J Optim Theory Appl (2014) 160:527–552
DOI 10.1007/s10957-013-0403-8

Numerical Solutions to the Bellman Equation of Optimal Control

Cesar O. Aguilar · Arthur J. Krener

Received: 22 June 2012 / Accepted: 15 August 2013 / Published online: 5 September 2013

Abstract In this paper, we present a numerical algorithm to compute high-order approximate solutions to Bellman's dynamic programming equation that arises in the optimal stabilization of discrete-time nonlinear control systems. The method uses a patchy technique to build local Taylor polynomial approximations defined on small domains, which are then patched together to create a piecewise smooth approximation. The numerical domain is dynamically computed as the level sets of the value function are propagated in reverse time under the closed-loop dynamics. The patch domains are constructed such that their radial boundaries are contained in the level sets of the value function and their lateral boundaries are constructed as invariant sets of the closed-loop dynamics. To minimize the computational effort, an adaptive subdivision algorithm is used to determine the number of patches on each level set depending on the relative error in the dynamic programming equation. Numerical tests in 2D and 3D are given to illustrate the accuracy of the method.

Keywords Discrete-time control systems · Nonlinear optimal regulation · Dynamic programming · Hamilton–Jacobi–Bellman equation · Numerical methods

1 Introduction

A major accomplishment in linear control systems theory is the development of stable and reliable numerical algorithms to compute solutions to algebraic Riccati equa-

Communicated by Lars Grüne.

C.O. Aguilar (✉)
Department of Mathematics, California State University, Bakersfield, CA, USA
e-mail: caguilar24@csub.edu

A.J. Krener
Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA, USA
e-mail: ajkrener@nps.edu

tions [1]. Riccati equation solvers are now standard routines in several commercial software packages such as MATLAB® (`care`, `dare`), MAPLE® (`CARE`, `DARE`), and MATHEMATICA® (`RiccatiSolve`, `DiscreteRiccatiSolve`). For nonlinear control systems, the development of analogous algorithms is far less mature than its linear counterparts. The main difficulty, of course, is the complexity of solving the associated Hamilton–Jacobi–Bellman (HJB) partial differential equation in continuous-time and the dynamic programming equation in the discrete-time case. Although a complete mathematical theory of solutions to Hamilton–Jacobi equations has been developed under the notion of *viscosity solution* [2], the lack of stable and reliable numerical methods to solve the HJB and/or the dynamic programming equations, even for low-dimensional systems, is a real obstruction to the application of nonlinear control theory to problems of practical interest.

Various methods have been proposed in the literature for computing numerical solutions to the HJB and the dynamic programming equations [3–13]. The reader is referred to [8, 13] for a discussion on the strengths and weaknesses of these methods and to [14] for a comparison of some of the methods. Recently in [15], a numerical method for an HJB equation, that also incorporates the concept of patchy vector fields, is developed for the minimum-time optimal control problem using an iterative scheme. The original Albrekht method is based on a Taylor series approximation. Loosely speaking, in the Taylor series method [3, 5], it is shown that if the linearized version of the problem is solvable, then the full nonlinear version is also (locally) solvable, and in the real analytic case, it is possible to construct term-by-term a real analytic solution to the HJB equation. One can then use a truncated power series of the solution to obtain a suboptimal control [16–18]. Two main drawbacks with the series method is that the region of attraction of the closed-loop system is not known in advance and may actually shrink as the order of the approximation increases, and the computational effort needed to compute the coefficients of the series solution grows rapidly as the order of the approximation is increased. For these reasons, the Taylor series method was extended in [12] using the general idea of patchy vector fields proposed in [19] and a numerical continuation method based on a Cauchy–Kowalevski technique. Roughly speaking, the method consists in computing Taylor polynomial solutions on disjoint patch domains and then piecing together the different local solutions. The boundaries on adjacent patches are computed as (approximately) invariant manifolds of the closed-loop system, and the remaining boundaries are contained in the level sets of the computed value function. The method can be seen as a finite volume continuation method since new local polynomial solutions are computed by inheriting some of the derivatives from a previously computed local solution and computing the remaining derivatives from the HJB equation using a Cauchy–Kowalevski-type algorithm. The original method in [12] has been improved, and a proof of its high-order accuracy is given in [20].

In this paper, we present a discrete-time version of the patchy method in [12]. A motivation for considering discrete-time systems is that, in practice, real-time controllers are implemented digitally and control design based on discretized models may lead to performance improvements over simply discretizing a continuous-time controller [21]. Moreover, under certain conditions [22, 23], if the sampling period is sufficiently fast, a feedback control asymptotically stabilizing an approximate discretization of a continuous-time model also asymptotically stabilizes the original

continuous model via a sample-hold implementation. Aside from these reasons, the functional form of the dynamic programming equation significantly simplifies the Cauchy–Kowalevski technique needed to compute new solutions from the previously computed solutions on lower cost level sets. The main simplification is that the Taylor coefficients for the optimal cost function can be readily obtained by *evaluation* after solving for the coefficients of the control. By contrast, in the continuous-time patchy method [12], one needs to solve a system of linear equations for the coefficients of the optimal cost function once the coefficients for the control are known. This difference decreases the overall computational effort of the algorithm and makes it more practical in higher-dimensional problems.

This paper is organized as follows. In Sect. 2, we review some background material on the optimal regulator problem and Albrekht's method in the discrete-time setting, and also give a brief description of our patchy algorithm. In Sect. 3, we present a continuation algorithm that will be used to compute a new local polynomial approximation to the solution of the dynamic programming equation from a previously computed local polynomial approximation. In Sect. 4, we give a detailed presentation of our numerical algorithm. In Sect. 5, we describe an adaptive patch subdivision scheme for determining the number of patch domains on which to build new polynomial solutions. In Sect. 6, we outline some implementation details of the method in the 3D case. In Sect. 7, we present some numerical tests illustrating the method. We end the paper with a conclusion and a discussion of research problems that merit further investigation.

2 Preliminaries

We first describe some notation used in the paper. If Q and P are matrices, by $Q \succeq 0$ we mean that Q is positive semi-definite, and by $P \succ 0$ we mean that P is positive definite. The transpose of the matrix A is denoted by A'. The Euclidean norm is denoted $\|\cdot\|$. The interior of a set $S \subset \mathbb{R}^n$ is denoted $\operatorname{int} S$. The set of nonnegative integers is denoted $\mathbb{N}_0 = \{1, 2, 3, \ldots\}$. If $f : \Omega \to \mathbb{R}$ is a continuous function on the compact set Ω, we denote $\|f\|_\infty = \max_{x\in\Omega} |f(x)|$. The complement of a set S is denoted S^c.

Consider the discrete-time nonlinear control system

$$x(k+1) = f\big(x(k), u(k)\big), \tag{1}$$

where $x(k) \in \mathbb{R}^n$ is the state, $u(k) \in \mathbb{R}^m$ is the control, $k \in \mathbb{N}_0$, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are the dynamics, assumed to be C^∞. We assume that $f(0, 0) = 0$, i.e., the origin $x^0 = 0$ is an equilibrium of the uncontrolled system. A fundamental problem in control systems theory is the design of a feedback control that asymptotically stabilizes the equilibrium x^0. Under appropriate conditions [24], the design of such a feedback control can be accomplished by formulating an optimal control problem as follows. Given a *running cost* $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, of class C^∞ and with $\ell(0, 0) = 0$, the *optimal regulator problem* for (1) is to find a feedback control $\kappa : \mathbb{R}^n \to \mathbb{R}^m$ (defined

possibly only locally) such that

$$\pi(x_0) := \min_{u(0),u(1),\ldots} \sum_{k=0}^{\infty} \ell\big(x(k),u(k)\big) = \sum_{k=0}^{\infty} \ell\big(x(k),\kappa\big(x(k)\big)\big)$$

for all $x_0 = x(0)$ for which $\pi(x_0)$ is well defined. If it exists, the *optimal cost function* (or *value function*) π satisfies Bellman's dynamic programming equation [25]

$$\pi(x) = \min_u \big[\pi\big(f(x,u)\big) + \ell(x,u)\big], \tag{2}$$

and the *optimal feedback control* κ is given by

$$\kappa(x) = \arg\min_u \big[\pi\big(f(x,u)\big) + \ell(x,u)\big].$$

Consequently,

$$\pi(x) = \pi\big(f\big(x,\kappa(x)\big)\big) + \ell\big(x,\kappa(x)\big), \tag{3}$$

and moreover, the resulting closed-loop system $x(k+1) = f(x(k),\kappa(x(k)))$ has x^0 as a locally asymptotically stable equilibrium with Lyapunov function π [24]. Hence, the problem of designing an asymptotically stabilizing control for (1) becomes that of solving (3) for the pair (π,κ). Regarding the existence of π and κ, the following theorem proved in [26] is suitable for our purposes.

Theorem 2.1 *Suppose that f and ℓ are C^∞. Let $A = \frac{\partial f}{\partial x}(0,0)$, let $B = \frac{\partial f}{\partial u}(0,0)$, let $Q = \frac{\partial^2 \ell}{\partial x^2}(0,0)$, let $R = \frac{\partial^2 \ell}{\partial u^2}(0,0) \succ 0$, let $S = \frac{\partial^2 \ell}{\partial x \partial u}(0,0)$, and assume that $\left[\begin{smallmatrix} Q & S \\ S' & R \end{smallmatrix}\right] \succeq 0$. Assume that (A,B) is stabilizable and $(A,Q^{1/2})$ is detectable. Then there exist C^∞ mappings (π,κ) solving (3) locally around the origin $x^0 = 0$.*

Henceforth, when not explicitly stated, we make the assumptions in Theorem 2.1. Moreover, we make the further assumption that the mapping $u \mapsto \pi(f(x,u)) + \ell(x,u)$ is strictly convex for all $x \in \mathbb{R}^n$ for which π is defined, and consequently the following first-order condition for a minimum is satisfied:

$$0 = \frac{\partial \pi}{\partial x} f\big(x,\kappa(x)\big) \frac{\partial f}{\partial u}\big(x,\kappa(x)\big) + \frac{\partial \ell}{\partial u}\big(x,\kappa(x)\big). \tag{4}$$

