From Bellman to Dijkstra: Set-Oriented Construction of Globally Optimal Controllers

Lars Grüne[1] and Oliver Junge[2(✉)]

[1] Mathematical Institute, University of Bayreuth, Bayreuth, Germany
lars.gruene@uni-bayreuth.de
[2] Department of Mathematics, Technical University of Munich, Munich, Germany
oliver.junge@tum.de

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.
>
> Richard Bellman, 1957

Abstract. We review an approach for discretizing Bellman's optimality principle based on piecewise constant functions. By applying this ansatz to a suitable dynamic game, a discrete feedback can be constructed which robustly stabilizes a given nonlinear control system. Hybrid, event and quantized systems can be naturally handled by this construction.

1 Introduction

Whenever the state of some dynamical system can be influenced be repeatedly applying some control ("decision") to the system, the question might arise how the sequence of controls – the policy – can be chosen in such a way that some given objective is met. For example, one might be interested in steering the system to an equilibrium point, i.e. to *stabilize* the otherwise unstable point. In many contexts, the application of some control comes at some cost (fuel, money, time, ...) which then is accumulated over time. Typically, one is interested in meeting the given objective at minimal accumulated cost. This is the context of Richard Bellman's famous quote which already hints at how to solve the problem: One can recursively construct an optimal sequence of controls backwards in time by starting at the/some final state. It just so happens that this is also the idea of Edsger Dijkstra's celebrated algorithm for finding shortest paths in weighted directed graphs.

At the core, this procedure requires one to store the minimal accumulated cost at each state, the *value function*. According to the recursive construction

O. Junge et al. (Eds.): SON 2020, SSDC 304, pp. 265–294, 2020.
https://doi.org/10.1007/978-3-030-51264-4_11

of the sequence of optimal controls, the value function satisfies a recursion, i.e. a fixed point equation, the *Bellman equation*. From the value function at some state, the optimal control associated to that state can be recovered by solving a static optimization problem. This assignment defines a function on (a subset of) the states into the set of all possible control values and so the state can be *fed back* into the system, yielding a dynamical system without any external input. By construction, the accumulated cost along some trajectory of this *closed loop system* will be minimal.

In the case of a finite state space (with a reasonable number of states), storing the value function is easy. In many applications from, e.g., the engineering sciences, however, the state space is a subset of Euclidean space and thus the value function a function defined on a continuum of states. In this case, the value function typically cannot be represented in a closed form. Rather, some approximation scheme has to be decided upon and the value function (and thus the feedback) has to be approximated numerically.

In this chapter, we review contributions by the authors developing an approach for approximating the value function and the associated feedback by *piecewise constant functions*. This may seem like a bad idea at first, since in general one would prefer approximation spaces of higher order. However, it turns out that this ansatz enables an elegant solution of the discretized problem by standard shortest path algorithms (i.e. Dijkstra's algorithm). What is more, it also enables a unified treatment of system classes which otherwise would require specialized algorithms, like hybrid systems, event systems or systems with quantized state spaces.

As is common for some discretization, the discrete value function does not inherit a crucial property of the true one: In general, it does not decrease monotonically along trajectories of the closed loop system. In other words, it does not constitute a *Lyapunov function* of the closed loop system. As a consequence, the associated feedback may fail to stabilize some initial state. This deficiency can be cured by considering a more general problem class, namely a system which can be influenced by two independent controls – a *dynamic game*. In particular, if the second input is interpreted as some perturbation induced by the discretization, a discrete feedback results which retains the Lyapunov function property.

On the other hand, as any construction based on the Bellman equation, or more generally as any computational scheme which requires to represent a function with domain in some Euclidean space, our construction is prone to the *curse of dimension* (a term already coined by Bellman): In general, i.e. unless some specialized approximation space is employed, the computational cost for storing the value function grows exponentially in the dimension of state space. That is, in practice, our approach is limited to systems with a low dimensional state space (i.e. of dimension ≤ 4, say).

2 Problem Formulation

We are given a control system in discrete time

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0, 1, \ldots, \tag{1}$$

where $x_k \in X$ is the *state* of the system, $u_k \in U$ is the *control input* and $w_k \in W$ is some external *perturbation*. We are further given an *instantaneous cost* function g which assigns the cost

$$g(x_k, u_k) \geq 0$$

to any transition $x_k \mapsto f(x_k, u_k, w)$, $w \in W$.

Our task is to globally and optimally stabilize a given *target set* $T \subset X$ by constructing a *feedback* $u : S \to U$, $S \subset X$, such that T is an asymptotically stable set for the *closed loop system*

$$x_{k+1} = f(x_k, u(x_k), w_k), \quad k = 0, 1, \ldots \tag{2}$$

with $x_0 \in S$ for *any* sequence $(w_k)_k$ of perturbations and such that the *accumulated cost*

$$\sum_{k=0}^{\infty} g(x_k, u(x_k)) \tag{3}$$

is minimal.

System Classes. Depending on the choice of the spaces X, U and W and the form of the map f, a quite large class of systems can be modelled by (1). Most generally, X, U and W have to be compact metric spaces – in particular, they may be discrete. Common examples which will also be considered later, include

- sampled-data systems: X, U and W are compact subsets of Euclidean space, f is the time-T-map of the control flow of some underlying continuous time control system and g typically integrates terms along the continuous time solution over one sampling interval;
- hybrid systems: $X = Y \times D$, where $Y \subset \mathbb{R}^n$ compact and D is finite, U and W may be continuous (compact) sets or finite (cf. Sect. 8);
- discrete event systems: f may be chosen as a (generalized) Poincaré map (cf. Sect. 8).
- quantized systems: The feedback may receive only quantized information on the state x, i.e. x is projected onto a finite subset of X before u is evaluated on this quantized state.

3 The Optimality Principle

The construction of the feedback law u will be based on a discretized version of the optimality principle. In order to convey the basic idea more clearly, we start by considering problem (1) without perturbations, i.e.

$$x_{k+1} = f(x_k, u_k), \quad k = 0, 1, \ldots \tag{4}$$

and assume that $X \subset \mathbb{R}^d$ and $U \subset \mathbb{R}^m$ are compact, $0 \in X$ and $0 \in U$. We further assume that $0 \in X$ is a fixed point of $f(\cdot\,, 0)$, i.e. $f(0, 0) = 0$, constituting our

target set $T := \{0\}$, that $f : X \times U \to X$ and $g : X \times U \to [0, \infty)$ are continuous, that $g(0, 0) = 0$ and $\inf_{u \in U} g(x, u) > 0$ for all $x \neq 0$.

For a given initial state $x_0 \in X$ and a given sequence $\boldsymbol{u} = (u_0, u_1, \ldots) \in U^{\mathbb{N}}$ of controls, there is a unique trajectory $\boldsymbol{x}(x_0, \boldsymbol{u}) = (x_k(x_0, \boldsymbol{u}))_{k \in \mathbb{N}}$ of (4). For $x \in X$, let

$$\mathcal{U}(x) = \{\boldsymbol{u} \in U^{\mathbb{N}} : x_k(x, \boldsymbol{u}) \to 0 \text{ as } k \to \infty\}$$

denote the set of *stabilizing control sequences* and

$$S = \{x \in X : \mathcal{U}(x) \neq \emptyset\}$$

the *stabilizable subset* of X. The accumulated cost along some trajectory $\boldsymbol{x}(x_0, \boldsymbol{u})$ is given by

$$J(x_0, \boldsymbol{u}) = \sum_{k=0}^{\infty} g(x_k(x_0, \boldsymbol{u}), u_k). \tag{5}$$

Note that this series might not converge for some $(x_0, \boldsymbol{u})$. The least possible value of the accumulated cost over all stabilizing control sequences defines the *(optimal) value function* $V : X \to [0, \infty]$,

$$V(x) = \inf_{\boldsymbol{u} \in \mathcal{U}(x)} J(x, \boldsymbol{u}) \tag{6}$$

of the problem. Let $S_0 := \{x \in X : V(x) < \infty\}$ be the set of states in which the value function is finite. Clearly, $S_0 \subset S$. On S_0, the value function satisfies the *optimality principle* [2]

$$V(x) = \inf_{u \in U} \{g(x, u) + V(f(x, u))\}. \tag{7}$$

The right hand side

$$L[v](x) := \inf_{u \in U} \{g(x, u) + v(f(x, u))\}$$

of (7) defines the *Bellman operator* L on real valued functions on X. The value function V is the unique fixed point of L satisfying the boundary condition $V(0) = 0$.

Using the value function V, one can construct the feedback $u : S_0 \to U$,

$$u(x) := \operatorname*{argmin}_{u \in U} \{g(x, u) + V(f(x, u))\}, \tag{8}$$

whenever this minimum exists. Obviously, V then satisfies

$$V(x) \geq g(x, u(x)) + V(f(x, u(x))), \tag{9}$$

for $x \in S_0$, i.e. the optimal value function is a Lyapunov function for the closed loop system on S_0 (provided that V is continuous at $T = \{0\}$[1]) – and this guarantees asymptotic stability of $T = \{0\}$ for the closed loop system. By construction, this feedback u is also optimal in the sense that the accumulated cost J is minimized along any trajectory of the closed loop system.

[1] This property can be ensured by suitable asymptotic controllability properties and bounds on g.

