymbol{x}) = \min\left\{1, \max\left\{-1, \frac{-c}{2w_2}\frac{\partial V^{(i-1)}(t,\boldsymbol{x})}{\partial x_2}\right\}\right\}.$$

The analytic solution for this specific example is unknown. We therefore compare our obtained results with a high accuracy level set approximation using the level set toolbox (Mitchell, 2007).

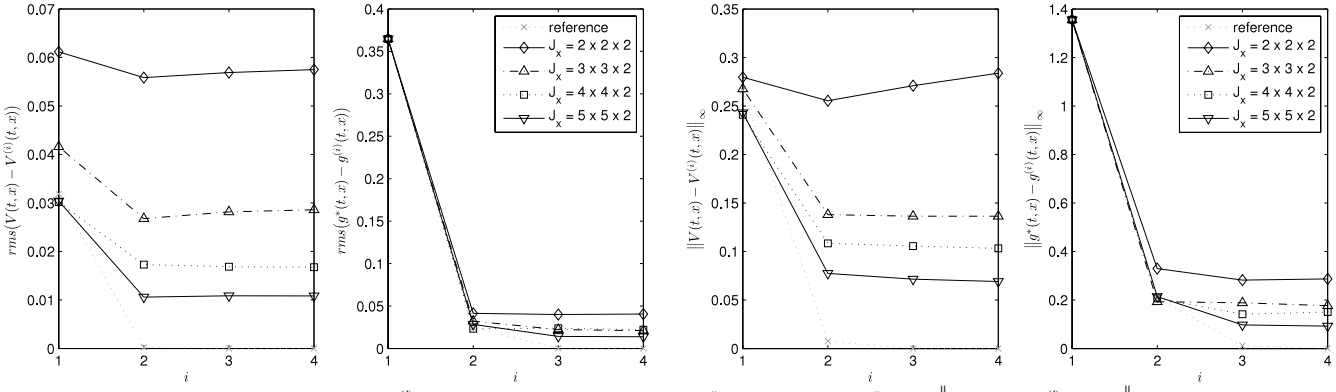


(a) The number of triangulations is set to $J_x = 3 \times 3 \times 2$.



(b) The number of triangulations is set to $J_x = 5 \times 5 \times 2$.

Fig. 5. The 2-norm of the optimal control-law approximation error in the fourth iteration, i.e. $\left\| g^*(t, \boldsymbol{x}) - g^{(4)}_{\text{apprx}}(t, \boldsymbol{x}) \right\|_2$.



(a) $\text{rms}(V(t, \boldsymbol{x}) - V^{(i)}(t, \boldsymbol{x}))$ and $\text{rms}(g^*(t, \boldsymbol{x}) - g^{(i)}_{\text{apprx}}(t, \boldsymbol{x}))$.

(b) $\left\| V(t, \boldsymbol{x}) - V^{(i)}(t, \boldsymbol{x}) \right\|_\infty$ and $\left\| g^*(t, \boldsymbol{x}) - g^{(i)}_{\text{apprx}}(t, \boldsymbol{x}) \right\|_\infty$.

Fig. 6. Results for example II for different discretization settings. Only the number of triangles J_x in $\mathcal{T}_{\boldsymbol{x}}$ is varied, other parameters are fixed to: $J_t = 4$, $d_x = d_t = 4$, $m_x = m_t = 1$.

Fig. 6 shows the obtained results for different discretization settings. Fig. 5 plots the error in the optimal feedback law for the fourth value-iteration step. The results suggest that refinements in the spatial triangulation $\mathcal{T}_{\boldsymbol{x}}$ lead to better approximations of the value function and the optimal feedback control-law.

7. Conclusions and future work

A new collocation method was presented to solve the time-dependent HJB equation through discretizing the PDE with simplex *B*-splines both in state and time. It was shown that, under the framework of a value-iteration procedure, an estimate of the spline coefficients can be obtained through solving a sequence of quadratic programs. Because of domain decomposition, the resulting quadratic programs are shown to be highly sparse, which lead to significant improvement in space-complexity, whereas the time-complexity is at worst as good as any RBF based methods. The correctness of the method was demonstrated by applying the method on two benchmark problems. Numerical results have indicated that, under fixed degree and first-order smoothness conditions, the approximations become progressively more refined when increasing the triangulation density.

Table 1
Results for example I with cubic splines (i.e. $d_x = d_t = 3$) using different type-I triangulation settings.