In general, obtaining explicit solutions to (3)–(4) for (π,κ) is unrealizable, and thus one is led to consider numerical methods. A natural approach to compute approximate solutions to (π,κ) is to consider Taylor series approximation methods, as done in [26] and which we now summarize. To begin, it is well known that when $f(x,u) = Ax + Bu$ and $\ell(x,u) = \frac{1}{2}x'Qx + \frac{1}{2}u'Ru$, with $Q \succeq 0$ and $R \succ 0$, there exists a unique symmetric matrix $P \succ 0$ such that $\pi(x) = \frac{1}{2}x'Px$ and $\kappa(x) = Kx$, where $K = -(B'PB + R)^{-1}B'PA$, provided that (A,B) is stabilizable and $(A,Q^{1/2})$ is detectable [27]. Moreover, the closed-loop matrix $A + BK$ has eigenvalues inside the unit circle in $\mathbb{C}$, and thus the closed-loop system is (globally) asymptotically stable. Suppose now that f and ℓ are nonlinear and have Taylor

expansions of the form

$$f(x,u) = Ax + Bu + \sum_{k=2}^{\infty} f^{[k]}(x,u),$$

$$\ell(x,u) = \frac{1}{2}x'Qx + \frac{1}{2}u'Ru + \sum_{k=3}^{\infty} \ell^{[k]}(x,u),$$

where $f^{[k]}(x,u)$ are the k-order terms of f, and similarly for $\ell^{[k]}(x,u)$. In analogy with the method of Albrekht [3] in the continuous-time case, the Taylor series coefficients of (π,κ) are computed by Taylor expanding (3)–(4) about x^0 and gathering terms of the same order. The Taylor expansions of π and κ take the form

$$\pi(x) = \frac{1}{2}x'Px + \sum_{k=3}^{\infty} \pi^{[k]}(x), \qquad \kappa(x) = Kx + \sum_{k=2}^{\infty} \kappa^{[k]}(x),$$

respectively. When computing for the Taylor coefficients, (3) is used to compute the $(d+1)$-order coefficients of π, and (4) is used to compute the d-order coefficients of κ, where $d \geq 1$ is a positive integer. After computing the first nonzero coefficients of π and κ, the subsequent equations for the higher-order coefficients are *linear* and have a triangular structure due to the fact that the equations for the $(d+1)$-order terms of π do not involve the d-order terms of κ. Hence, once the $(d+1)$-order terms for π are solved for, the d-order terms of κ can be directly computed. There is, however, an obstruction to computing the $(d+1)$-order terms of π, and it relies on the spectrum of the closed-loop matrix $A+BK$. Specifically, when solving for the Taylor coefficients of π of order $d \geq 3$, one is led to the invertibility of the linear transformation

$$p(x) \mapsto p(x) - p\big((A+BK)x\big), \tag{5}$$

where $p(x)$ is an $\mathbb{R}$-valued d-homogeneous polynomial, i.e., $p(\alpha x) = \alpha^d p(x)$ for all $x \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$. The eigenvalues of the linear transformation (5) are of the form $1 - \lambda_{i_1}\lambda_{i_2}\cdots\lambda_{i_d}$ where the λ_i are eigenvalues of the closed-loop matrix $A+BK$. Now, because $|\lambda_i| < 1$, we have that $1 - \lambda_{i_1}\lambda_{i_2}\cdots\lambda_{i_d} \neq 0$, and thus the transformation is invertible. Hence, as a consequence of the linear regulator problem, one is able to solve for the Taylor coefficients of π and κ to any desired order.

Let π^0 and κ^0 denote the Taylor polynomial approximations of π and κ to degree $d+1$ and d, respectively, based at x^0. From now on, the pair (π^0,κ^0) will be called the *Albrekht approximation of* (π,κ) *of order* d. As stated in the introduction, increasing the order d results in a more accurate approximation, but there are two main drawbacks in doing so. First, increasing d may decrease the size of the domain on which (π^0,κ^0) is a satisfactory approximation because of the rapidly growing behavior of high-order polynomials away from the origin. Second, the number of Taylor coefficients of degree d in n variables is $\binom{n+d-1}{d} = \frac{(n+d-1)!}{d!(n-1)!}$, and this number grows rapidly in d and thereby increases the computational effort substantially.

In this paper, we present a numerical algorithm that extends the approximation (π^0, κ^0) and produces a piecewise smooth approximation by patching together local polynomial approximations on disjoint domains. We now give a brief description of the method. The Albrekht approximation (π^0, κ^0) is accepted on a sublevel set $\Omega^0 = \{x \in \mathbb{R}^n : \pi^0(x) \leq c\}$, with c chosen sufficiently small so that Ω^0 is topologically equivalent to the closed unit disk $\mathbb{D}^n$ in $\mathbb{R}^n$. We then partition the boundary of Ω^0 into patches, select *patch points* inside each patch, and compute new polynomial approximations to (π, κ) (based at the patch points) by using (3)–(4) and the derivatives of (π^0, κ^0) at the patch points. Each new local approximation is accepted on a domain, called a *patch tube*, radiating outward from the boundary of Ω^0. The patch tubes of the new local approximations are pairwise disjoint, and their outer-most boundaries define a piecewise smooth hypersurface. We then repeat the procedure on the newly computed outer-most boundaries of the patch tubes. We can choose to refine the partition on the outer-most boundaries depending on how well the computed approximations are satisfying the dynamic programming equation, say using the relative error incurred by the computed solutions.

3 Continuation Algorithm for the Computation of New Local Solutions

In this section, we present a continuation algorithm that is used repeatedly for computing new approximations from previously computed ones. It is the analog of the Cauchy–Kowalevski-type algorithm in [12]. We will explain the algorithm for the case where we compute a new polynomial approximation to (π, κ) given that we have computed the Albrekht approximation (π^0, κ^0), but the same algorithm applies when computing solutions on patches contained in higher level sets of the value function (Sect. 4). We assume that (π^0, κ^0) is an approximation of degree $d \geq 1$.

The polynomial $\pi^0(x)$ begins with the quadratic term $\frac{1}{2}x'Px$ where $P \succ 0$, and therefore π^0 has the origin as a nondegenerate local minimum. Hence, by Morse's lemma, the sublevel sets $\{x : \pi^0(x) \leq c_1\}$ near the origin are diffeomorphic to the unit disc $\mathbb{D}^n$, provided that $c_1 > 0$ is sufficiently small, and therefore the level sets $\{x : \pi^0(x) = c \leq c_1\}$ near the origin are diffeomorphic to the sphere $\mathbb{S}^{n-1}$. Hence, we assume that $c_1 > 0$ is such that $\Omega^0 := \{x \in \mathbb{R}^n : \pi^0(x) \leq c_1\} \cong \mathbb{D}^n$, and we let $\mathcal{S}^1 := \partial\Omega^0 \cong \mathbb{S}^{n-1}$. By making c_1 smaller if necessary, we can assume that Ω^0 is contained in the domain of attraction of the asymptotically stable closed-loop dynamics resulting by applying the control κ^0. Furthermore, π^0 is a Lyapunov function for the closed-loop dynamics, that is, $\pi^0(f(x, \kappa^0(x))) - \pi^0(x) < 0$ for $x \in \Omega^0$. In particular, for all $x \in \mathcal{S}^1$, we have $\pi^0(f(x, \kappa^0(x))) < \pi^0(x) = c_1$, and consequently $f(x, \kappa^0(x))$ will lie in the interior of Ω^0 for all $x \in \mathcal{S}^1$.

Now let $x^1 \in \mathcal{S}^1$, and we seek to extend (π^0, κ^0) to a new polynomial approximation (π^1, κ^1) centered at x^1 and whose domain radiates outward from the boundary $\mathcal{S}^1$. To this end, we solve the single-stage optimization problem

$$\pi^*(x) := \min_u \big[\pi^0\big(f(x,u)\big) + \ell(x,u)\big] \tag{6}$$

for x near x^1. The interpretation of this new optimization problem is clear. Indeed, since π^0 approximates the optimal cost function π, we replace π with π^0 on the

right-hand side of (2) and thus obtain (6). We then seek a minimizer κ^* satisfying

$$\pi^*(x) = \pi^0\big(f\big(x,\kappa^*(x)\big)\big) + \ell\big(x,\kappa^*(x)\big) \tag{7}$$

and the first-order necessary condition for a minimum

$$0 = \frac{\partial \pi^0}{\partial x} f\big(x,\kappa^*(x)\big)\frac{\partial f}{\partial u}\big(x,\kappa^*(x)\big) + \frac{\partial \ell}{\partial u}\big(x,\kappa^*(x)\big) \tag{8}$$

for x near x^1. Then, we let (π^1,κ^1) be the Taylor approximation of (π^*,κ^*) at x^1. A major advantage in this formulation of extending (π^0,κ^0) is that the computation of π^* and κ^* are decoupled. Indeed, notice that in (8) the only unknown is κ^*, whereas in the original necessary condition (4), both the optimal cost π and optimal regulator κ are unknown. Moreover, once κ^1 is computed, we can compute π^1 by evaluating the right-hand side of (7) using κ^1 and reading off the Taylor coefficients. This is in contrast to the continuous-time version of the patchy algorithm [12], which requires solving a system of linear equations for each Taylor coefficient of π^1. Now, regarding the solvability of (8), in the following theorem, we use continuity arguments to establish the existence of a unique and C^∞ solution κ^* to (8) and describe how its Taylor series can be computed term-by-term.

Theorem 3.1 *Assume that f and ℓ are C^∞ and that $\frac{\partial \ell}{\partial u}(0,0) = 0$. Let π^0 denote the Albrekht approximation of π of order ≥ 2. There exists a unique and C^∞ mapping κ^* solving (8) in a neighborhood $\Omega^* \subset \mathbb{R}^n$ of the origin. Moreover, the Taylor coefficients of κ^* at any $x^* \in \Omega^*$ can be computed term-by-term from (8).*

Proof Define the C^∞ mapping $\psi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ by

$$\psi(x,u) = \frac{\partial \pi^0}{\partial x}\big(f(x,u)\big)\frac{\partial f}{\partial u}(x,u) + \frac{\partial \ell}{\partial u}(x,u).$$

From $f(0,0) = 0$, $\frac{\partial \pi^0}{\partial x}(0) = 0$, and $\frac{\partial \ell}{\partial u}(0,0) = 0$ it follows that $\psi(0,0) = 0$. Next, using the fact that $\frac{\partial^2 \pi^0}{\partial x^2}(0) = P \succ 0$ and $\frac{\partial^2 \ell}{\partial u^2}(0,0) = R \succ 0$, it follows that $\frac{\partial \psi}{\partial u}(0,0) = B'PB + R$ is positive definite and therefore $\frac{\partial \psi}{\partial u}(0,0)$ is invertible. By the implicit function theorem, there exists a neighborhood $\Omega^* \subset \mathbb{R}^n$ of the origin and a unique and C^∞ mapping $\kappa^* : \Omega^* \to \mathbb{R}^m$ such that $\psi(x,\kappa^*(x)) = 0$ for all $x \in \Omega^*$. In other words,

$$0 = \frac{\partial \pi^0}{\partial x}\big(f\big(x,\kappa^*(x)\big)\big)\frac{\partial f}{\partial u}\big(x,\kappa^*(x)\big) + \frac{\partial \ell}{\partial u}\big(x,\kappa^*(x)\big),$$

and this proves the first claim. By continuity, we can assume that $\frac{\partial \psi}{\partial u}(x,\kappa^*(x))$ is positive definite, and therefore invertible, for all $x \in \Omega^*$. For convenience, let $M(x,u) = \frac{\partial \psi}{\partial u}(x,u)$.

Now fix $x^* \in \Omega^*$ and consider the computation of the derivatives of κ^* at x^*. To this end, write (8) in the component form

$$0 = \sum_a \frac{\partial \pi^0}{\partial x_a}\frac{\partial f_a}{\partial u_\alpha} + \frac{\partial \ell}{\partial u_\alpha}$$

for $\alpha = 1, \ldots, m$, and then apply $\frac{\partial}{\partial x_j}$ yielding

$$0 = \sum_{\beta} M_{\alpha\beta} \frac{\partial \kappa_\beta^*}{\partial x_j} + \sum_{a,b} \frac{\partial^2 \pi^0}{\partial x_a \partial x_b} \frac{\partial f_b}{\partial x_j} \frac{\partial f_a}{\partial u_\alpha} + \sum_{a} \frac{\partial \pi^0}{\partial x_a} \frac{\partial^2 f_a}{\partial u_\alpha \partial x_j} + \frac{\partial^2 \ell}{\partial u_\alpha \partial x_j}. \tag{9}$$

The number of unknowns $\frac{\partial \kappa_\beta^*}{\partial x_j}(x^*)$ is mn, and (9) produces mn equations. Now because $M(x^*, \kappa^*(x^*))$ is invertible, we can solve uniquely for the unknowns $\frac{\partial \kappa_\beta^*}{\partial x_j}(x^*)$ from (9).