4 A Discrete Optimality Principle

In general, the value function (resp. the associated feedback) cannot be determined exactly and some numerical approximation has to be sought. Here, we are going to approximate V by functions which are piecewise constant on some partition of X. This approach is motivated by the fact that the resulting discrete problem can be solved efficiently and that, via a generalization of the framework to perturbed systems in Sect. 5 the feedback is also piecewise constant and can be computed offline.

Let $\mathcal{P}$ be a finite partition of the state space X, i.e. a finite collection of pairwise disjoint subsets of X whose union covers X. For $x \in X$, let $\pi(x) \in \mathcal{P}$ denote the partition element that contains x. In what follows, we identify any subset $\{P_1, \ldots, P_k\}$ of $\mathcal{P}$ with the corresponding subset $\bigcup_{i=1,\ldots k} P_i$ of X.

Let $\mathbb{R}^{\mathcal{P}} \subset \mathbb{R}^X = \{v : X \to \mathbb{R}\}$ be the subspace of real valued functions on X which are piecewise constant on the elements of $\mathcal{P}$. Using the projection

$$\psi[v](x) := \inf_{x' \in \pi(x)} v(x'), \tag{10}$$

from $\mathbb{R}^X$ onto $\mathbb{R}^{\mathcal{P}}$, we define the *discretized Bellman operator*

$$L_{\mathcal{P}} := \psi \circ L.$$

Again, this operator has a unique fixed point $V_{\mathcal{P}}$ satisfying the boundary condition $V_{\mathcal{P}}(0) = 0$, which will serve as an approximation to the exact value function V.

Explicitely, the discretized operator reads

$$L_{\mathcal{P}}[v](x) = \inf_{x' \in \pi(x)} \left\{ \inf_{u \in U} \{g(x', u) + v(f(x', u))\} \right\}.$$

and $V_{\mathcal{P}}$ satisfies the optimality principle

$$V_{\mathcal{P}}(x) = \inf_{x' \in \pi(x), u \in U} \{g(x', u) + V_{\mathcal{P}}(f(x', u))\}. \tag{11}$$

Recalling that $V_{\mathcal{P}}$ is constant on each element P of the partition $\mathcal{P}$, we write $V_{\mathcal{P}}(P)$ in order to denote the value $V_{\mathcal{P}}(x)$ for some $x \in P$. We can rewrite (11) as

$$V_{\mathcal{P}}(x) = \min_{P} \inf_{(x', u)} \{g(x', u) + V_{\mathcal{P}}(P)\} \tag{12}$$

where the min is taken over all $P \in \mathcal{P}$ for which $P \cap f(\pi(x), U) \neq \emptyset$ and the inf over all pairs $x' \in \pi(x)$, $u \in U$ such that $f(x', u) \in P$. Now define the multivalued map $\mathcal{F} : \mathcal{P} \rightrightarrows \mathcal{P}$,

$$\mathcal{F}(P) = \{P' \in \mathcal{P} : P' \cap f(P, U) \neq \emptyset\} \tag{13}$$

and the cost function $\mathcal{G} : \mathcal{P} \times \mathcal{P} \to [0, \infty)$,

$$\mathcal{G}(P, P') = \inf_{u \in U} \{g(x, u) \mid x \in P, f(x, u) \in P'\}. \tag{14}$$

Equation (12) can then be rewritten as

$$V_{\mathcal{P}}(P) = \min_{P' \in \mathcal{F}(P)} \{\mathcal{G}(P, P') + V_{\mathcal{P}}(P')\}.$$

Graph Interpretation. It is useful to think of this reformulation of the discrete optimality principle in terms of a directed weighted graph $G_{\mathcal{P}} = (\mathcal{P}, E_{\mathcal{P}})$. The nodes of the graph are given by the elements of the partition $\mathcal{P}$, the edges are defined by the map $\mathcal{F}$: there is an edge $(P, P') \in E_{\mathcal{P}}$ whenever $P' \in \mathcal{F}(P)$ and the edge $e = (P, P')$ is weighted by $\mathcal{G}(e) := \mathcal{G}(P, P')$, cf. Fig. 1. In fact, the value $V_{\mathcal{P}}(P)$ is the length $\mathcal{G}(p) := \sum_{k=1}^{m} \mathcal{G}(e_k)$ of the shortest path $p = (e_1, \ldots, e_m)$ from P to the element $\pi(0)$ containing 0 in this graph. As such, it can be computed by (e.g.) the following algorithm with complexity $\mathcal{O}(|\mathcal{P}| \log(|\mathcal{P}|) + |E|)$:

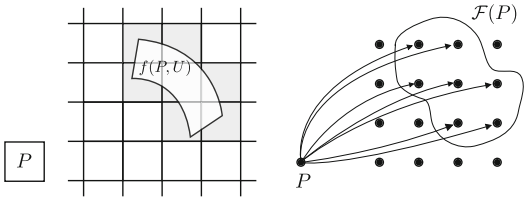

Fig. 1. Partition of phase space, image of an element (left) and corresponding edges in the induced graph (right).

Algorithm DIJKSTRA [5]

for each $P \in \mathcal{P}$: $V(P) := \infty$; $V(\pi(0)) := 0$; $\mathcal{Q} := \mathcal{P}$
while $\mathcal{Q} \neq \emptyset$
    $P := \operatorname{argmin}_{P' \in \mathcal{Q}} V(P')$
    $\mathcal{Q} := \mathcal{Q} \backslash \{P\}$
    for each $Q \in \mathcal{P}$ with $(Q, P) \in E_{\mathcal{P}}$
        if $V(Q) > \mathcal{G}(Q, P) + V(P)$ then
            $V(Q) := \mathcal{G}(Q, P) + V(P)$

□

The time complexity of this algorithm depends on the data structure which is used in order to store the set $\mathcal{Q}$. In our implementation we use a binary heap which leads to a complexity of $\mathcal{O}((|\mathcal{P}| + |E|) \log |\mathcal{P}|)$. This can be improved to $\mathcal{O}(|\mathcal{P}| \log |\mathcal{P}| + |E|)$ by employing a Fibonacci heap.

A similar idea is at the core of *fast marching methods* [16,18] and *ordered upwind methods* [17].

Implementation. We use the approach from [3,4] as implemented in GAIO in order to construct a cubical partition of X, stored in binary tree. For the construction of the edges and their weights, we use a finite set of sample points $\tilde{U} \subset U$ and $\tilde{P} \subset P$ for each $P \in \mathcal{P}$ and compute the approximate image

$$\tilde{\mathcal{F}}(P) = \{P' \in \mathcal{P} : P' \cap f(\tilde{P}, \tilde{U}) \neq \emptyset\}, \tag{15}$$

so that the set of edges is approximately given by all pairs (P, P') for which $P' \in \tilde{\mathcal{F}}(P)$. Correspondingly, the weight of the edge (P, P') is approximated by

$$\tilde{\mathcal{G}}(P, P') = \min_{(x,u)\in\tilde{P}\times\tilde{U}} \{g(x, u) \mid f(x, u) \in P'\}.$$

This construction of the graph via the mapping of sample points indeed constitutes the main computational effort in computing the discrete value function. It might be particularly expensive if the control system f is given by the control flow of a continuous time system. Note, however, that a sampling of the system will be required in any method that computes the value function. In fact, in standard methods like value iteration, the same point might be sampled multiple times (in contrast to the approach described here).

Certainly, this approximation of the box images introduces some error, i.e. one always has that $\tilde{\mathcal{F}}(P) \subset \mathcal{F}(P)$, but typically $\mathcal{F}(P) \subsetneqq \tilde{\mathcal{F}}(P)$. In experiments, one often increases the number of sample points until the result of the computation stabilizes. Alternatively, in the case that one is interested in a *rigorous* computation, either techniques based on Lipschitz estimates [13] or interval arithmetic [19] can be employed.

Example 1 ***(A simple 1D system).*** Consider the system

$$x_{k+1} = x_k + (1 - a)u_k x_k, \quad k = 0, 1, \ldots, \tag{16}$$

where $x_k \in X = [0, 1], u_k \in U = [-1, 1]$ and $a \in (0, 1)$ is a fixed parameter. Let

$$g(x, u) = (1 - a)x,$$

such that the optimal control policy is to steer to the origin as fast as possible, i.e. for every x, the optimal sequence of controls is $(-1, -1, \ldots)$. This yields $V(x) = x$ as the value function.

For the experiment, we consider $a = 0.8$ and use partitions of equally sized subintervals of $[0, 1]$. The edge weights (14) are approximated by minimizing over 100 equally spaced sample points in each subinterval and 10 equally spaced points in U. Figure 2 shows the exact and two discrete value functions, resulting from running the code in Fig. 3 in Matlab (requires the GAIO toolbox[2]).

[2] Available at http://www.github.com/gaioguy/gaio.