J_x		10	20	30	40
(a) $\left\|V(t,\boldsymbol{x}) - V^{(1)}_{\text{apprx}}(t,\boldsymbol{x})\right\|_\infty$					
J_t	1	$1.98 \cdot 10^{-1}$	$2.51 \cdot 10^{-1}$	$2.16 \cdot 10^{-1}$	$6.47 \cdot 10^{-2}$
	2	$3.26 \cdot 10^{-2}$	$1.64 \cdot 10^{-2}$	$1.03 \cdot 10^{-2}$	$8.21 \cdot 10^{-3}$
	3	$3.25 \cdot 10^{-2}$	$1.63 \cdot 10^{-2}$	$1.03 \cdot 10^{-2}$	$6.66 \cdot 10^{-3}$
	4	$3.23 \cdot 10^{-2}$	$1.62 \cdot 10^{-2}$	$1.03 \cdot 10^{-2}$	$5.85 \cdot 10^{-3}$
(b) $\text{rms}(V(t,x) - V^{(1)}_{\text{apprx}}(t,x))$					
J_t	1	$6.35 \cdot 10^{-3}$	$5.97 \cdot 10^{-3}$	$3.33 \cdot 10^{-3}$	$3.34 \cdot 10^{-3}$
	2	$3.83 \cdot 10^{-3}$	$1.10 \cdot 10^{-3}$	$5.53 \cdot 10^{-4}$	$3.11 \cdot 10^{-4}$
	3	$3.89 \cdot 10^{-3}$	$1.11 \cdot 10^{-3}$	$5.37 \cdot 10^{-4}$	$2.97 \cdot 10^{-4}$
	4	$3.91 \cdot 10^{-3}$	$1.12 \cdot 10^{-3}$	$5.35 \cdot 10^{-4}$	$2.88 \cdot 10^{-4}$

With the overall goal of developing nonlinear optimal controllers for real-time applications, future work will primarily be geared towards developing customized distributed solvers, so that also problems of higher dimensions can be addressed using the method.

Acknowledgments

We would like to thank J.H. van Schuppen for his support and useful suggestions in writing the paper. Furthermore, the first author would like to thank M. Verhaegen for supporting him as visiting researcher at the Delft Center for Systems and Control.