Next, to get equations for $\frac{\partial^2 \kappa_\beta^*}{\partial x_j \partial x_k}(x^*)$, we apply $\frac{\partial}{\partial x_k}$ to (9) yielding (summation convention is in use)

$$\begin{aligned}
0 = {} & M_{\alpha\beta} \frac{\partial^2 \kappa_\beta^*}{\partial x_j \partial x_k} + \left[\frac{\partial M_{\alpha\beta}}{\partial x_k} + \frac{\partial M_{\alpha\beta}}{\partial u_\rho} \frac{\partial \kappa_\rho^*}{\partial x_k} \right] \frac{\partial \kappa_\beta^*}{\partial x_j} \\
& + \frac{\partial^3 \pi^0}{\partial x_a \partial x_b \partial x_c} \left[\frac{\partial f_c}{\partial x_k} + \frac{\partial f_c}{\partial u_\beta} \frac{\partial \kappa_\beta^*}{\partial x_k} \right] \frac{\partial f_b}{\partial x_j} \frac{\partial f_a}{\partial u_\alpha} \\
& + \frac{\partial^2 \pi^0}{\partial x_a \partial x_b} \left[\frac{\partial^2 f_b}{\partial x_j \partial x_k} + \frac{\partial^2 f_b}{\partial x_j \partial u_\beta} \frac{\partial \kappa_\beta^*}{\partial x_k} \right] \frac{\partial f_a}{\partial u_\alpha} \\
& + \frac{\partial^2 \pi^0}{\partial x_a \partial x_b} \frac{\partial f_b}{\partial x_j} \left[\frac{\partial^2 f_a}{\partial u_\alpha \partial x_k} + \frac{\partial^2 f_a}{\partial u_\alpha \partial u_\beta} \frac{\partial \kappa_\beta^*}{\partial x_k} \right] \\
& + \frac{\partial^2 \pi^0}{\partial x_a \partial x_b} \left[\frac{\partial f_b}{\partial x_k} + \frac{\partial f_b}{\partial u_\beta} \frac{\partial \kappa_\beta^*}{\partial x_k} \right] \frac{\partial^2 f_a}{\partial u_\alpha \partial x_j} \\
& + \frac{\partial \pi^0}{\partial x_a} \left[\frac{\partial^3 f_a}{\partial u_\alpha \partial x_j \partial x_k} + \frac{\partial^3 f_a}{\partial u_\alpha \partial x_j \partial u_\beta} \frac{\partial \kappa_\beta^*}{\partial x_k} \right] \\
& + \frac{\partial^3 \ell}{\partial u_\alpha \partial x_j \partial x_k} + \frac{\partial^3 \ell}{\partial u_\alpha \partial x_j \partial u_\beta} \frac{\partial \kappa_\beta^*}{\partial x_k},
\end{aligned}$$

which can be written as

$$0 = \sum_{\beta} M_{\alpha\beta} \frac{\partial^2 \kappa_\beta^*}{\partial x_j \partial x_k} + T_{jk}, \tag{10}$$

where T_{jk} is an expression involving the derivatives of π^0 to degree 3 and the derivatives of κ^* to degree 1. The number of unknowns $\frac{\partial^2 \kappa_\beta^*}{\partial x_j \partial x_k}(x^*)$ is $m\frac{n(n+1)}{2}$, and (10) produces $m\frac{n(n+1)}{2}$ equations. From the invertibility of $M(x^*, \kappa^*(x^*))$, we can solve uniquely for the unknowns $\frac{\partial^2 \kappa_\beta^*}{\partial x_j \partial x_k}(x^*)$ from (10).

By induction, to compute the higher-order derivatives $\frac{\partial^N \kappa_\beta^*}{\partial x_I}(x^*)$, where $I = (i_1, i_2, \ldots, i_N)$ is a multi-index with $i_a \in \{1, 2, \ldots, n\}$ and we use the notation

$\frac{\partial^N \kappa^*_\beta}{\partial x_I} = \frac{\partial^N \kappa^*_\beta}{\partial x_{i_1} \cdots \partial x_{i_N}}$, we apply $\frac{\partial^N}{\partial x_I}$ to (8) and obtain by induction an expression of the form

$$0 = \sum_\beta M_{\alpha\beta} \frac{\partial^N \kappa^*_\beta}{\partial x_I} + T_I, \tag{11}$$

where T_I is an expression involving the derivatives of π^0 to degree $\leq N+1$ and the derivatives of κ^* to degree $\leq N-1$. The number of unknowns $\frac{\partial^N \kappa^*_\beta}{\partial x_I}(x^*)$ is $m\binom{n+N-1}{N}$, and (11) produces the same number of equations. Invertibility of $M(x^*, \kappa^*(x^*))$ implies that we can solve uniquely for $\frac{\partial^N \kappa^*_\beta}{\partial x_I}(x^*)$ from (11). This completes the proof. □

Provided that c_1 is sufficiently small, it will be possible to use the algorithm in the proof of Theorem 3.1 to compute a Taylor approximation κ^1 to κ^* at $x^1 \in \mathcal{S}^1$. In practice, κ^1 is computed to the same degree $d \geq 1$ as the Albrekht approximation κ^0. Then, having computed κ^1, we can compute a Taylor approximation π^1 to π^* at x^1 by simply differentiating the right-hand side of (7) and evaluating the resulting expression at x^1. For consistency with the order of π^0, equal to $d+1$, the approximation π^1 is also computed to degree $d+1$. By inspection we observe from (7) that by the chain rule the $(d+1)$th derivative of the right-hand-side of (7) depends on the $(d+1)$th derivative of κ^*, and the latter has not been computed. It turns out, however, that computing the $(d+1)$th derivative of κ^* is unnecessary.

Lemma 3.1 *Let π^* be defined as in (7), where κ^* satisfies (8) on Ω^*. Then the N-order derivatives of π^* on Ω^* depend on at most the $(N-1)$-order derivatives of κ^* on Ω^*.*

Proof Using (7), it is straightforward to show by induction that for any $N \geq 1$

$$\frac{\partial^N \pi^*}{\partial x_I} = S_I + \sum_{\alpha,a} \left(\frac{\partial \pi^0}{\partial x_a} \frac{\partial f_a}{\partial u_\alpha} + \frac{\partial \ell}{\partial u_\alpha} \right) \frac{\partial^N \kappa^*_\alpha}{\partial x_I} \tag{12}$$

where $I = (i_1, \ldots, i_N)$ is a multi-index, and where S_I is an expression involving the derivatives of π^0 to degree N and the derivatives of κ^* to degree $N-1$. Now, from (8) we observe that the coefficient of the N-order derivatives of κ^* in (12) vanish on Ω^*. Therefore, on Ω^* the Nth derivatives of π^* depend only on the $N-1$ derivatives of κ^*, and this completes the proof. □

Applying Lemma 3.1 with $N = d+1$, it follows that in order to compute the $d+1$ derivatives of π^1, we need only know the derivatives of κ^1 to degree d.

Having computed (π^1, κ^1), we can extend the initial approximation (π^0, κ^0) by constructing a patch tube on which (π^1, κ^1) will be accepted and adjoining it to Ω^0. This entire process can be repeated at distinct points on $\mathcal{S}^1$ in such a way that the initial domain Ω^0 is covered by the patch tubes of the newly computed approximations. In the next section, we give the details of how to construct the patch tubes.

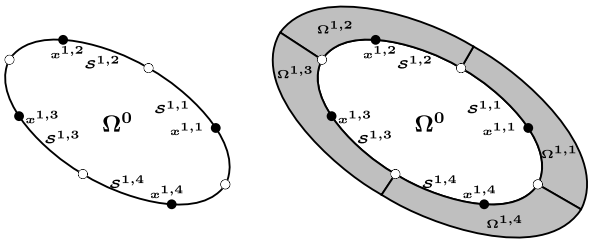

Fig. 1 Dynamic construction of patch domains; (**a**) partition of the boundary $\mathcal{S}^1 = \partial\Omega^0$, (**b**) construction of the patch tubes $\Omega^{1,j}$

4 Extending the Albrekht Approximation

4.1 Level One Extension

In this section, we describe how to extend the Albrekht approximation (π^0, κ^0) defined on Ω^0 to a larger domain $\Omega^0 \cup \Omega^1$, where Ω^1 radiates outward from the boundary $\mathcal{S}^1 = \partial\Omega^0$ and surrounds Ω^0 in the sense that $\Omega^0 \cap \Omega^1 = \partial\Omega^0$.

Let $\mathcal{S}^{1,1}, \ldots, \mathcal{S}^{1,p_1}$ be a partition of the boundary $\mathcal{S}^1$ such that each $\mathcal{S}^{1,j}$ has a nonempty interior relative to the subspace topology on $\mathcal{S}^1 \subset \mathbb{R}^n$. The algorithm described in Sect. 3 is executed at distinct points $x^{1,j} \in \mathcal{S}^{1,j} \cap (\partial\mathcal{S}^{1,j})^c$ for $j = 1, \ldots, p_1$, resulting in polynomial approximations $(\pi^{1,1}, \kappa^{1,1}), \ldots, (\pi^{1,p_1}, \kappa^{1,p_1})$ centered at $x^{1,1}, \ldots, x^{1,p_1}$, respectively. We call the points $x^{1,j}$ *patch points*. Figure 1a illustrates the setup for a system in $\mathbb{R}^2$ and $p_1 = 4$, and in which the white dots represent the boundaries of $\mathcal{S}^{1,j}$ and the black dots represent the patch points $x^{1,j}$.

We now describe how to construct the patch tubes $\Omega^{1,j}$ in such a way that their adjacent boundaries are closed-loop invariant. In words, $\Omega^{1,j}$ will be the union of $\mathcal{S}^{1,j}$, lateral boundaries radiating from $\mathcal{S}^{1,j}$, and an outer-most boundary contained in the level set $\{x : \pi^{1,j}(x) = c_2\}$, where $c_2 > c_1$. For each $j \in \{1, \ldots, p_1\}$, let $f^{1,j}(x) := f(x, \kappa^{1,j}(x))$, and for $z \in \mathcal{S}^{1,j}$, let

$$v^{1,j}(z) := \frac{z - f^{1,j}(z)}{\|z - f^{1,j}(z)\|},$$

that is, $-v^{1,j}(z)$ is the direction vector from z to its image under the closed-loop dynamics $f^{1,j}$. To build the domain $\Omega^{1,j}$, we begin with $z \in \mathcal{S}^{1,j}$ and follow the curve

$$t \mapsto x(t, z) := \big(f^{1,j}\big)^{-1}\big(f^{1,j}(z) + v^{1,j}(z)t\big)$$

in positive time until reaching the level set $\{x : \pi^{1,j}(x) = c_2\}$. In other words,

$$\Omega^{1,j} = \bigcup_{z \in \mathcal{S}^{1,j}} \big\{x(t, z) : \pi^{1,j}\big(x(t, z)\big) \le c_2,\ t \in [0, t_z]\big\},$$

where $t_z = \min\{t > 0 : \pi^{1,j}(x(t, z)) = c_2\}$. Now let

$$\mathcal{S}^2 = \bigcup_{j=1}^{p_1} \big\{x(t_z; z) : z \in \mathcal{S}^{1,j}\big\}.$$

By construction, $\Omega^{1,j} \cap \mathcal{S}^1 = \mathcal{S}^{1,j}$, and thus we define $\mathcal{S}^{1,j}$ as the *inner boundary* of $\Omega^{1,j}$, define $\Omega^{1,j} \cap \mathcal{S}^2$ as its *outer boundary*, and define the remaining boundary as its *lateral boundary*. The construction of the domains $\Omega^{1,j}$ defined above is illustrated in Fig. 1b for a system on the plane and $p_1 = 4$.