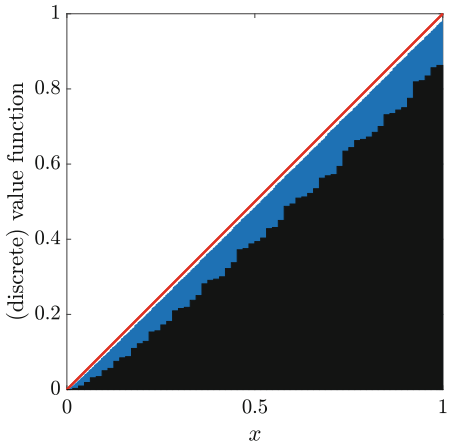

Fig. 2. Exact (red) and discrete value functions for the simple example on partitions of 64 (black) and 1024 (blue) intervals.

```
a = 0.8;
f = @(x,u) [x + (1-a).*x.*u, (1-a)*x];      % control system
X = linspace(-1,1,100)';                    % state samples
U = linspace(-1,1,10)';                     % control samples

depth = 6; c = 0.5; r = 0.5;
tree = Tree(c, r);                          % construct full tree
subdivide(tree, depth);                     % construct partition
A = dpgraph(tree, f, X, U, depth);          % compute graph
D = tree.search(0, depth);                  % find destination box
[V,~] = dijkstra(A', D);                    % compute value function

n = 2^depth; dx = 1/n; x = linspace(dx/2,1-dx/2,n);
clf;  plot(0:1,0:1,'r'); hold on; bar(x,V,1);
axis tight; axis square;
xlabel('$x$'); ylabel('(discrete) value function');
```

Fig. 3. Code: value function for a simple 1d system.

4.1 The Discrete Value Function

Proposition 1 [14]. *For every partition* $\mathcal{P}$ *of* X, $V_{\mathcal{P}}(x) \le V(x)$ *for all* $x \in X$.

Proof. The statement obviously holds for $x \in X$ with $V(x) = \infty$. So let $x \in S_0$, i.e. $V(x) < \infty$. For arbitrary $\varepsilon > 0$, let $\boldsymbol{u} = (u_0, u_1, \ldots) \in \mathcal{U}(x)$ be a control sequence such that $J(x, \boldsymbol{u}) < V(x) + \varepsilon$ and $(x_k(x, \boldsymbol{u}))_k$ the associated trajectory of (4). Consider the path

$$(e_1, \ldots, e_m), \quad e_k = (\pi(x_{k-1}), \pi(x_k)), \quad k = 1, \ldots, m,$$

where $x = x_0$ and and m is minimal with $x_m \in \pi(0)$. The length of this path is

$$\sum_{k=1}^{m} \mathcal{G}(e_k) = \sum_{k=1}^{m} \inf_{u\in U}\{g(x,u) \mid x \in \pi(x_{k-1}), f(x,u) \in \pi(x_k)\}$$
$$\leq \sum_{k=1}^{m} g(x_{k-1}, u_{k-1}) \leq \sum_{k=1}^{\infty} g(x_{k-1}, u_{k-1}) = J(x, \boldsymbol{u}),$$

yielding the claim. □

This property immediately yields an efficient aposteriori error estimate for $V_{\mathcal{P}}$: For $x \in S_0$ consider

$$e(x) = \inf_{u\in U}\{g(x,u) + V_{\mathcal{P}}(f(x,u))\} - V_{\mathcal{P}}(x). \tag{17}$$

Note that $e(x) \geq 0$. Since

$$V(x) - V_{\mathcal{P}}(x) = \inf_{u\in U}\{g(x,u) + V(f(x,u))\} - V_{\mathcal{P}}(x)$$
$$\geq \inf_{u\in U}\{g(x,u) + V_{\mathcal{P}}(f(x,u))\} - V_{\mathcal{P}}(x) = e(x),$$

we obtain

Proposition 2. *The function $e : S_0 \to [0,\infty)$ is a lower bound on the error between the true value function V and its approximation $V_{\mathcal{P}}$:*

$$e(x) \leq V(x) - V_{\mathcal{P}}(x), \quad x \in S_0.$$

Now consider a sequence $(\mathcal{P}^{(\ell)})_{\ell\in\mathbb{N}}$ of partitions of X which is nested in the sense that for all ℓ and every $P \in \mathcal{P}^{(\ell+1)}$ there is a $P' \in \mathcal{P}^{(\ell)}$ such that $P \subset P'$. For the next proposition recall that $S \subset X$ is the set of initial conditions that can be asymptotically controlled to 0.

Proposition 3 [14]**.** *For fixed $x \in S$, the sequence $(V_{\mathcal{P}^{(\ell)}}(x))_{\ell\in\mathbb{N}}$ is monotonically increasing.*

Proof. For $x \in S$, the value $V_{\mathcal{P}^{(\ell)}}(x)$ is the length of a shortest path $p = (e_1, \ldots, e_m)$, $e_k \in E_{\mathcal{P}^{(\ell)}}$, connecting $\pi(x)$ to $\pi(0)$ in $\mathcal{P}^{(\ell)}$. Suppose that the claim was not true, i.e. for some ℓ there are shortest paths p in $G_{\mathcal{P}^{(\ell)}}$ and p' in $G_{\mathcal{P}^{(\ell+1)}}$ such that $\mathcal{G}(p') < \mathcal{G}(p)$. Using p', we are going to construct a path $\tilde{p}$ in $G_{\mathcal{P}^{(\ell)}}$ with $\mathcal{G}(\tilde{p}) < \mathcal{G}(p)$, contradicting the minimality of p: Let $p' = (e'_1, \ldots, e'_{m'})$, with $e'_k = (P'_{k-1}, P'_k) \in E_{\mathcal{P}^{(\ell+1)}}$. Hence, $f(P'_{k-1}, U) \cap P'_k \neq \emptyset$, for $k = 1, \ldots, m'$. Since the partitions $\mathcal{P}^{(\ell)}$ are nested, there are sets $\tilde{P}_k \in \mathcal{P}^{(\ell)}$ such that $P'_k \subset \tilde{P}_k$ for $k = 0, \ldots, m'$. Thus, $f(\tilde{P}_{k-1}, U) \cap \tilde{P}_k \neq \emptyset$, i.e. $\tilde{e}_k = (\tilde{P}_{k-1}, \tilde{P}_k)$ is an edge in $E_{\mathcal{P}^{(\ell)}}$ and $\tilde{p} = (\tilde{e}_1, \ldots, \tilde{e}_{m'})$ is a path in $G_{\mathcal{P}^{(\ell)}}$. Furthermore, for $k = 1, \ldots, m'$,

$$\mathcal{G}(\tilde{e}_k) = \inf_{u\in U}\{g(x,u) \mid x \in \tilde{P}_{k-1}, f(x,u) \in \tilde{P}_k\}$$
$$\leq \inf_{u\in U}\{g(x,u) \mid x \in P'_{k-1}, f(x,u) \in P'_k\} = \mathcal{G}(e'_k).$$

This yields $\mathcal{G}(\tilde{p}) \leq \mathcal{G}(p') < \mathcal{G}(p)$, contradicting the minimality of p. □

So far we have shown that for every $x \in S$ we have a monotonically increasing sequence $(V_{\mathcal{P}^{(\ell)}}(x))_{\ell\in\mathbb{N}}$, which is bounded by $V(x)$ due to Proposition 1. The following theorem states that for points $x \in S$ the limit is indeed $V(x)$ if the maximal diameter of the partition elements goes to 0. For some finite partition $\mathcal{P}$ of X, let $\operatorname{diam}(\mathcal{P}) := \max_i \operatorname{diam}(P_i)$ be the *diameter of the partition* $\mathcal{P}$.

Theorem 1 [14]. *If* $\operatorname{diam}(\mathcal{P}^{(\ell)}) \to 0$ *then* $V_{\mathcal{P}^{(\ell)}}(x) \to V(x)$ *as* $\ell \to \infty$ *for all* $x \in S$.

4.2 The Discrete Feedback

Recall that an optimally stabilizing feedback can be constructed using the (exact) value function for the problem (cf. (8)). We will use this idea, replacing V by its approximation $V_{\mathcal{P}}$: using $\tilde{U}$ from (15)[3], for $x \in S$ we define

$$u_{\mathcal{P}}(x) := \operatorname*{argmin}_{u\in\tilde{U}} \{g(x,u) + V_{\mathcal{P}}(f(x,u))\} \tag{18}$$

(the minimum exists because $\tilde{U}$ is a finite set) and consider the closed loop system

$$x_{k+1} = f(x_k, u_{\mathcal{P}}(x_k)), \quad k = 0, 1, \ldots. \tag{19}$$

The following theorems state in which sense this feedback is stabilizing and approximately optimal. Let again $(\mathcal{P}^{(\ell)})_{\ell\in\mathbb{N}}$ be a nested sequence of partitions of X and $D \subseteq S$, $0 \in D$, an open set with the property that for each $\varepsilon > 0$ there exists $\ell_0(\varepsilon) > 0$ such that

$$\max_{x\in D} |V(x) - V_{\mathcal{P}^{(\ell)}}(x)| \le \varepsilon, \quad \text{for } \ell \ge \ell_0(\varepsilon).$$

Let further $c > 0$ be the largest value such that

$$V^{-1}_{\mathcal{P}^{(1)}}([0,c]) \subset D.$$

Note that by Proposition 3 this implies that $V^{-1}_{\mathcal{P}^{(\ell)}}([0,c]) \subset D$ for all $\ell \in \mathbb{N}$.

Theorem 2 [7]. *Under the assumptions above, there exists* $\varepsilon_0 > 0$ *and a function* $\delta : \mathbb{R} \to \mathbb{R}$ *with* $\lim_{\alpha\to 0} \delta(\alpha) = 0$, *such that for all* $\varepsilon \in (0, \varepsilon_0]$, *all* $\ell \ge \ell_0(\varepsilon/2)$, *all* $\eta \in (0,1)$ *and all* $x_0 \in V^{-1}_{\mathcal{P}^{(\ell)}}([0,c])$ *the trajectory* $(x_k)_k$ *generated by the closed loop system* (19) *with feedback* $u_{\mathcal{P}^{(\ell)}}$ *satisfies*

$$V(x_k) \le \max\left\{ V(x_0) - (1-\eta)\sum_{j=0}^{k-1} g(x_j, u_{\mathcal{P}(\ell)}(x_j)),\ \ \delta(\varepsilon/\eta) + \varepsilon \right\}.$$

[3] The subsequent statements remain true if we replace $\tilde{U}$ by any set $\widehat{U} \subset U$ with $\tilde{U} \subset \widehat{U}$ for which the argmin in (18) exists.