References

Abu-Khalaf, M., & Lewis, F. L. (2005). Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica*, *41*(5), 779–791.
Alwardi, H., Wang, S., & Jennings, L. S. (2013). An adaptive domain decomposition method for the Hamilton–Jacobi–Bellman equation. *Journal of Global Optimization*, *56*(4), 1361–1373. http://dx.doi.org/10.1007/s10898-012-9850-2.
Alwardi, H., Wang, S., Jennings, L. S., & Richardson, S. (2012). An adaptive least-squares collocation radial basis function method for the HJB equation. *Journal of Global Optimization*, *52*(2), 305–322.
Awanou, G., & Lai, M. J. (2004). Trivariate spline approximations of 3D Navier–Stokes equations. *Mathematics of Computation*, *74*(250), 585–601.
Awanou, G., Lai, M. J., & Wenston, P. (2005). The multivariate spline method for scattered data fitting and numerical solutions of partial differential equations. In G. Chen, & M. J. Lai (Eds.), *Wavelets and splines* (pp. 24–75). Nashburo Press.
Bardi, M., & Capuzzo-Dolcetta, I. (2008). *Optimal control and viscosity solutions of Hamilton–Jacobi–Bellman equations*. Boston: Birkhäuser.
Beard, R. W. (1995). *Improving the closed-loop performance of nonlinear systems*. (Ph.D. thesis). Rensselaer Polytechnic Institute.
Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press.
Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Athena scientific optimization and computation series, Neuro-dynamic programming*. Athena Scientific.
Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, *3*(1), 1–122.
Cheng, T., Lewis, F. L., & Abu-Khalaf, M. (2007). A neural network solution for fixed-final time optimal control of nonlinear systems. *Automatica*, *43*(3), 482–490.
Crandall, M. G., Evans, L. C., & Lions, P. L. (1984). Some properties of viscosity solutions of Hamilton–Jacobi equations. *Transactions of the American Mathematical Society*, *282*(2), 487–502.
de Boor, C. (1987). B-form basics. In G. E. Farin (Ed.), *Geometric modeling: algorithms and new trends*. Society for Industrial and Applied Mathematics.
de Boor, C., & Swartz, B. (1973). Collocation at Gaussian points. *SIAM Journal on Numerical Analysis*, *10*(4), 582–606.
de Visser, C. C., Chu, Q. P., & Mulder, J. A. (2009). A new approach to linear regression with multivariate splines. *Automatica*, *45*(12), 2903–2909.
de Visser, C. C., Chu, Q. P., & Mulder, J. A. (2011). Differential constraints for bounded recursive identification with multivariate splines. *Automatica*, *47*(9), 2059–2066.
de Visser, C. C., & Verhaegen, M. (2013). Wavefront reconstruction in adaptive optics systems using nonlinear multivariate splines. *Journal of Optical Society of America A*, *30*(1), 82–95.
Finlayson, B. A. (1972). *Mathematics in science and engineering series, The method of weighted residuals and variational principles: with application in fluid mechanics, heat and mass transfer*. Acad. Press.
Hanselmann, T., Noakes, L., & Zaknich, A. (2007). Continuous-time adaptive critics. *IEEE Transactions on Neural Networks*, *18*(3), 631–647.
Hu, X., Han, D., & Lai, M. J. (2007). Bivariate splines of various degrees for numerical solution of partial differential equations. *SIAM Journal on Scientific Computing*, *29*(3), 1338–1354.
Huang, C. S., Wang, S., Chen, C. S., & Li, Z. C. (2006). A radial basis collocation method for Hamilton Jacobi Bellman equations. *Automatica*, *42*(12), 2201–2207.
Lai, M. J., & Schumaker, L. L. (2007). *Spline functions on triangulations*. Cambridge University Press.
Mitchell, I. M. (2007). A toolbox of level set methods (version 1.1) UBC CS TR-2007-11.
Osher, S., & Fedkiw, R. (2003). *Level set methods and dynamic implicit surfaces*. New York: Springer-Verlag.
Powell, W. B. (2007). *Approximate dynamic programming: solving the curses of dimensionality*. Wiley-Interscience.
Saridis, G. N., & Lee, C. G. (1979). An approximation theory of optimal control for trainable manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(3), 152–159.
Sutton, R. S., & Barto, A. G. (1998). *Adaptive computation and machine learning, Reinforcement learning: an introduction*. MIT Press.
Vamvoudakis, K. G., & Lewis, F. L. (2009). Online synchronous policy iteration method for optimal control. In *Recent advances in intelligent control systems* (pp. 357–374).
Vamvoudakis, K. G., & Lewis, F. L. (2010). Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, *46*(5), 878–888.
Wang, S., Jennings, L. S., & Teo, K. L. (2003). Numerical solution of Hamilton–Jacobi–Bellman equations by an upwind finite volume method. *Journal of Global Optimization*, [ISSN: 0925-5001] *27*(2–3), 177–192. http://dx.doi.org/10.1023/A:1024980623095.
Wang, F., Zhang, H., & Liu, D. (2009). Adaptive dynamic programming: an introduction. In *IEEE computational intelligence magazine* (pp. 39–47).

Nithin Govindarajan received the B.Sc. and M.Sc. degrees from the Delft University of Technology in 2009 and 2012, respectively. In 2013, he worked as a visiting researcher at the Delft Center for Systems and Control (DCSC). He currently is a Ph.D. student of Mechanical Engineering at the University of California, Santa Barbara. His research interests are optimal control, dynamic programming, multi-agent systems, distributed optimization and Kalman Filtering.

Cornelis. C. de Visser received the M.Sc. degree from the Delft University of Technology in 2007 and 2011. In 2011 he received his Ph.D. degree from the faculty of Aerospace Engineering at the Delft University of Technology in The Netherlands. Currently, he is an assistant professor at the Control & Simulation section of the Delft University of Technology, where he performs research on methods for aircraft system identification, flight envelope prediction, and fault tolerant control. He has (co)authored more than 40 journal and conference papers on topics of aircraft system identification and fault tolerant control.

Kalmanje Krishnakumar received his Ph.D. degree in Electrical and Aerospace Engineering from the University of Alabama in 1989. Currently he is the technical lead for the Autonomous Systems and Robotics Area in the Exploration Technology Directorate at NASA Ames Research Center where he guides the research and development of autonomous systems and their applications to robotics and aerospace systems and vehicles. He has (co)authored more than 100 journal and conference papers ranging from pilot-in-the-loop flight simulation studies, intelligent and adaptive control, robust control, and spacecraft guidance and control to nonlinear optimization for aerospace vehicles. He is an associate fellow of the American Institute of Aeronautics and Astronautics (AIAA).