Remark 4.1 In practice, the lateral boundaries between patches $\Omega^{1,j}$ will not generally match nor will $\mathcal{S}^2$ be a smooth hypersurface. Hence, it will in general be necessary to redefine the patch tubes $\Omega^{1,j}$ to avoid overlaps between adjacent patches.

Remark 4.2 The computation of the curve $t \mapsto x(t,z) = (f^{1,j})^{-1}(f^{1,j}(z) + v^{1,j}(z)t)$ is relatively straightforward due to the fact that it satisfies an ODE. Indeed, it is straightforward to show that

$$\frac{\partial x}{\partial t}(t,z) = \big(\mathbf{D}f^{1,j}\big(x(t,z)\big)\big)^{-1} v^{1,j}(z),$$

and thus we can compute the curve $t \mapsto x(t,z)$ using standard high-order numerical ODE solvers that require only *evaluations* of the mapping $x \mapsto (\mathbf{D}f^{1,j}(x))^{-1}v^{1,j}(z)$, such as Runge–Kutta methods. For example, up to first order $x(t,z) \approx z + (\mathbf{D}f^{1,j}(z))^{-1}v^{1,j}(z)t$, where

$$\mathbf{D}f^{1,j}(z) = \frac{\partial f}{\partial x}\big(z, \kappa^{1,j}(z)\big) + \frac{\partial f}{\partial u}\big(z, \kappa^{1,j}(z)\big)\frac{\partial \kappa^{1,j}}{\partial x}(z)$$

can be easily computed.

We now augment to the original polynomial approximation (π^0, κ^0) defined on Ω^0 the domains $\Omega^{1,j}$ and the corresponding approximations $(\pi^{1,j}, \kappa^{1,j})$ for $j = 1, \ldots, p_1$, thereby obtaining a piecewise smooth approximation to π and κ defined on $\Omega^0 \cup \Omega^1$, where $\Omega^1 := \bigcup_{j=1}^{p_1} \Omega^{1,j}$. In the next section, we use an inductive argument to describe how to further extend the current approximation beyond $\Omega^0 \cup \Omega^1$.

4.2 Level Two and Beyond Extensions

Suppose that the initial polynomial approximation (π^0, κ^0) defined on Ω^0 has been extended to the union $\Omega^0 \cup \Omega^1 \cup \cdots \cup \Omega^r$, $r \geq 1$, and we wish to extend it further in the radial direction from the outer boundary $\mathcal{S}^{r+1} = \partial\Omega^r$. We recall that the domains Ω^i for $i = 1, \ldots, r$ are the unions of patches $\Omega^{i,j}$, $j = 1, \ldots, p_i$, with $p_i \leq p_{i+1}$. In what follows, for notational consistency, we define $\Omega^{0,1} := \Omega^0$, $\kappa^{0,1} := \kappa^0$, $\pi^{0,1} := \pi^0$, $p_0 = 1$, and $c_0 = 0$.

We begin by partitioning $\mathcal{S}^{r+1}$ into sets $\mathcal{S}^{r+1,1}, \ldots, \mathcal{S}^{r+1,p_{r+1}}$ and choose patch points $x^{r+1,j} \in \mathcal{S}^{r+1,j}$ not on the boundary of $\mathcal{S}^{r+1,j}$. More precisely, the sets $\mathcal{S}^{r+1,j}$ are the result of partitioning the outer boundaries of $\Omega^{r,1}, \ldots, \Omega^{r,p_r}$ so that each $\mathcal{S}^{r+1,j} \subset \Omega^{r,\sigma_j}$ for some unique $\sigma_j \in \{1, 2, \ldots, p_r\}$. For example, a trivial partition of $\mathcal{S}^{r+1}$ would involve taking the outer boundaries of $\Omega^{r,1}, \ldots, \Omega^{r,p_r}$ to serve as the $\mathcal{S}^{r+1,1}, \ldots, \mathcal{S}^{r+1,p_{r+1}}$, that is, $\mathcal{S}^{r,j} = \mathcal{S}^{r+1} \cap \Omega^{r,j}$, and thus $p_{r+1} = p_r$. In Sect. 5, we describe an adaptive subdivision method for partitioning $\mathcal{S}^{r+1}$ that takes into account the error of the numerically computed solutions. We now make the following assumption.

Assumption 4.1 *For $j = 1, \ldots, p_{r+1}$, the image of $\mathcal{S}^{r+1,j}$ under the corresponding closed-loop dynamics*

$$x(k+1) = f\big(x(k), \kappa^{r,\sigma_j}(x)\big)$$

is contained in the interior of $\Omega^0 \cup \Omega^1 \cup \cdots \cup \Omega^r$.

By Assumption 4.1, for each $x^{r+1,j} \in \mathcal{S}^{r+1,j}$, there exists a unique patch $\Omega^{\alpha_j,\beta_j}$, with $0 \le \alpha_j \le r$ and $1 \le \beta_j \le p_{\alpha_j}$, such that $y^{r+1,j} = f(x^{r+1,j}, \kappa^{r,\sigma_j}(x^{r+1,j})) \in \Omega^{\alpha_j,\beta_j}$ and $y^{r+1,j}$ does not lie in the outer boundary of $\Omega^{\alpha_j,\beta_j}$. Therefore, as in the level one extension, to compute a new polynomial approximation $(\pi^{r+1,j}, \kappa^{r+1,j})$ centered at $x^{r+1,j}$, we consider the single-stage optimization problem

$$\pi^*(x) := \min_u \big[\pi^{\alpha_j,\beta_j}\big(f(x,u)\big) + \ell(x,u)\big]$$

for x near $x^{r+1,j}$. We therefore seek a new pair (π^*, κ^*) satisfying the equations

$$\pi^*(x) = \pi^{\alpha_j,\beta_j}\big(f\big(x,\kappa^*(x)\big)\big) + \ell\big(x,\kappa^*(x)\big), \tag{13a}$$

$$0 = \frac{\partial \pi^{\alpha_j,\beta_j}}{\partial x}\big(f\big(x,\kappa^*(x)\big)\big)\frac{\partial f}{\partial u}\big(x,\kappa^*(x)\big) + \frac{\partial \ell}{\partial u}\big(x,\kappa^*(x)\big) \tag{13b}$$

for x near $x^{r+1,j}$. As in the level one extension, the computation of π^* and κ^* is decoupled, and once κ^* is known, one can then compute π^* from (13a). We use (13a)–(13b) and the algorithm in Sect. 3 to compute polynomial approximations $(\pi^{r+1,j}, \kappa^{r+1,j})$ to (π^*, κ^*) of degrees $d+1$ and d, respectively, at $x^{r+1,j}$. In Sect. 4.2.1 we consider the solvability of (13b).

We now construct domains $\Omega^{r+1,j}$ for each new solution $(\pi^{r+1,j}, \kappa^{r+1,j})$. To this end, define the closed-loop system $f^{r+1,j}(x) := f(x, \kappa^{r+1,j}(x))$. We begin with $z \in \mathcal{S}^{r+1,j}$ and follow the curve $x(t,z) = (f^{r+1,j})^{-1}(f^{r+1,j}(z) + tv^{r+1,j}(z))$ in positive time until reaching the level set $\{x : \pi^{r+1,j}(x) = c_{r+1}\}$, where $c_{r+1} > c_r$. In other words,

$$\Omega^{r+1,j} = \bigcup_{z \in \mathcal{S}^{r+1,j}} \big\{x(t,z) : \pi^{r+1,j}\big(x(t,z)\big) \le c_{r+1},\ t \in [0,t_z]\big\},$$

where $t_z = \min\{t > 0 : \pi^{r+1,j}(x(t,z)) = c_{r+1}\}$, and we let

$$\mathcal{S}^{r+2} = \bigcup_{j=1}^{p_{r+1}} \big\{x(t_z;z) : z \in \mathcal{S}^{r+1,j}\big\}.$$

Setting $\Omega^{r+1} := \bigcup_{j=1}^{p_{r+1}} \Omega^{r+1,j}$, we can augment to the running approximation defined on $\Omega^0 \cup \Omega^1 \cup \cdots \cup \Omega^r$ the domains $\Omega^{r+1,j}$ and the corresponding approximations $(\pi^{r+1,j}, \kappa^{r+1,j})$ for $j = 1, \ldots, p_{r+1}$, thereby extending the approximation to $\Omega^0 \cup \Omega^1 \cup \cdots \cup \Omega^{r+1}$. In Fig. 2, we illustrate the construction of the domain $\Omega^0 \cup \Omega^1 \cup \Omega^2 \cup \Omega^3$ for a system on the plane with $p_1 = 4$, $p_2 = 12$, and $p_3 = 24$.

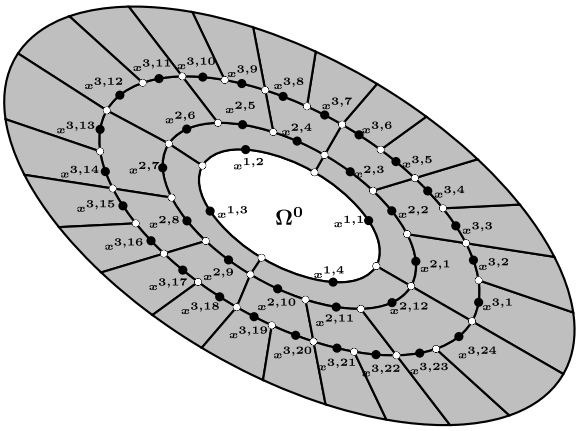

Fig. 2 Construction of sublevel set domains $\Omega^0 \cup \Omega^1 \cup \Omega^2 \cup \Omega^3$

Each outer boundary on level one is subdivided into three sets to form the inner boundaries of the patches on level two, and then each outer boundary on level two is subdivided into two sets to form the inner boundaries of the patches on level three.

The final piecewise smooth approximations to (π, κ), which we call the *patchy approximations* and denote by $\pi_{\mathrm{pch}} : \Omega \to \mathbb{R}$ and $\kappa_{\mathrm{pch}} : \Omega \to \mathbb{R}^m$, are given by

$$\pi_{\mathrm{pch}}(x) = \pi^{i,j}(x), \quad \text{if } x \in \Omega^{i,j} \cap \big(\mathcal{S}^{i+1}\big)^c,$$
$$\kappa_{\mathrm{pch}}(x) = \kappa^{i,j}(x), \quad \text{if } x \in \Omega^{i,j} \cap \big(\mathcal{S}^{i+1}\big)^c,$$

where $0 \le i \le r$ and $1 \le j \le p_i$, and $\Omega = \bigcup_{i=0}^{r} \Omega^i$ is the dynamically constructed patchy computational domain.