This apriori estimate shows in which sense the feedback $u_{\mathcal{P}}$ approximately yields optimal performance. However, the theorem does not give information about the partition $\mathcal{P}$ which is needed in order to achieve a desired level of accuracy. This can be achieved by employing the error function e from above.

Consider some partition $\mathcal{P}$ of X. Let $g_0(x) := \inf_{u\in U} g(x,u)$ and $C_\varepsilon(\mathcal{P}) := \{x \in V_{\mathcal{P}}^{-1}([0,c]) \,|\, g_0(x) \le \varepsilon\}$ and define $\delta(\varepsilon) := \sup_{x\in C_\varepsilon} V(x)$. Note that if V is continuous at $T = \{0\}$ then $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$ because $C_\varepsilon(\mathcal{P})$ shrinks down to 0 since g and thus g_0 are continuous.

Theorem 3 [7]. *Assume that for some $\varepsilon > 0$ and some $\eta \in (0,1)$, the error function e satisfies*

$$e(x) \le \max\{\eta g_0(x), \varepsilon\} \quad \text{for all } x \in V_{\mathcal{P}}^{-1}([0,c]). \tag{20}$$

Then, for each $x_0 \in V_{\mathcal{P}}^{-1}([0,c]$, the trajectory $(x_k)_k$ generated by the closed loop system (19) satisfies

$$V_{\mathcal{P}}(x_k) \le \max\left\{ V_{\mathcal{P}}(x_0) - (1-\eta)\sum_{j=0}^{k-1} g(x_j, u_{\mathcal{P}}(x_j)),\ \ \delta(\varepsilon/\eta) + \varepsilon \right\}. \tag{21}$$

Example 2 ***(An inverted pendulum).*** We consider a model for an inverted pendulum on a cart, cf. [7,14]. We ignore the dynamics of the cart, and so we only have one degree of freedom, namely the angle $\varphi \in [0, 2\pi]$ between the pendulum and the upright vertical. The origin $(\varphi, \dot\varphi) = (0,0)$ is an unstable equilibrium (with the pendulum pointing upright) which we would like to stabilize. The model reads

$$\left(\tfrac{4}{3} - m_r \cos^2\varphi\right)\ddot\varphi + \tfrac{m_r}{2}\dot\varphi^2 \sin 2\varphi - \tfrac{g}{\ell}\sin\varphi = -u\ \tfrac{m_r}{m\ell}\cos\varphi, \tag{22}$$

where $m = 2$ is the mass of the pendulum, $M = 8$ the mass of the cart, $m_r = m/(m+M)$, $\ell = 0.5$ the length of the pendulum and $g = 9.8$ the gravitational constant. We consider the discrete time control system (4) with $f(x,u) = \Phi^t(x,u)$, $x = (\varphi, \dot\varphi)$, for $t = 0.1$, where $\Phi^t(x,u)$ denotes the controlled flow of (22) with constant control input $u(\tau) = u$ for $\tau \in [0,t]$. For the instantaneous cost function we choose

$$g(x,u) = \int_0^t q(\Phi^\tau(x,u), u)\ d\tau,$$

with the quadratic cost $q(x,u) = \frac{1}{2}\left(0.1\varphi^2 + 0.05\dot\varphi^2 + 0.01u^2\right)$.

We use the classical Runge-Kutta scheme of order 4 with step size 0.02 in order to approximate Φ^t, choose $X = [-8,8] \times [-10,10]$ as state space for $x = (\varphi, \dot\varphi)$, which we partition into $2^9 \times 2^9$ boxes of equal size, and $U = [-64, 64]$ as the control space. In approximating the graph's edges and their weights, we map an equidistant grid of 3×3 points on each partition box, choosing from 17 equally spaced values in U.

Figure 4 shows the discrete value function as well as the trajectory generated by the discrete feedback for the initial value $(3.1, 0.1)$, as computed by the GAIO code in Fig. 6. As shown on the right of this figure, the discrete value function does not decrease monotonically along the feedback trajectory, indicating that the assumptions of Theorem 3 are not satisfied. And indeed, as shown in Fig. 5, this trajectory repeatedly moves through regions in state space where the error function e is not smaller than g_0. In fact, on a coarser partition ($2^7 \times 2^7$ boxes), the discrete feedback (18) is not even stabilizing this initial condition any more. We will adress this deficiency in the next sections.

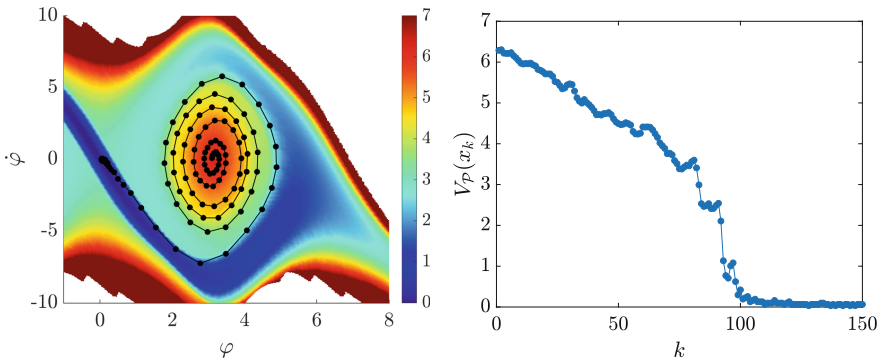

Fig. 4. Left: Discrete value function and feedback trajectory for the inverted pendulum. Right: Behaviour of the discrete value function along the feedback trajectory.

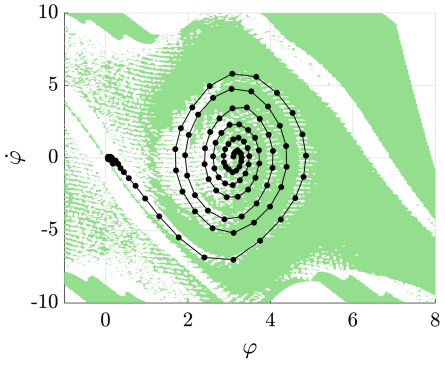

Fig. 5. Inverted pendulum: region where $e(x) < g_0(x)$ (green) and feedback trajectory.

```
m = 2; M = 8; m_r = m/(m+M); l = 0.5; g = 9.8;
q1 = 0.1; q2 = 0.05; r0 = 0.01;
v = @(x,u) [ x(:,2), ...                                % vector field & cost
             (g/l*sin(x(:,1)) - 0.5*m_r*x(:,2).^2.*sin(2*x(:,1)) - ...
             m_r/(m*l)*u.*cos(x(:,1)))./(4.0/3.0 - m_r*cos(x(:,1)).^2), ...
             0.5*( q1*(x(:,1).^2) + q2*(x(:,2).^2) + r0*u.^2 )];
h = 0.02; steps = 5;                                    % step size, # of steps
f = @(x,u) rk4u(v,[x zeros(size(x,1),1)],u,h,steps);% control system
n = 2; x1 = linspace(-1,1,n)';
[XX,YY] = meshgrid(x1,x1); X = [ XX(:) YY(:) ];     % sample points
U = linspace(-64,64,17)';                               % control samples

depth = 18; c = [0 0]; r = [8 10];
tree = Tree(c, r);
subdivide(trre, depth);                                 % construct partition
G = dpgraph(tree, f, X, U, depth);                      % compute graph
dest = tree.search([0;0], depth);                       % target set
[V,~] = dijkstra(G', dest);                             % discrete value function

V(find(V == Inf)) = NaN; V(find(V > 7)) = 7;
clf; boxplot2(tree, 'density', V);                      % plot value function
colorbar; colormap jet; shading flat; axis('tight')
xlabel('$\varphi$'); ylabel('$\dot\varphi$');

%% discrete feedback, trajectory of the closed loop system
x(1,:) = [3.1, 0.1];
for k = 1:200
  fxc = f(ones(size(U))*x(k,:), U)';    % map current point under all controls
  bn = tree.search(fxc(1:2,:),depth)';  % determine corresp. boxes
  [v, k_min] = min(fxc(3,:) + V(bn));   % determine minimizing control
  V_fb(k) = V(bn(k_min));               % value at next point
  x(k+1,:) = fxc(1:2,k_min);            % next iterate
end
hold on; plot(x(:,1),x(:,2),'k.-','linewidth',1,'markersize',22);
axis([-0.1 6 -8 8]);
figure(2); plot(V_fb,'.-','linewidth',1,'markersize',22)
xlabel('$k$'); ylabel('$V_\mathcal{P}(x_k)$');
```

Fig. 6. Code: discrete value function for the inverted pendulum

5 The Optimality Principle for Perturbed Systems

Let us now return to the full problem from Sect. 2 of optimally stabilizing the discrete time perturbed control system