4.2.1 Conditions for the Existence of High-level Extensions

In this section, we consider the solvability of (13b) for the unknown κ^* and the computation of its Taylor series. In general, as we move away from the origin, the question of when (13b) is solvable for κ^* is difficult to answer. That being said, when f is control-affine, say $f(x,u) = f_0(x) + G(x)u$, and ℓ is assumed to be quadratic in u, say $\ell(x,u) = Q(x) + \frac{1}{2}u'R(x)u$, then the mapping

$$u \mapsto \frac{\partial \pi^{\alpha_j,\beta_j}}{\partial x}\big(f(x,u)\big)\frac{\partial f}{\partial u}(x,u) + \frac{\partial \ell}{\partial u}(x,u) \tag{14}$$

is a polynomial in $u \in \mathbb{R}^m$ of degree $d \ge 1$. An important case is where $d = 1$, i.e., where π^{α_j,β_j} is quadratic, for then (14) is linear in u. In this case, if $\pi^{\alpha_j,\beta_j}(x) = P_0 + P_1 \cdot x + \frac{1}{2}x'P_2x$ where $P_0 \in \mathbb{R}$, $P_1 \in \mathbb{R}^{1\times n}$, and $P_2 \in \mathbb{R}^{n\times n}$, then it is straightforward to verify that (13b) reduces to the linear equation

$$0 = u'\big[G(x)'P_2G(x) + R(x)\big] + \big[P_1 + f_0(x)'P_2\big]G(x),$$

and consequently

$$\kappa^*(x) = -\big[G(x)'P_2G(x) + R(x)\big]^{-1}G(x)'\big[P_1' + P_2f_0(x)\big],$$

provided that $G(x)'P_2G(x) + R(x)$ is invertible.

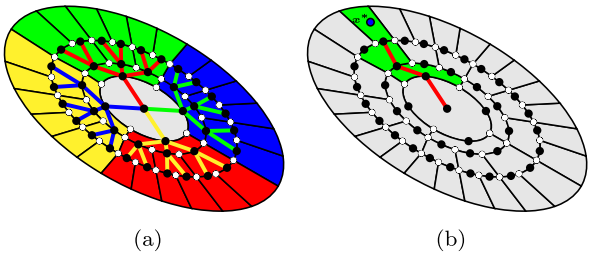

Fig. 3 Inherent rooted tree structure; (**a**) rooted tree associated to the patchy domain decomposition for the closed-loop dynamics, (**b**) unique path on rooted tree from x^* to the origin

4.3 Inherent Rooted Tree Structure of the Patches

A useful feature of the construction of the patches as described in the previous section is the inherent rooted tree structure of the patch points, see Fig. 3a. In this rooted tree, the origin in $\mathbb{R}^n$ corresponds to the root. The patch points on the outer boundary $\mathcal{S}^1 = \partial\Omega^0$ of the Albrekht level set Ω^0 are on the same level in the rooted tree and are the children of the root. Then, each patch point on the Albrekht level set has children that are on the next level in the rooted tree, etc. This rooted tree structure is extremely useful when evaluating the numerically computed solution for the optimal control because it minimizes the number of patches to search as a closed-loop trajectory evolves through the patches. For example, suppose that the initial condition of the system is x^*, represented by the blue dot in Fig. 3b. Then as the lateral boundaries of the patches are (approximately) invariant, i.e., trajectories of the closed-loop system will not cross the lateral boundaries, one will need to evaluate only the solutions $\kappa^{i,j}$ that correspond to the green patches as shown in Fig. 3b. Thus, as the closed-loop trajectory evolves, in order to evaluate the patchy feedback control κ_{pch}, it is sufficient to do a point search in only those patches that correspond to the unique path from the initial node associated to x^* to the root. Hence, the patches corresponding to the yellow, blue, and red patches can be ignored, and thereby decreases the time it takes to evaluate the numerically computed control.

5 Adaptive Partitioning of Outer Boundaries

In this section, we outline an adaptive subdivision method for partitioning the outer boundaries of a newly constructed domain level Ω^r. The main advantage of the method, say over a predetermined partitioning scheme, is to reduce the number of patch points at where the algorithm in Sect. 3 is executed and to determine the regions of the state space where the error is growing more rapidly.

Suppose that the rth-level domain Ω^r has been computed and we seek to extend the approximation from the outer boundary $\mathcal{S}^{r+1} = \partial\Omega^r$. The boundary $\mathcal{S}^{r+1}$ is the union of the outer boundaries of $\Omega^{r,1}, \ldots, \Omega^{r,p_r}$. Therefore, to construct the $(r+1)$ level domain Ω^{r+1}, we first need to partition the outer boundary of each $\Omega^{r,j}$. This can be done in a predetermined manner. For example, we can partition each outer boundary of $\Omega^{r,j}$ into two sets so that the number of patches from level-to-level doubles, but this would seriously limit the practicality of the method in high dimensions.

Instead, one could partition the outer boundary of $\Omega^{r,j}$ in an adaptive way by considering the accuracy of the solutions $(\pi^{r,j}, \kappa^{r,j})$ on $\Omega^{r,j}$. To this end, we define the *relative error* $\rho^{r,j} : \Omega^{r,j} \to \mathbb{R}$ by

$$\rho^{r,j}(x) = \frac{\pi^{r,j}(x) - \pi^{\alpha_j,\beta_j}(f(x,\kappa^{r,j}(x))) - \ell(x,\kappa^{r,j}(x))}{\pi^{r,j}(x)},$$

where $0 \le \alpha_j \le r$, $1 \le \beta_j \le p_{\alpha_j}$, and $\Omega^{\alpha_j,\beta_j}$ is the unique patch domain that contains the image of the patch point $x^{r,j}$ under the closed-loop dynamics. We can decide to subdivide the outer boundary of $\Omega^{r,j}$ if

$$\sup_{x \in \partial_{\text{out}}\Omega^{r,j}} \left|\rho^{r,j}(x)\right| > \epsilon_r \tag{15}$$

for some desired tolerance $\epsilon_r > 0$, where by $\partial_{\text{out}}\Omega^{r,j}$ we denote the outer boundary of $\Omega^{r,j}$. If (15) holds, we subdivide $\partial_{\text{out}}\Omega^{r,j}$ into $q \ge 2$ sets of approximately equal size. As a result of this subdivision process, we obtain a final partition of each outer boundary of $\Omega^{r,j}$ and, consequently, a partition $\mathcal{S}^{r+1,1}, \ldots, \mathcal{S}^{r+1,p_{r+1}}$ of the boundary $\mathcal{S}^{r+1}$. With this method, the number of sets used to partition the outer boundary of each $\Omega^{r,j}$ will vary. In particular, the outer boundaries of the patches $\Omega^{r,j}$ where the relative error is growing rapidly will be partitioned into more sets than those where the relative error is growing more slowly.

For later use, we define the *relative error function* $\rho_{\text{pch}} : \Omega \to \mathbb{R}$ by

$$\rho_{\text{pch}}(x) = \rho^{i,j}(x), \quad \text{if } x \in \Omega^{i,j} \cap \left(\mathcal{S}^{i+1}\right)^c$$

for $0 \le i \le r$, $1 \le j \le p_i$, and where $\Omega = \bigcup_{i=0}^{r} \Omega^i$. We also define the *absolute error function* $\varepsilon : \Omega \to \mathbb{R}$ by

$$\varepsilon_{\text{pch}}(x) = \pi(x) - \pi_{\text{pch}}(x).$$

6 Construction of the Initial Level Set in 3D

In this section, we describe some of the details in constructing the initial level set $\pi^0(x) = c_1$ for 3D systems. Although the method described in the previous section is applicable in arbitrary dimensions, we focus on the 3D systems as it is representative of the general case.

To construct the Albrekht level set $\mathcal{S}^1 = \partial\Omega^0 = \{x : \pi^0(x) = c_1\}$, we use the fact that near x^0, the level sets of π^0 are approximated by the level sets $\frac{1}{2}x'Px = c$. To this end, we need the following.

Lemma 6.1 *Let $\{v_1, \ldots, v_n\}$ be a set of orthonormal eigenvectors of the symmetric matrix $P \succ 0$, and form the matrix $V = [v_1\ v_2\ \cdots\ v_n]$. Let $\lambda_1, \ldots, \lambda_n > 0$ denote the eigenvalues of P and set $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Then*

$$\left\{x : \frac{1}{2}x'Px = \tilde{c}_1\right\} = \left\{x = V\sqrt{\Lambda}^{-1}z : \frac{1}{2}z'z = \tilde{c}_1\right\}.$$

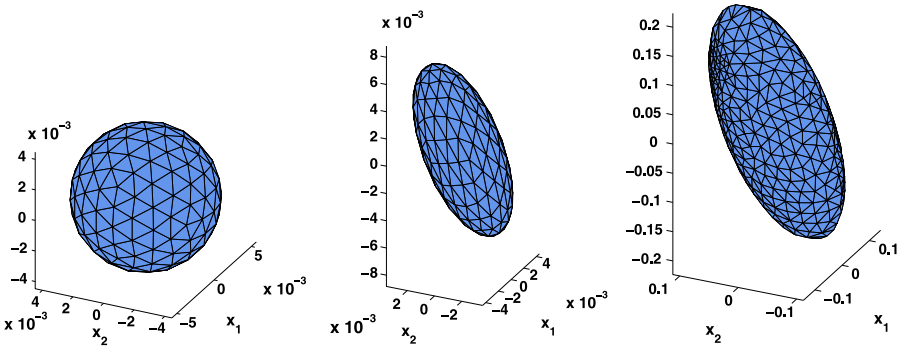

Fig. 4 Construction of the initial level set in 3D: (**a**) icosahedral grid of sphere, (**b**) level set $\frac{1}{2}x'Px = \tilde{c}_1$, (**c**) level set $\pi^0(x) = c_1$

Proof We first note that $V^{-1} = V'$. Suppose that $z'z = 2\tilde{c}_1$ and let $x = V\sqrt{\Lambda}^{-1}z$. Then

$$\frac{1}{2}x'Px = \frac{1}{2}z'\sqrt{\Lambda}^{-1}V'PV\sqrt{\Lambda}^{-1}z = \frac{1}{2}z'\sqrt{\Lambda}^{-1}\Lambda\sqrt{\Lambda}^{-1}z = \frac{1}{2}z'z = \tilde{c}_1.$$

This completes the proof. □

We can therefore create the level sets $\tilde{\mathcal{S}}^1 := \{x : \frac{1}{2}x'Px = \tilde{c}_1\}$ by mapping the sphere of radius $\sqrt{2\tilde{c}_1}$ under the transformation $V\sqrt{\Lambda}^{-1}$. The value of $\tilde{c}_1$ is chosen so that $\tilde{c}_1 \ll c_1$ and thus $\tilde{\mathcal{S}}^1 \subset \operatorname{int}(\Omega^0)$. We then construct $\mathcal{S}^1$ as follows. Let $f^0(x) := f(x, \kappa^0(x))$, and let $v(x) := \frac{x - f^0(x)}{\|x - f^0(x)\|}$. Starting with $x \in \tilde{\mathcal{S}}^1$, we follow the curve

$$\alpha(t, x) = \left(f^0\right)^{-1}\left(f^0(x) + tv(x)\right) \tag{16}$$

from $t = 0$ in forward time until reaching the level set $\mathcal{S}^1$. Of course, this is performed on a discretization of $\tilde{\mathcal{S}}^1$. We choose an icosahedral discretization of the sphere [28] with refinement level $k \in \mathbb{N}_0$. In Fig. 4, we illustrate the result of this process for the 3D Example 7.3, starting with an icosahedral partition of level $k = 2$. Each vertex z of the icosahedral grid is mapped under $V\sqrt{\Lambda}^{-1}$, $x = V\sqrt{\Lambda}^{-1}z$, and then we follow the curve (16) until $\pi^0(\alpha(t, x)) = c_1$. This results in a grid and corresponding partition of the level set $\pi^0(x) = c_1$ that contains $p_1 = 20 \times 4^k = 320$ triangle patches. The collection of triangle patches is the partition $\mathcal{S}^{1,1}, \ldots, \mathcal{S}^{1,p_1}$ of the level set $\pi^0(x) = c_1$. The patch points $x^{1,j} \in \mathcal{S}^{1,j}$ are chosen as the centroids of the triangles specified by $\mathcal{S}^{1,j}$.

7 Numerical Tests

In this section, we present three numerical tests that illustrate the high-order accuracy of our numerical algorithm. The numerical tests were performed on a PC with an Intel Duo-Core E8600 3.33 GHz processor and with 4 GB RAM, running Windows XP Pro and Matlab 7.8 (R2009a).

In the first example, we apply our method to a system for which the optimal cost function π and optimal control κ are known explicitly.

Example 7.1 Consider the linear system

$$z_1(k+1) = z_1(k) + hz_2(k),$$
$$z_2(k+1) = z_2(k) + hu(k)$$

and running cost $\ell(z,u) = \frac{h}{2}(z_1^2 + z_2^2 + u^2)$, where $h = 0.05$. The optimal cost function π and optimal feedback κ are given by $\pi(z) = \frac{1}{2}z'Pz$ and $\kappa(z) = Kz$, where P and K are the solutions to the associated Riccati equation. Consider now the change of coordinates $x \mapsto z = \phi(x)$ given by

$$z = (z_1, z_2) = \big(\phi_1(x), \phi_2(x)\big) := \left(x_1, x_2 + \sin(x_1)\exp\left(-\frac{x_1^2}{100}\right)\right).$$

In the $x = (x_1, x_2)$ coordinates, the system becomes

$$x(k+1) = \phi^{-1}\big(\phi_1\big(x(k)\big) + h\phi_2\big(x(k)\big), \phi_2\big(x(k)\big) + hu\big),$$

where $\phi^{-1}(z_1, z_2) = (z_1, z_2 - \sin(z_1)\exp(-\frac{z_1^2}{100}))$, and the running cost is $\ell(x,u) = \frac{h}{2}(\phi_1(x)^2 + \phi_2(x)^2 + u^2)$. The optimal cost function and optimal control in the x-coordinates are then $\pi(x) = \frac{1}{2}\phi(x)'P\phi(x)$ and $\kappa(x) = K\phi(x)$, respectively. Having an explicit expression for π, we will be able to explicitly compute the absolute error function $\varepsilon_{\mathrm{pch}}(x) = \pi(x) - \pi_{\mathrm{pch}}(x)$.

We computed the patchy approximations $(\pi_{\mathrm{pch}}, \kappa_{\mathrm{pch}})$ to (π, κ) using our method with initial Albrekht solution (π^0, κ^0) of degree $d = 3$. The Albrekht solution was then extended to $N = 220$ patch levels as described in Sect. 4. The values of the cost levels c_r were chosen as $c_r = (c_1 + (r-1)\Delta c)^2$ for $r = 1, \ldots, N+1$, with $c_1 = 0.1$ and $\Delta c = 0.02$. The outer cost level on the level set $N = 220$ is therefore $c_{N+1} = 20.25$. The adaptive partitioning method of the outer boundaries as described in Sect. 5 was performed with the parameters $q = 2$, e.g., each patch is subdivided into two patches, and the relative error tolerance was set to $\epsilon_r = 1 \times 10^{-2}$. The number of patch points on the first level set initialized to $p_1 = 66$. Through the level-to-level adaptive partitioning scheme, the number of patch points on the last cost level was $p_N = 114$. The total CPU time for the computation was 347 seconds, the maximum absolute error was $\|\varepsilon_{\mathrm{pch}}\|_\infty = 0.4751$, and the maximum relative error was $\|\rho_{\mathrm{pch}}\|_\infty$ =9.3629e-3. In Fig. 5, we plot the domain of computation Ω defined by the union of the dynamically computed patch domains $\Omega^{i,j}$, and in Fig. 6, we display the

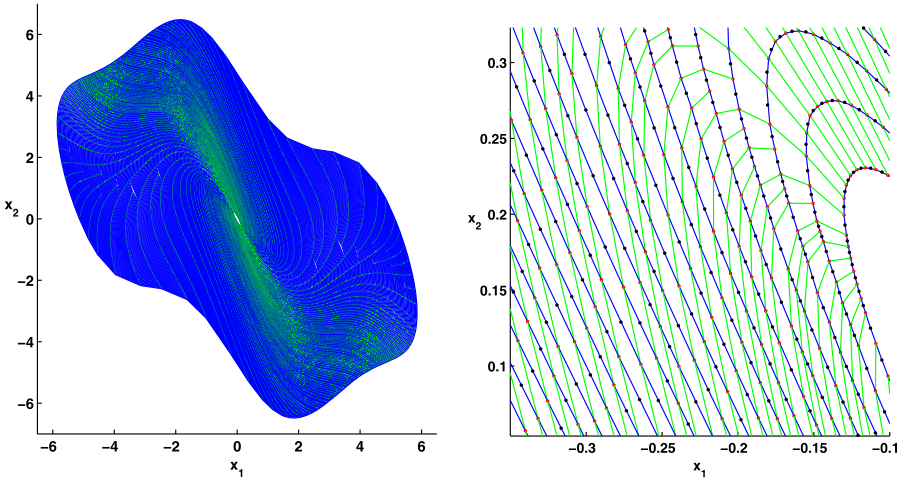

Fig. 5 Computational domain for Example 7.1: (**a**) patchy domain Ω, (**b**) zoomed in portion $[-0.35, -0.1] \times [0.05, 0.30]$

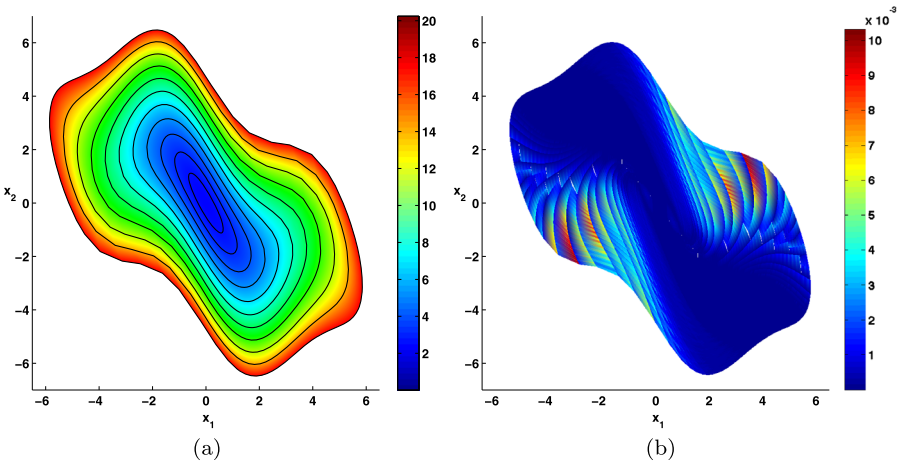

Fig. 6 (**a**) Contour plot of patchy value function, (**b**) contour plot of the relative error function ρ_{pch}

contour plot of the patchy value function $\pi_{\text{pch}}(x)$ and the contour plot of the relative error function $\rho_{\text{pch}}(x)$.

In Table 1, we perform an error analysis and investigate the dependence of the errors $\|\varepsilon_{\text{pch}}\|_\infty$ and $\|\rho_{\text{pch}}\|_\infty$ on the cost step-size Δc. To this end, we fixed the degree $d = 3$ and varied the cost step-size Δc, or equivalently varied the number of level sets N, but of course required that the value of the final cost c_N be constant for each different N. For each test, we fixed the initial cost at $c_1 = 0.3$ and set the final cost to $c_N = 2.25$. In Fig. 7a, we display a semi-log plot of the absolute error $\|\varepsilon_{\text{pch}}\|_\infty$ as a function of the cost step-size Δc. The plot displays an exponential dependency

Table 1 Error as a function of the cost step-size Δc for Example 7.1

Δc	# level sets (N)	# of patches (Σp_i)	CPU time (sec)	$\|\varepsilon_{\text{pch}}\|_\infty$	$\|\rho_{\text{pch}}\|_\infty$
0.05	24	1192	19	3.6049e-2	3.1423e-3
0.04	30	1504	24	2.1275e-2	2.3871e-4
0.03	40	2216	35	1.3099e-2	8.1553e-4
0.02	60	3362	53	6.7619e-3	9.7153e-4
0.01	120	6943	112	5.8620e-3	3.3256e-4
0.005	240	13969	225	4.1992e-3	2.2673e-4
0.0025	480	28015	490	3.7223e-3	2.1967e-4
0.001	1200	70143	1225	3.3013e-3	1.9613e-4

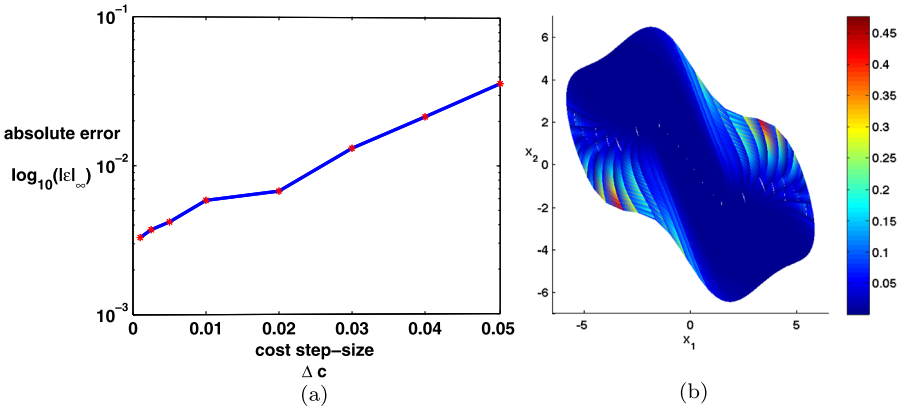

Fig. 7 (**a**) Absolute error $\|\varepsilon_{\text{pch}}\|_\infty$ as a function of the cost step-size Δc, (**b**) contour plot of the absolute error function ε_{pch}

of the absolute error $\|\varepsilon_{\text{pch}}\|$ on Δc and illustrates that, as we increase the number of levels to reach a fixed cost value, the error incurred by the patchy solution decreases. In Fig. 7b, we display the contour plot of the absolute error function.