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0, 1, \ldots. \tag{23}$$

subject to an instantaneous cost $g(x_k, u_k)$. For the convergence statements later, we assume $f : X \times U \times W \to X$ and $g : X \times U \to [0, \infty)$ to be continuous and $X \subset \mathbb{R}^d, U \subset \mathbb{R}^m$ and $W \subset \mathbb{R}^\ell$ to be compact. More general spaces will be discussed in Sect. 8. For a given initial state $x_0 \in X$, a control sequence $\boldsymbol{u} = (u_k)_{k\in\mathbb{N}} \in U^{\mathbb{N}}$ and a perturbation sequence $\boldsymbol{w} = (w_k)_{k\in\mathbb{N}} \in W^{\mathbb{N}}$, we obtain the *trajectory* $(x_k(x, \boldsymbol{u}, \boldsymbol{w}))_{k\in\mathbb{N}}$ satisfying (23) while the associated accumulated cost is given by

$$J(x, \boldsymbol{u}, \boldsymbol{w}) = \sum_{k=0}^{\infty} g(x_k(x, \boldsymbol{u}, \boldsymbol{w}), u_k).$$

Recall that our goal is to derive a *feedback* $u : S \to U$, $S \subset X$, that *stabilizes* the closed loop system

$$x_{k+1} = f(x_k, u(x_k), w_k), \quad k = 0, 1, 2, \ldots \tag{24}$$

for any perturbation sequence $(w_k)_k$, i.e. for every trajectory $(x_k(x_0, \boldsymbol{w}))_k$ of (24) with $x_0 \in S$ and $\boldsymbol{w} \in W^{\mathbb{N}}$ arbitrary, we have $x_k \to T$ as $k \to \infty$, where $T \subset S$ is a given *target set*, and the *accumulated cost* $\sum_{k=0}^{\infty} g(x_k, u(x_k))$ is minimized.

The problem formulation can be interpreted as describing a *dynamic game* (see e.g. [6]), where at each step of the iteration (23) two *players* choose a control u_k and a perturbation w_k, respectively. The goal of the controlling player is to minimize J, while the perturbing player wants to maximize it. We assume that the controlling player chooses u_k first and that the perturbing player knows u_k when choosing w_k. We further assume that the perturbing player cannot foresee future choices of the controlling player. This can be formalized by restricting the possible $\boldsymbol{w}$ to

$$\boldsymbol{w} = \beta(\boldsymbol{u}),$$

where $\beta : U^{\mathbb{N}} \to W^{\mathbb{N}}$ is a *nonanticipating strategy*, i.e. a strategy satisfying

$$u_k = u'_k \quad \forall k \le K \quad \Rightarrow \quad \beta_k(\boldsymbol{u}) = \beta_k(\boldsymbol{u}') \quad \forall k \le K$$

for any $\boldsymbol{u} = (u_k)_k, \boldsymbol{u}' = (u'_k)_k \in U^{\mathbb{N}}$. We denote by $\mathcal{B}$ the set of all nonanticipating strategies $\beta : U^{\mathbb{N}} \to W^{\mathbb{N}}$.

The control task is finished once we are in T, we therefore assume that T is compact and robustly forward invariant, i.e. for all $x \in T$ there is a control $u \in U$ such that $f(x, u, w) \subset T$ for all $w \in W$, that $g(x, u) = 0$ for all $x \in T$, $u \in U$ and $g(x, u) > 0$ for all $x \notin T$, $u \in U$.

Our construction of the feedback $u : S \to U$ will be based on the *upper value function* $V : X \to [0, \infty]$,

$$V(x) = \sup_{\beta \in \mathcal{B}} \inf_{\boldsymbol{u} \in U^{\mathbb{N}}} J(x, \boldsymbol{u}, \beta(\boldsymbol{u})), \tag{25}$$

of the game (23), which is finite on the set $S_0 := \{x \in X \mid V(x) < \infty\}$. The upper value function satisfies the *optimality principle* [9]

$$V(x) = \inf_{u \in U} \left[g(x, u) + \sup_{w \in W} V(f(x, u, w)) \right], \quad x \in S_0. \tag{26}$$

The right hand side $L[v](x) = \inf_{u \in U} [g(x, u) + \sup_{w \in W} v(f(x, u, w))]$ of this fixed point equation again defines a *dynamic programming operator* $L : \mathbb{R}^X \to \mathbb{R}^X$. The upper value function is the unique fixed point of L satisying the boundary condition $V(x) = 0$, $x \in T$. Like in the unperturbed case, using the upper value function V, one can construct the feedback $u : S_0 \to U$,

$$u(x) := \operatorname*{argmin}_{u \in U} \left[g(x, u) + \sup_{w \in W} V(f(x, u, w)) \right], \tag{27}$$

whenever this minimum exists.

6 A Discrete Optimality Principle for Perturbed Systems

Analogously to the discretization in Sect. 4 we now derive a discrete version of (26), cf. [9]. Again, to this end, we will approximate the upper value function by a function which is piecewise constant on the elements of some partition of X. This approach will lead to a directed weighted hypergraph instead of the ordinary directed graph in Sect. 4 and, again, the approximate upper value function can be computed by an associated shortest path algorithm.

Let $\mathcal{P}$ be a finite partition of X. Using the projection (10), the *discretized dynamic game operator* $L_{\mathcal{P}} : \mathbb{R}^{\mathcal{P}} \to \mathbb{R}^{\mathcal{P}}$ is defined by

$$L_{\mathcal{P}} := \psi \circ L.$$

Again, this operator has a unique fixed point $V_{\mathcal{P}}$ satisfying the boundary condition $V_{\mathcal{P}}(x) = 0$, $x \in T$, which will serve as an approximation to the exact value function V.

Explicitely, the discretized operator reads

$$L_{\mathcal{P}}[v](x) = \inf_{x' \in \pi(x)} \left(\inf_{u \in U} \left[g(x', u) + \sup_{w \in W} v(f(x', u, w)) \right] \right)$$

and $V_{\mathcal{P}}$ satisfies the optimality principle

$$V_{\mathcal{P}}(x) = \inf_{x' \in \pi(x), u \in U} \left[g(x', u) + \sup_{w \in W} V_{\mathcal{P}}(f(x', u, w)) \right]. \tag{28}$$

Note that since $V_{\mathcal{P}}$ is constant on each partition element, we can rewrite this as

$$V_{\mathcal{P}}(x) = \inf_{x' \in \pi(x), u \in U} \left[g(x', u) + \sup_{P' \in \mathcal{F}(x', u)} V_{\mathcal{P}}(P') \right],$$

where

$$\mathcal{F}(x', u) = \{P \in \mathcal{P} \mid f(x', u, w) \in P \text{ for some } w \in W\}.$$

Since the partition $\mathcal{P}$ is finite, there are only finitely many possible sets $\mathcal{F}(x', u)$ and we can further rewrite (28) as

$$V_{\mathcal{P}}(x) = \min_{\mathcal{N}} \inf_{(x', u)} \left[g(x', u) + \sup_{P' \in \mathcal{N}} V_{\mathcal{P}}(P') \right],$$

where the min is taken over all collections $\mathcal{N} \in \{\mathcal{F}(x', u) \mid x' \in \pi(x), u \in U\}$ and the inf over all (x', u) such that $\mathcal{F}(x', u) = \mathcal{N}$. Now define the multivalued map $\mathcal{F} : \mathcal{P} \rightrightarrows 2^{\mathcal{P}}$,

$$\mathcal{F}(P) = \{\mathcal{F}(x, u) : (x, u) \in P \times U\},$$

and the cost function

$$\mathcal{G}(P, \mathcal{N}) = \inf_{u \in U} \{g(x, u) : x \in P, \mathcal{F}(x, u) = \mathcal{N}\}.$$

Equation (28) can then be rewritten as

$$V_{\mathcal{P}}(P) = \min_{\mathcal{N}\in\mathcal{F}(P)} \left[\mathcal{G}(P,\mathcal{N}) + \sup_{P'\in\mathcal{N}} V_{\mathcal{P}}(P') \right],$$

Graph Interpretation. Like in the unperturbed case, we can think of this reformulation of the optimality principle in terms of a graph. More precisely, we have a directed hypergraph $(\mathcal{P}, E_{\mathcal{P}})$ with the set $E \subset \mathcal{P} \times 2^{\mathcal{P}}$ of directed hyperedges given by

$$E_{\mathcal{P}} = \{(P,\mathcal{N}) \mid \mathcal{N} = \mathcal{F}(x,u) \text{ for some } (x,u) \in P \times U\},$$

and each edge $(P,\mathcal{N})$ is weighted by $\mathcal{G}(P,\mathcal{N})$, c.f. Fig. 7. The discrete upper value function $V_{\mathcal{P}}(P)$ is the length of a shortest path from P to some element P' which has a nonempty intersection with the target set T (and, thus, by the boundary condition, $V_{\mathcal{P}}(P') = 0$).

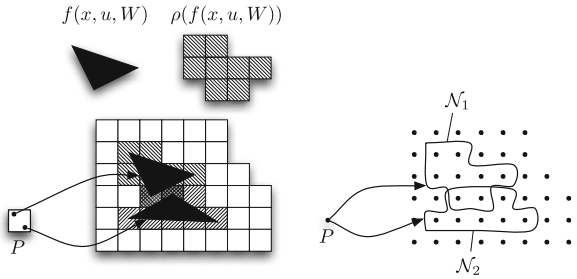

Fig. 7. Illustration of the construction of the hypergraph.

Shortest Paths in Hypergraphs. Algorithm 1 can be generalized to the hypergraph case, cf. [9,20]. To this end, we modify lines 5–7 such that the maximization over the perturbations is taken into account:

for each $(Q,\mathcal{N}) \in E_{\mathcal{P}}$ with $P \in \mathcal{N}$
  if $V(Q) > \mathcal{G}(Q,\mathcal{N}) + \max_{N\in\mathcal{N}} V(N)$ then
    $V(Q) := \mathcal{G}(Q,\mathcal{N}) + \max_{N\in\mathcal{N}} V(N)$

Note that during the while-loop of Algorithm 1,

$$V(P) \geq V(P') \quad \text{for all } P' \in \mathcal{P}\backslash\mathcal{Q}.$$

Thus, if $\mathcal{N} \subset \mathcal{P}\backslash\mathcal{Q}$, then $\max_{N\in\mathcal{N}} V(N) = V(P)$, and the value of the node Q will never be decreased again. On the other hand, if $\mathcal{N} \not\subset \mathcal{P}\backslash\mathcal{Q}$, then the value of Q will be further decreased at a later time – and thus we can save on changing it in the current iteration of the while-loop. We can therefore save on the explicit maximization and replace lines 5–7 by

for each $(Q, \mathcal{N}) \in E_{\mathcal{P}}$ with $P \in \mathcal{N}$
  if $\mathcal{N} \subset \mathcal{P} \backslash \mathcal{Q}$ then
    if $V(Q) > \mathcal{G}(Q, \mathcal{N}) + V(P)$ then
      $V(Q) := \mathcal{G}(Q, \mathcal{N}) + V(P)$

The overall algorithm for the hypergraph case is as follows. Here, $\mathcal{T} := \{P \in \mathcal{P} \mid P \cap T \neq \emptyset\}$ is the set of target nodes.

Algorithm MINMAX-DIJKSTRA

for each $P \in \mathcal{P}$: $V(P) := \infty$; for each $P \in \mathcal{T}$: $V(P) := 0$; $\mathcal{Q} := \mathcal{P}$
while $\mathcal{Q} \neq \emptyset$
  $P := \operatorname{argmin}_{P' \in \mathcal{Q}} V(P')$
  $\mathcal{Q} := \mathcal{Q} \backslash \{P\}$
  for each $(Q, \mathcal{N}) \in E_{\mathcal{P}}$ with $P \in \mathcal{N}$
    if $\mathcal{N} \subset \mathcal{P} \backslash \mathcal{Q}$ then
      If $V(Q) > \mathcal{G}(Q, \mathcal{N}) + V(P)$ then
        $V(Q) := \mathcal{G}(Q, \mathcal{N}) + V(P)$

□

Time Complexity. In line 5, each hyperedge is considered at most N times, with N being a bound on the cardinality of the hypernodes $\mathcal{N}$. Additionally, we need to perform the check in line 6, which has linear complexity in N. Thus, the overall complexity of the minmax-Dijkstra algorithm is $\mathcal{O}(|\mathcal{P}| \log |\mathcal{P}| + |E| N (N + \log |\mathcal{P}|))$ (when using a binary heap for storing $\mathcal{Q}$), cf. [20].

Space Complexity. The storage requirement grows linearly with $|\mathcal{P}|$. This number, however, grows exponentially with the dimension of state space (if the entire state space is covered and under the assumption of uniformly large elements). The number of hyperedges is determined by the Lipschitz constant of f, the size of the hypernodes $\mathcal{N}$ will be given by the magnitude of the perturbation.

Implementation. We use the same approach as in the unperturbed case: A cubical partition is constructed hierarchically and stored in a binary tree. In order to approximate the set $E_{\mathcal{P}} \subset \mathcal{P} \times 2^{\mathcal{P}}$ of hyperedges, we choose finite sets $\tilde{P} \subset P$, $\tilde{U} \subset U$ and $\tilde{W} \subset W$ of sample points, set

$$\tilde{\mathcal{F}}(x, u) = \{P \in \mathcal{P} \mid f(x, u, w) \in P \text{ for some } w \in \tilde{W}\}$$

and compute

$$\tilde{\mathcal{F}}(P) := \{\tilde{\mathcal{F}}(x, u) : (x, u) \in \tilde{P} \times \tilde{U}\} \subset 2^{\mathcal{P}}$$

as an approximation to $\mathcal{F}(P)$. Correspondingly, the weight on the hyperedge $(P, \mathcal{N})$ is approximated by

$$\tilde{\mathcal{G}}(P, \mathcal{N}) = \min\{g(x, u) : (x, u) \in \tilde{P} \times \tilde{U}, \tilde{\mathcal{F}}(x, u) = \mathcal{N}\}.$$

Example: A simple 1D System. We reconsider system (16), adding a small perturbation at each time step:

$$x_{k+1} = x_k + (1 - a) u_k x_k + w_k, \quad k = 0, 1, \ldots,$$

with $x_k \in [0,1]$, $u_k \in [-1,1]$, $w_k \in [-\varepsilon,\varepsilon]$ for some $\varepsilon > 0$ and the fixed parameter $a \in (0,1)$. The cost function is still $g(x,u) = (1-a)x$ so that the optimal control policy is again $u_k = -1$ for all k, independently of the perturbation sequence. The optimal strategy for the perturbing player is to slow down the dynamics as much as possible, corresponding to $w_k = \varepsilon$ for all k. The dynamical system resulting from inserting the optimal strategies is

$$x_{k+1} = ax_k + \varepsilon, \quad k = 0, 1, \ldots.$$

This map has a fixed point at $x = \varepsilon/(1-a)$. In the worst case, i.e. $w_k = \varepsilon$ for all k, it is not possible to get closer than $\alpha_0 := \varepsilon/(1-a)$ to the origin. We therefore define $T = [0,\alpha]$ with $\alpha > \alpha_0$ as the target set. With

$$k(x) = \left\lceil \frac{\log \frac{\alpha-\alpha_0}{x-\alpha_0}}{\log a} \right\rceil + 1,$$

the exact optimal value function is

$$V(x) = (x-\alpha_0)\left(1 - a^{k(x)}\right) + \varepsilon k(x),$$

as shown in Fig. 8 for $a = 0.8$, $\varepsilon = 0.01$ and $\alpha = 1.1\alpha_0$. In that figure, we also show the approximate optimal value functions on partitions of $64, 256$ and 1024 intervals, respectively. In the construction of the hypergraph, we used an equidistant grid of ten points in each partition interval, in the control space and in the perturbation space.

6.1 Convergence

It is natural to ask whether the approximate value function converges to the true one when the element diameter of the underlying partition goes to zero. This has been proven pointwise on the stabilizable set S in the unperturbed case [14], as well as in an L^1-sense on S and an L^∞ sense on the domain of continuity in the perturbed case, assuming continuity of V on the boundary of the target set T [9]. The same reference also provides an analysis for state constrained problems. Here an additional robustness condition is needed, namely that the optimal value function changes continuously with respect to the L^p-norm for some $p \in \{1, \ldots, \infty\}$ if the state constraints are tightened. If this condition holds, then the convergence statement remains valid under state constraints, with L^∞ replaced by L^p.

Due to the construction of the discretization, the approximation $V_\mathcal{P}$ of the optimal value function is always less or equal than the true optimal value function. This is not necessarily a good property. For instance, for proving stability of the system controlled by the numerical feedback law it would be convenient if $V_\mathcal{P}$ was a Lyapunov function. Lyapunov functions, however, are supersolutions to the dynamic programming equation, rather than subsolutions as our $V_\mathcal{P}$. In order to overcome this disadvantage, in the next section we present a particular construction of a dynamic game in which the discretization error is treated as a perturbation.

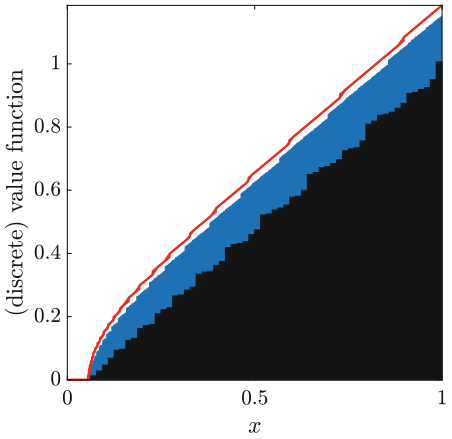

Fig. 8. Exact (red) and discrete upper value functions for the perturbed simple example on partitions of 64 (black) and 1024 (blue) intervals.

```
a = 0.8; ep = 0.01;
alpha0 = ep/(1-a); alpha = 1.1*alpha0;
f = @(x,u,w) [x + (1-a).*x.*u + w, (1-a)*x];        % control system
X = linspace(-1,1,5)';                               % state samples
U = linspace(-1,1,2)';                               % control samples
W = linspace(-ep,ep,3)';                             % perturbation samples

depth = 6; c = [0.5]; r = [0.5];
tree = Tree(c, r);                                   % construct full tree
subdivide(tree, depth);                              % construct partition

G = compute_hypergraph(tree, f, X, U, W, depth);     % construct hypergraph
Gt = trnsp_hgraph(G);                                % transpose hypergraph
T = tree.search_box([0],[alpha],depth);              % target boxes
V_P = minmax_dijkstra(G, Gt, T);                     % upper value function

n = 2^depth; dx = 1/n; x = linspace(dx/2,1-dx/2,n);
k = @(x) floor(log((alpha-alpha0)./(x-alpha0))/log(a))+1;
V = @(x) (x>alpha).*((x-alpha0).*(1-a.^k(x))+ep*k(x));% exact value function
bar(x,V_P,1,'k'); hold on; plot(x,V(x),'r');
axis tight; axis square; xlabel('$x$');
ylabel('(discrete) value function');
```

Fig. 9. Code: upper value function for the perturbed simple 1d system.