Example 7.2 In this example, we consider a controlled Duffing oscillator given by

$$\begin{aligned} x_1(k+1) &= x_1(k) + hx_2(k), \\ x_2(k+1) &= x_2(k) + h\big(x_1(k) - x_1^3(k) - \delta x_2(k) + u(k)\big). \end{aligned} \tag{17}$$

System (17) is an Eulerian discretization (sampling interval h) of the continuous-time dynamic model of a magneto-elastic beam in the field of two permanent magnets [29]. A thin steel beam is clamped vertically at one of its ends onto a rigid frame. Near the free tip of the beam, two magnets are symmetrically placed on the frame and exert a magnetic force on the beam causing it to buckle with its free tip settling close to one of the magnets. The state variable x_1 is a measure of the deflection of the beam's free tip from the static vertical position. The unforced system has three equilibria, one at

Table 2 Relative error and CPU time as a function of d

d	p_N	CPU time (sec)	$\|\rho_{\text{pch}}\|_\infty$
1	57	22	3.8113e-3
3	58	25	3.7562e-5
5	58	34	7.2084e-8
7	58	53	7.7843e-10

the origin and at $(\pm 1, 0)$. The origin is a saddle with eigenvalues $1 - \frac{h\delta}{2} \pm \frac{h}{2}\sqrt{\delta^2 + 4}$ and is therefore unstable, and the other equilibria are sinks. A horizontal force u can be applied to the rigid frame and the control problem is to stabilize the unstable vertical position. To this end, we consider the quadratic cost

$$J\big(x(0), u(\cdot)\big) = \frac{h}{2}\sum_{k=0}^{\infty}\left(x_1(k)^2 + \frac{1}{2}x_2(k)^2 + \frac{1}{2}u(k)^2\right)$$

and seek an optimal regulator to stabilize the system. Using our numerical method, we computed patchy approximations $(\pi_{\text{pch}}, \kappa_{\text{pch}})$ to the optimal cost function and the optimal control (π, κ) using $N = 140$ patch levels as described in Sect. 4. The sampling interval was set to $h = 0.05$, and the damping coefficient in the model (17) was set to $\delta = 0.01$. The method of adaptive partitioning of the outer boundaries as described in Sect. 5 was performed with the parameters $q = 2$ and $\epsilon_r = 1 \times 10^{-2}$, and with the number of patch points on the initial level set chosen as $p_1 = 24$. Through the adaptive partitioning scheme, the number of patch points on the last level was $p_N = 58$. The values of the cost levels were chosen as $c_r = (c_1 + (r-1)\Delta c)^2$ with $c_1 = 0.01$ and $\Delta c = 0.01$ for $r = 1, \ldots, N+1$. Hence, the cost on the outer boundary of the last level set is $c_{N+1} = 1.99$. The initial Albrekht approximation (π^0, κ^0) was computed to degree $d = 3$. The total CPU time for the computation was 86 seconds, and the maximum relative error on the patchy domain Ω was $\|\rho_{\text{pch}}\|_\infty = 3.5904\text{e-}4$. In Fig. 8a, we plot the computational domain Ω determined by the dynamically computed patches, and in Fig. 8b, we plot representative closed-loop trajectories with initial conditions on $\partial\Omega$. In Fig. 9a, we display the contour plot of the patchy value function π_{pch}, and in Fig. 9b, we display the contour plot of the relative error function ρ_{pch}.

In Table 2, we perform an error analysis by investigating the dependence of $\|\rho_{\text{pch}}\|_\infty$ on the order of approximation d. To this end, we fixed the cost step-size to $\Delta c = 0.01$ and varied the degree of approximation $d = 1, 3, 5, 7$. Odd degrees were chosen because the computed polynomial approximations were of odd degree, i.e., the degree $d = 1$ and degree $d = 2$ polynomial approximations coincided, etc. The total number of level sets was fixed to $N = 50$, and the number of patch points on the last level set p_N remained (approximately) constant for each chosen degree d. Using the data from Table 2, in Fig. 10, we plot the relative error as a function of the order d. The plot displays an exponential dependence of $\|\rho_{\text{pch}}\|_\infty$ on d and illustrates the high-order accuracy of our numerical method.

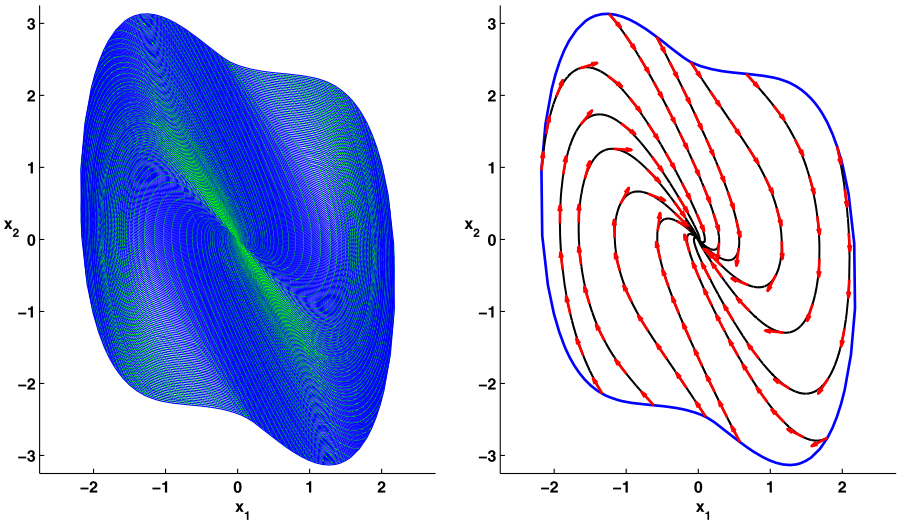

Fig. 8 Duffing oscillator: (**a**) patchy domain, (**b**) representative closed-loop trajectories

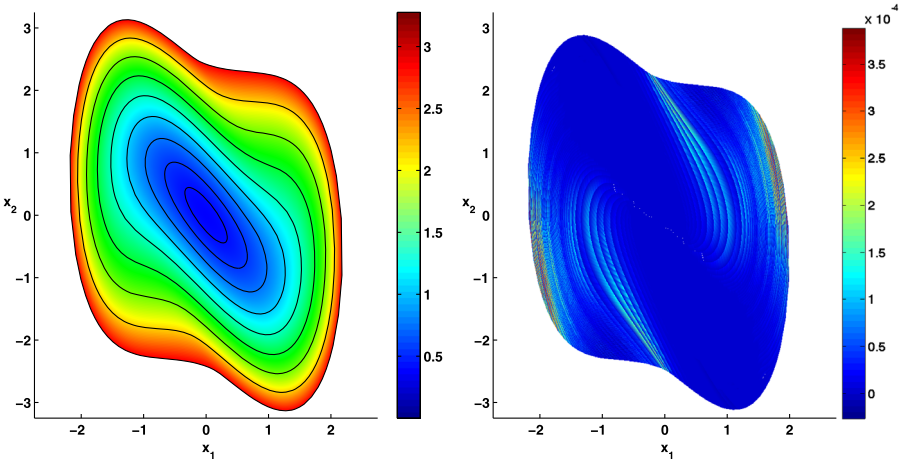

Fig. 9 Duffing oscillator: (**a**) contour plot of the patchy value function, (**b**) relative error ρ_{pch}

Example 7.3 Consider the three-dimensional single-input system

$$\begin{aligned} x_1(k+1) &= x_1(k) + h\big(x_2 + x_2(k)x_3^3(k)\big), \\ x_2(k+1) &= x_2(k) + h\big(x_3(k) - x_1^3(k)x_3(k)\big), \\ x_3(k+1) &= x_3(k) + hu(k). \end{aligned} \tag{18}$$

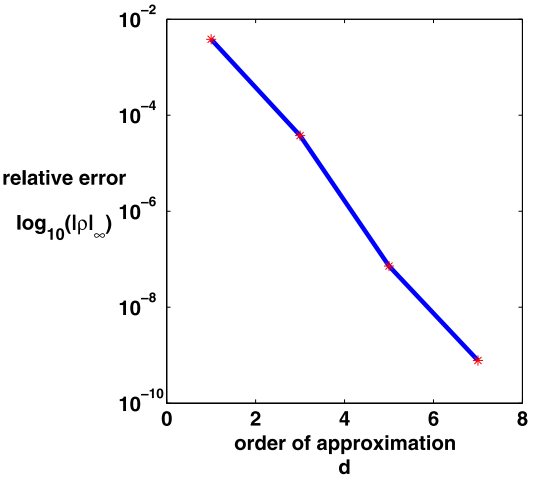

Fig. 10 Relative error as a function of d

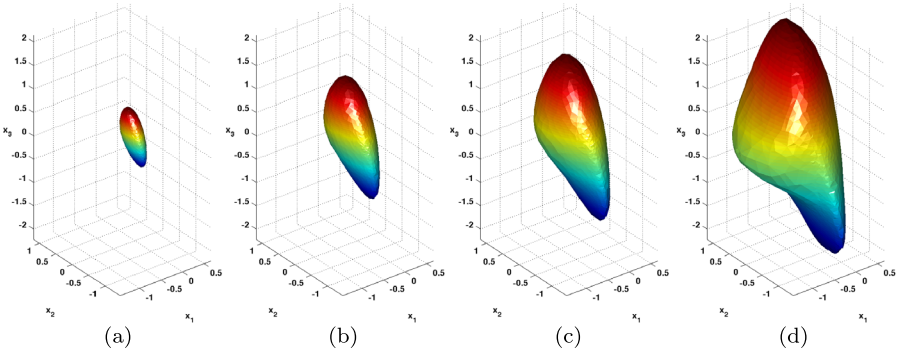

Fig. 11 Evolution of the level sets for Example 7.3: (**a**) level set $r = 10$, (**b**) level set $r = 25$, (**c**) level set $r = 35$, (**d**) level set $r = 50$

As cost we take

$$J\big(x(0), u(\cdot)\big) = \frac{h}{2}\sum_{k=0}^{\infty}\left(\frac{1}{4}x_1(k)^2 + \frac{1}{4}x_2(k)^2 + \frac{1}{4}x_3(k)^2 + u^2(k)\right).$$

The value $h = 0.05$ was chosen. A total number of $N = 50$ patch levels were constructed with cost values $c_r = (c_1 + (r-1)\Delta c)^2$ for $r = 1, \ldots, N+1$, where $c_1 = 0.1$ and $\Delta c = 0.02$. Hence, the cost on the outer boundary of the last level set is $c_{N+1} = 1.21$. The method of adaptive partitioning of the outer boundaries as described in Sect. 5 was performed with the parameters $q = 4$ and $\epsilon_r = 5 \times 10^{-3}$, and with the number of patch points on the initial level set chosen as $p_1 = 320$. Through the adaptive partitioning scheme, the number of patch points on the last level was $p_N = 1378$. The initial Albrekht solution (π^0, κ^0) was computed to degree $d = 3$. The total CPU time for the computation was 501 seconds, and the maximum relative error was $\|\rho_{\text{pch}}\|_\infty$ =3.6101e-3. In Fig. 11, we plot the evolution of the level sets of