7 The Discretization as a Perturbation

As shown in Theorems 2 and 3, the discrete feedback (18) will practically stabilize the closed loop system (19) under suitable conditions. Our numerical experiment in Example 2, however, revealed that a rather fine partition might be needed in order to achieve stability. More generally, as we have seen in Fig. 4 (right), the discrete value function is not a Lyapunov function of the closed loop system in every case.

Construction of the Dynamic Game. In order to cope with this problem we are going to use the ideas on treating perturbed systems in Sect. 5 and 6. The idea is to view the discretization error as a perturbation of the original system. Under the discretization described in Sect. 4, the original map $(x,u) \mapsto f(x,u)$ is perturbed to

$$(x,u) \mapsto \hat{f}(x,u,w) := f(x+w,u), \quad x+w \in \pi(x).$$

Note that this constitutes a generalization of the setting in Sects. 5 and 6 since the perturbation space W here depends on the state, $W = W(x)$. Correspondingly, the associated cost function is

$$\hat{g}(x,u) = \sup_{x' \in \pi(x)} g(x',u). \tag{29}$$

Theorem 4 [8]. *Let V denote the value function (6) of the control system (f,g), $\hat{V}$ the value function (25) of the associated game $(\hat{f},\hat{g})$ and $V_{\mathcal{P}}$ the discrete value function (28) of $(\hat{f},\hat{g})$ on a given partition $\mathcal{P}$ with numerical target set $T_{\mathcal{P}} \subset \mathcal{P}$, $T = \{0\} \subset T_{\mathcal{P}}$. Then $V_{\mathcal{P}}(x) = \hat{V}(x)$ and*

$$V(x) - \max_{y \in T_{\mathcal{P}}} V(y) \le V_{\mathcal{P}}(x), \tag{30}$$

i.e. $V_{\mathcal{P}}$ is an upper bound for $V - \max V|_{T_{\mathcal{P}}}$. Furthermore, $V_{\mathcal{P}}$ satisfies

$$V_{\mathcal{P}}(x) \ge \min_{u \in U} \{g(x,u) + V_{\mathcal{P}}(f(x,u))\} \tag{31}$$

for all $x \in X \setminus T_{\mathcal{P}}$.

Proof. We first note that $\hat{V}$ is constant on the elements of $\mathcal{P}$: On $T_{\mathcal{P}}$, this is true since $T_{\mathcal{P}}$ is a union of partition elements by assumption. Outside of $T_{\mathcal{P}}$, by definition of the game $(\hat{f},\hat{g})$ we have

$$\hat{V}(x) = \inf_{u \in U} \left\{ \sup_{x' \in \pi(x)} g(x',u) + \sup_{x' \in f(\pi(x),u)} \hat{V}(x') \right\},$$

so that $\inf_{x' \in \pi(x)} \hat{V}(x') = \hat{V}(x)$. On the other hand, according to [9, Proposition 7.1] we have $V_{\mathcal{P}}(x) = \inf_{x' \in \pi(x)} \hat{V}(x')$, so that $V_{\mathcal{P}} = \hat{V}$.

Now for $x \notin T_{\mathcal{P}}$, Eq. (26) yields

$$\begin{aligned} \hat{V}(x) &= \inf_{u \in U} \sup_{x' \in \pi(x)} \left\{ g(x',u) + \hat{V}(f(x',u)) \right\} \\ &\ge \min_{u \in U} \left\{ g(x,u) + \hat{V}(f(x,u)) \right\} \end{aligned} \tag{32}$$

which shows (31).

In order to prove (30), we order the elements $P_1, P_2, \ldots \in \mathcal{P}$ such that $i \ge j$ implies $V_{\mathcal{P}}(P_i) \ge V_{\mathcal{P}}(P_j)$. Since $\inf_{u \in U} g(x,u) > 0$ for $x \ne 0$ by assumption,

$V_{\mathcal{P}}(P_i) = 0$ is equivalent to $P_i \subseteq T_{\mathcal{P}}$. By the ordering of the elements this implies that there exists $i^* \geq 1$ such that $P_i \subseteq T_{\mathcal{P}} \Leftrightarrow i \in \{1, \dots, i^*\}$ and thus (30) holds for $x \in P_1, \dots, P_{i^*}$. We now use induction: fix some $i \in \mathbb{N}$, assume (30) holds for $x \in P_1, \dots, P_{i-1}$ and consider $x \in P_i$. If $V_{\mathcal{P}}(P_i) = \infty$ there is nothing to show. Otherwise, since V satisfies the dynamic programming principle, using (32) we obtain

$$\begin{aligned} V(x) - \hat{V}(x) &\leq \inf_{u \in U} \{g(x,u) + V(f(x,u))\} - \min_{u \in U} \left\{ g(x,u) + \hat{V}(f(x,u)) \right\} \\ &\leq V(f(x,u^*)) - \hat{V}(f(x,u^*)), \end{aligned}$$

where $u^* \in U$ realizes the minimum in (32). Now, since $g(x,u^*) > 0$, we have $\hat{V}(f(x,u^*)) < \hat{V}(x)$ implying $f(x,u^*) \in P_j$ for some $j < i$. Since by the induction assumption the inequality in (30) holds on P_j, this implies that it also holds on P_i which finishes the induction step. □

The Feedback Is the Shortest Path. As usual, we construct the discrete feedback by

$$u_{\mathcal{P}}(x) := \operatorname*{argmin}_{u \in U} \left[\hat{g}(x,u) + \sup_{x' \in f(\pi(x),u)} V_{\mathcal{P}}(x') \right].$$

By construction, this feedback is constant on each partition element. Moreover, we can directly extract $u_{\mathcal{P}}$ from the minmax-Dijkstra algorithm: We associate the minimizing control value $\underline{u}(P, \mathcal{N})$ to each hyperedge $(P, \mathcal{N})$,

$$\underline{u}(P, \mathcal{N}) = \operatorname*{argmin}_{u \in U, \mathcal{F}(P) = \mathcal{N}} \left[\sup_{x \in P} g(x,u) \right]. \tag{33}$$

The feedback is then immediately given by

$$u_{\mathcal{P}}(x) = \underline{u}(\pi(x), \underline{\mathcal{N}}(\pi(x))), \tag{34}$$

where

$$\underline{\mathcal{N}}(P) = \operatorname*{argmin}_{\mathcal{N} \in \mathcal{F}(P)} \left\{ \mathcal{G}(P, \mathcal{N}) + \sup_{N \in \mathcal{N}} V_{\mathcal{P}}(N) \right\}$$

is defining the hypernode of minimal value adjacent to some node P in the hypergraph. The computation of $\underline{\mathcal{N}}(P)$ can be done on the fly within the minmax-Dijkstra Algorithm 2:

Algorithm. MINMAX-DIJKSTRA WITH FEEDBACK

for each $P \in \mathcal{P}$: $V(P) := \infty$, $\underline{\mathcal{N}}(P) := \emptyset$; for each $P \in \mathcal{T}$: $V(P) := 0$; $\mathcal{Q} := \mathcal{P}$
while $\mathcal{Q} \neq \emptyset$
  $P := \operatorname{argmin}_{P' \in \mathcal{Q}} V(P')$
  $\mathcal{Q} := \mathcal{Q} \backslash \{P\}$
  for each $(Q, \mathcal{N}) \in E_{\mathcal{P}}$ with $P \in \mathcal{N}$
    if $\mathcal{N} \subset \mathcal{P} \backslash \mathcal{Q}$ then
      if $V(Q) > \mathcal{G}(Q, \mathcal{N}) + V(P)$ then
        $V(Q) := \mathcal{G}(Q, \mathcal{N}) + V(P)$
        $\underline{\mathcal{N}}(Q) := \mathcal{N}$

□

Consequently, the discrete feedback $\underline{u}$ can be computed offline. Once $\underline{u}(P, \underline{\mathcal{N}}(P))$ has been computed for every partition element P, the only remaining online computation is the determination of $\pi(x_k)$ for each state x_k on the feedback trajectory. In our case, this can be done efficiently, since we store the partition in a binary tree. Note, however, that the fast online evaluation of the feedback law is enabled by a comparatively large offline computation, namely the construction of the hypergraph.

Behaviour of the Closed Loop System

Theorem 5 [8]. *Under the assumptions of Theorem 4, if $(x_k)_k$ denotes the trajectory of the closed loop system (19) with feedback (34) and if $V_{\mathcal{P}}(x_0) < \infty$, then there exists $k^* \in \mathbb{N}$ such that $x_{k^*} \in T$ and*

$$V_{\mathcal{P}}(x_k) \geq g(x_k, u_{\mathcal{P}}(x_k)) + V_{\mathcal{P}}(x_{k+1}), \quad k = 0, \ldots, k^* - 1.$$

Proof. From the construction of $u_{\mathcal{P}}$ we immediately obtain the inequality

$$V_{\mathcal{P}}(x_k) \geq g(x_k, u_{\mathcal{P}}(x_k)) + V_{\mathcal{P}}(x_{k+1}) \tag{35}$$

for all $k \in \mathbb{N}_0$ with $x_k \in X \setminus T_{\mathcal{P}}$. This implies the existence of k^* such that the first two properties hold since $g(x_k, u_{\mathcal{P}}(x_k)) > 0$ for $x_k \notin T_{\mathcal{P}}$, $V_{\mathcal{P}}$ is piecewise constant and equals zero only on $T_{\mathcal{P}}$. □

Theorem 5 implies that the closed-loop solution reaches the target $T_{\mathcal{P}}$ at time step k^* and that the optimal value function decreases monotonically until the target is reached, i.e., it behaves like a Lyapunov function. While it is in principle possible that the closed-loop solution leaves the target after time k^*, this Lyapunov function property implies that after such excursions it will return to $T_{\mathcal{P}}$.