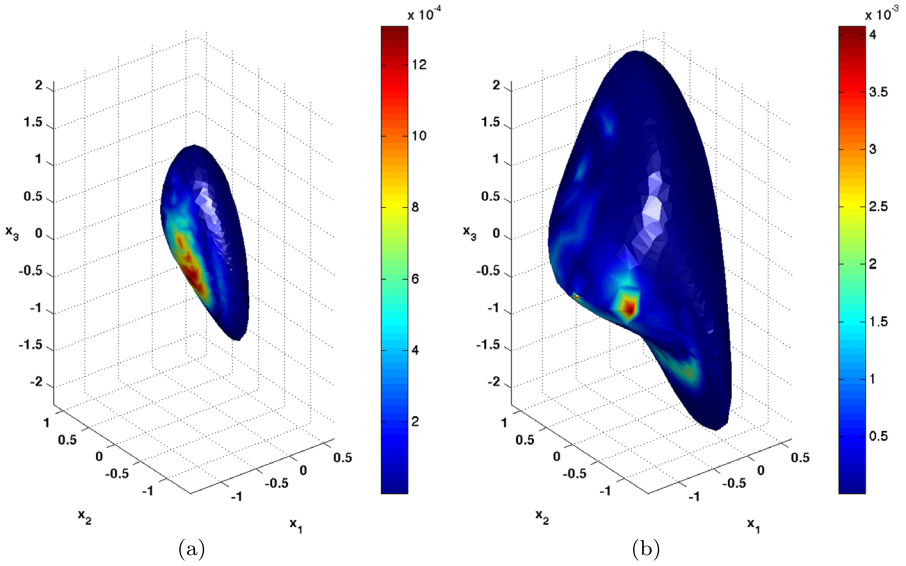

Fig. 12 Relative error of patchy approximation: (**a**) level set $r = 25$, (**b**) level set $r = 50$

Table 3 Relative error and CPU time as functions of the number of patch points

p_N	CPU time (sec)	$\|\rho_{\text{pch}}\|_\infty$
64	7	2.7099e-2
90	9	1.8521e-2
176	18	9.6424e-3
348	35	2.8898e-3
535	54	2.0575e-3
684	68	2.0865e-3
863	85	1.7612e-3
1064	105	1.6124e-3
1252	123	1.4745e-3
1386	139	1.4099e-3

the value function under the reversed closed-loop dynamics, and in Fig. 12, we plot the relative error incurred by the patchy approximation on the outer boundaries of the level sets $r = 25$ and $r = 50$.

In Table 3, we perform an error analysis by investigating the dependence of $\|\rho_{\text{pch}}\|_\infty$ on the number of patch points on each level set. To this end, we set the initial cost to $c_0 = 0.1$ and fixed the cost step-size to $\Delta c = 0.05$. We computed $N = 10$ levels so that the final cost on the outer boundary was $c_{N+1} = 0.36$. We performed 10 numerical tests by varying the number of patch points from 64 to 1386 on each level set. In all the tests, the adaptive partitioning scheme was not implemented, and thus the number of patch points p_i on each level set was constant for each $i = 1, \ldots, N$. Using the data from Table 3, in Fig. 13, we plot the relative error as a function of the

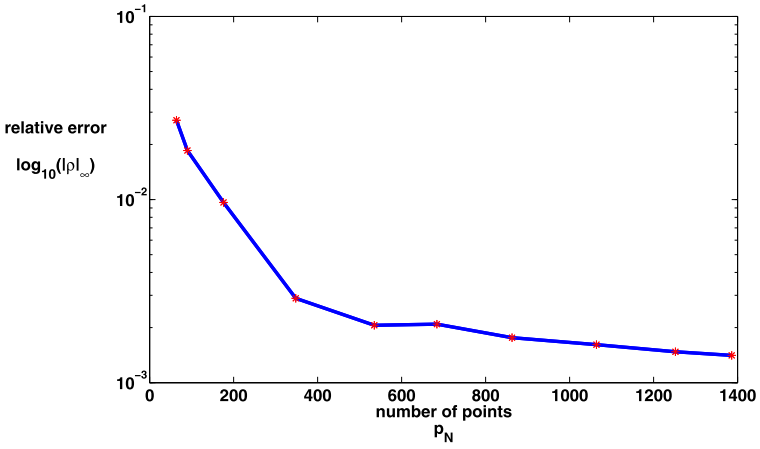

Fig. 13 Relative error as a function of the number of patch points

number of patch points p_N. The plot shows that $\|\rho_{\text{pch}}\|_\infty$ initially declines rapidly as the number of patch points are increased but then begins to level off, and the effect of adding more patch points becomes less noticeable. This is due, in part, to the fact that the closed-loop trajectories deviate rapidly in some areas of the state space and simply adding more points on the initial level set does not guarantee that the patch points will stay close to each other as the level sets evolve under the closed-loop dynamics. This motivates further investigation in using local geometric information of the level sets and closed-loop trajectories to identify regions where the density of points should be increased.

8 Conclusions

We have presented a numerical method that computes an approximate solution to Bellman's dynamic programming equation arising from an optimal stabilization problem for nonlinear discrete-time systems. In our method, we propagate the level sets of the computed value function under the computed closed-loop dynamics in reverse time. Patch domains are dynamically computed from level set to level set, and polynomial solutions associated to the patch domains are computed using a Cauchy–Kowalevski-type algorithm. An adaptive scheme is used to increase the density of points on the level sets depending on the relative error incurred by the computed solution. The adaptive scheme decreases the computational burden of the algorithm by adding points only on regions where the error is greater than some prescribed relative error tolerance. Numerical tests in 2D and 3D where included that illustrate the accuracy of the method.

Future research directions are focused on developing the patchy algorithm on systems of state dimension $n \geq 4$. One of the main tasks consists in developing and implementing efficient algorithms that describe the local geometry of the level sets and, in particular, in efficiently storing and dynamically changing nearest neighbor

information as the level sets are propagated. Nearest neighbor information is needed when computing the relative error of the patchy solution to determine which patches need to be partitioned. Another issue is the construction of the initial Albrekht level set in high dimensions. The current implementation in 3D relies on the ability to discretize the 2-sphere using an icosahedral discretization, and this intuition is lost in dimensions $n \geq 4$. A possible approach in high dimensions is to uniformly sample the Albrekht level set $\pi^0(x) = c_1$ and then use a local repulsion algorithm to uniformly distribute the points on the level set [30]. Finally, Fig. 11 shows the onset of a cusp for the 3D Example 3. This type of nonlinear phenomenon is typical in Hamilton–Jacobi equations and motivates further investigation of how our numerical algorithm can be extended to handle this type of nonlinearity. Other future work will investigate error estimates and convergence results for the numerical method presented.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that improved the presentation of this paper. The first author acknowledges the support of the Natural Research Council Postdoctoral Associateship program and the Naval Postgraduate School.

References

1. Laub, A.J.: A Schur method for solving algebraic Riccati equations. IEEE Trans. Autom. Control **24**, 913–921 (1979)
2. Crandall, M.G., Lions, P.L.: Viscosity solutions of Hamilton–Jacobi equations. Trans. Am. Math. Soc. **277**, 1–42 (1983)
3. Albrekht, E.G.: On the optimal stabilization of nonlinear systems. J. Appl. Math. Mech. **25**, 1254–1266 (1961)
4. Leake, R.J., Liu, R.-W.: Construction of suboptimal control sequences. SIAM J. Control **5**, 54–63 (1967)
5. Lukes, D.L.: Optimal regulation of nonlinear dynamical systems. SIAM J. Control **7**, 75–100 (1969)
6. Capuzzo-Dolcetta, I., Ishii, H.: Approximate solutions of the Bellman equation of deterministic control theory. Appl. Math. Optim. **11**, 161–181 (1984)
7. Falcone, M., Ferretti, R.: Discrete time high-order schemes for viscosity solutions of Hamilton–Jacobi–Bellman equations. Numer. Math. **67**, 315–344 (1994)
8. Beard, R.W., Sardis, G.N., Wen, J.T.: Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation. Automatica **33**, 2159–2177 (1997)
9. Mracek, C.P., Cloutier, J.R.: Control designs for the nonlinear benchmark problem via the state-dependent Riccati equation method. Int. J. Robust Nonlinear Control **8**, 401–433 (1998)
10. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Cambridge University Press, Cambridge (1999)
11. Markman, J., Katz, I.N.: An iterative algorithm for solving Hamilton–Jacobi type equations. SIAM J. Sci. Comput. **22**, 312–329 (2000)
12. Navasca, C., Krener, A.J.: Patchy solution of the HJB PDE. In: Chiuso, A., Ferrante, A., Pinzoni, S. (eds.) Modeling, Estimation and Control. Lecture Notes in Control and Information Sciences, vol. 364, pp. 251–270 (2007)
13. Sakamoto, N., van der Schaft, A.J.: Analytical approximation methods for the stabilizing solution of the Hamilton–Jacobi equation. IEEE Trans. Autom. Control **53**, 2335–2350 (2008)
14. Beeler, S.C., Tran, H.T., Banks, H.T.: Feedback control methodologies for nonlinear systems. J. Optim. Theory Appl. **107**, 1–33 (2000)
15. Cacace, S., Cristiani, E., Falcone, M., Picarelli, A.: A patchy dynamic programming scheme for a class of Hamilton–Jacobi–Bellman equations. SIAM J. Sci. Comput. **34**, 2625–2649 (2012)
16. Garrard, W.L., Jordan, J.M.: Design of nonlinear automatic flight control systems. Automatica **13**, 497–505 (1977)
17. Yoshida, T., Loparo, K.A.: Quadratic regulatory theory for analytic non-linear systems with additive controls. Automatica **25**, 531–544 (1989)

18. Spencer, B.F., Timlin, T.L., Sain, M.K., Dyke, S.J.: Series solution of a class of nonlinear optimal regulators. J. Optim. Theory Appl. **91**, 321–345 (1996)
19. Ancona, F., Bressan, A.: Patchy vector fields and asymptotic stabilization. ESAIM Control Optim. Calc. Var. **4**, 445–471 (1999)
20. Hunt, T.: A proof of the higher order accuracy of the patchy method for solving the Hamilton–Jacobi–Bellman equation. PhD thesis, University of California, Davis, CA (2011)
21. Nešić, D., Teel, A.R.: Backstepping on the Euler approximate model for stabilization of sampled-data nonlinear systems. In: Proc. 40th IEEE Conference on Decision and Control, pp. 1737–1742 (2001)
22. Nešić, D., Teel, A.R., Kokotović, P.V.: Sufficient conditions for the stabilization of sampled-data nonlinear systems via discrete-time approximations. Syst. Control Lett. **38**, 259–270 (1999)
23. Grüne, L., Nešić, D.: Optimization-based stabilization of sampled-data nonlinear systems via their approximate discrete-time models. SIAM J. Control Optim. **42**(42), 98–122 (2003)
24. Keerthi, S.S., Gilbert, E.G.: Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations. J. Optim. Theory Appl. **57**, 265–293 (1988)
25. Bellman, R.: Introduction to the Mathematical Theory of Control Processes. Academic Press, New York (1971)
26. Navasca, C.: Local solutions of the dynamic programming equations and the Hamilton–Jacobi–Bellman PDEs. PhD Thesis, University of California, Davis (2002)
27. Lewis, F.: Optimal Control. Wiley-Interscience, New York (1986)
28. Baumgardner, J.R., Frederickson, P.O.: Icosahedral discretization of the two-sphere. SIAM J. Numer. Anal. **22**, 1107–1115 (1985)
29. Guckenheimer, J., Holmes, P.: Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields. Springer, New York (1983)
30. Witkin, A., Heckbert, P.: Using particles to sample and control implicit surfaces. Comput. Graph. **28**, 269–278 (1994)