If the system (4) is asymptotically controllable to the origin and V is continuous, then we can use the same arguments as in [9] in order to show that on increasingly finer partitions $\mathcal{P}_\ell$ and for targets $T_{\mathcal{P}_\ell}$ shrinking down to $\{0\}$ we

obtain $V_{\mathcal{P}_\ell} \to V$. This can also be used to conclude that the distance of possible excursions from the target $T_{\mathcal{P}_\ell}$ become smaller and smaller as $\mathcal{P}_\ell$ becomes finer.

We note that the Lyapunov function property of $V_\mathcal{P}$ outside $T_\mathcal{P}$ holds regardless of the size of the partition elements. However, if the partition is too coarse then $V_\mathcal{P} = \infty$ will hold on large parts of X, which makes the Lyapunov function property useless. In case that large partition elements are desired—for instance, because they correspond to a quantization of the state space representing, e.g., the resolution of certain sensors—infinite values can be avoided by choosing the control value not only depending on one partition element but on two (or more) consecutive elements. The price to pay for this modification is that the construction of the hypergraph becomes significantly more expensive, but the benefit is that stabilization with much coarser discretization or quantization is possible. For details we refer to [10,11].

Example 3 ***(The inverted pendulum reconsidered.).*** We reconsider Example 2 and apply the construction from this section. Figure 10, which results from running the code shown in Fig. 11 as well as lines 25ff. from the code in Fig. 6, shows the discrete upper value function on a partition of 2^{16} boxes with target set $T = [-0.1, 0.1]^2$ as well as the trajectory generated by the discrete feedback (33) for the initial value $(3.1, 0.1)$. As expected, the approximate value function is decreasing monotonically along this trajectory. Furthermore, this trajectory is clearly closer to the optimal one because it converges to the origin much faster.

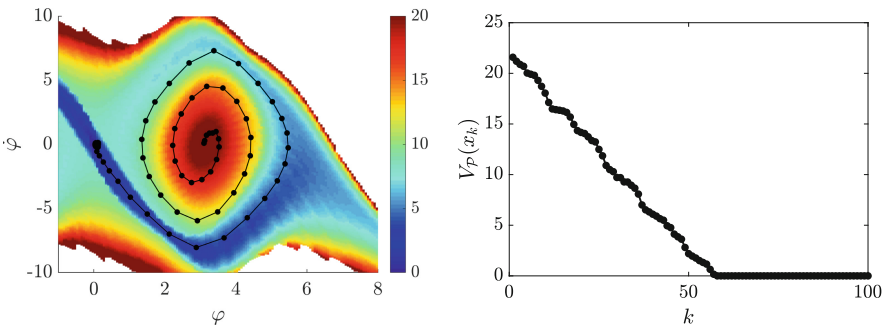

Fig. 10. Inverted pendulum: Discrete upper value function and robust feedback trajectory (left); decrease of the discrete value function along the feedback trajectory.

8 Hybrid, Event and Quantized Systems

Hybrid Systems. The discretization of the optimality principle described in Sects. 4–7 can be used in order to deal with *hybrid systems* in a natural way. Hybrid systems can often be modeled by a discrete time control system of the form

$$\begin{aligned} x_{k+1} &= f_c(x_k, y_k, u_k) \\ y_{k+1} &= f_d(x_k, y_k, u_k) \end{aligned} \quad k = 0, 1, \ldots, \tag{36}$$

```
m = 2; M = 8; m_r = m/(m+M); l = 0.5; g = 9.8;
q1 = 0.1; q2 = 0.05; r0 = 0.01;
v = @(x,u) [ x(:,2), ...                                % vector field & cost
             (g/l*sin(x(:,1)) - 0.5*m_r*x(:,2).^2.*sin(2*x(:,1)) - ...
             m_r/(m*l)*u.*cos(x(:,1)))./(4.0/3.0 - m_r*cos(x(:,1)).^2), ...
             0.5*( q1*(x(:,1).^2) + q2*(x(:,2).^2) + r0*u.^2 )];
h = 0.02; steps = 5;                                    % step size, # of steps
f = @(x,u) rk4u(v,[x zeros(size(x,1),1)],u,h,steps);% control system
n = 3; x1 = linspace(-1,1,n)';
[XX,YY] = meshgrid(x1,x1); X = [ XX(:) YY(:) ];         % sample points
U = linspace(-64,64,17)';                               % control samples

depth = 16; c = [0 0]; r = [8 10];
tree = Tree(c, r);  sd = 8;                             % construct full tree
subdivide(tree, depth),

G = dphgraph2(tree, f, X, U, depth);                    % construct hypergraph
Gt = trnsp_hgraph(G);                                   % transpose hypergraph
T = tree.search_box([0;0], [0.1;0.1], depth);           % target boxes
[V, u] = minmax_dijkstra(G, Gt, T);                     % value function,feedback

V(find(V == Inf)) = NaN;                                % unstabilizable set
figure(1); clf; boxplot2(tree, 'density', V);           % plot value function
colorbar; shading flat; axis('tight')
xlabel('$\varphi$'); ylabel('$\dot\varphi$');
```

Fig. 11. Code: discrete upper value function and robust feedback for the inverted pendulum

with two maps $f_c : X \times Y \times U \to X \subset \mathbb{R}^n$ and $f_d : X \times Y \times U \to Y$. The set U of control inputs can be discrete or continuous, the (compact) set $X \subset \mathbb{R}^n$ is the continuous part of state space and the set Y of discrete states (or *modes*) is a finite set. The class of hybrid systems described by (36) is quite general: It comprises

- models with purely continuous state space (i.e. $Y = \{0\}$, $f_c(x, y, u) = f_c(x, u)$, $f_d \equiv 0$), but discrete or finite control space U;
- models in which the continuous part f_c is controlled by the mode y and only the discrete part f_d of the map is controlled by the input ($f_c(x, y, u) = f_c(x, y)$ and $f_d(x, y, u) = f_d(y, u)$ may be given by an automaton);
- models with state dependent switching: Here we have a general map f_c and $f_d(x, y, u) = f_d(x)$.

As in the previous chapters, we denote the solutions of (36) for initial values $x_0 = x$, $y_0 = y$ and some control sequence $\boldsymbol{u} = (u_0, u_1, \ldots) \in U^{\mathbb{N}}$ by $x_k(x, y, \boldsymbol{u})$ and $y_k(x, y, \boldsymbol{u})$, respectively. We assume that for each k, the map $x_k(\cdot, y, \boldsymbol{u})$ is continuous for each $y \in Y$ and each $\boldsymbol{u} \in U^{\mathbb{N}}$. We prescribe a target set $T \subset X$ (i.e. a subset of the continuous part of state space) and our aim is to find a control sequence $\boldsymbol{u} = (u_k)_{k\in\mathbb{N}}$ such that $x_k(x, y, \boldsymbol{u}) \to T$ as $k \to \infty$ for initial values x, y in some stabilizable set $S \subset X \times Y$, while minimizing the accumulated cost $\sum_{k=0}^{\infty} g(x_k, y_k, u_k)$, where $g : X \times Y \times U \to [0, \infty)$ is a given instantaneous cost with $g(x, y, u) > 0$ for all $x \notin T$, $y \in Y$ and $u \in U$. To this end, we would like to construct an approximately optimal feedback $u : S \to U$ such that a suitable asymptotic stability property for the resulting closed loop system holds. Again,

the construction will be based on a discrete value function. For an appropriate choice of g this function is continuous in x at least in a neighborhood of T [12].

Computational Approach. Let $\mathcal{Q}$ be a partition of the continuous part X of state space. Then the sets

$$\mathcal{P} := \{Q_i \times \{y\} \,|\, Q_i \in \mathcal{Q},\, y \in Y\} \tag{37}$$

form a partition of the product state space $Z = X \times Y$. On $\mathcal{P}$ the approaches from Sects. 4–7 can be applied literally.

Example 4 ***(Example: A switched voltage controller).*** This is an example taken from [15]: Within a device for DC to DC conversion, a semiconductor is switching the polarity of a voltage source V_{in} in order to keep the ouput voltage x_1 as constant as possible close to a prescribed value V_{ref}, cf. Fig. 12, while the load is varying and thus the output current I_{load} changes. The model is

$$\begin{aligned} \dot{x}_1 &= \frac{1}{C}\,(x_2 - I_{\text{load}}) \\ \dot{x}_2 &= -\frac{1}{L}x_1 - \frac{R}{L}x_2 + \frac{1}{L}uV_{\text{in}} \\ \dot{x}_3 &= V_{\text{ref}} - x_1, \end{aligned} \tag{38}$$

where $u \in \{-1, 1\}$ is the control input. The corresponding discrete time system is given by the time-t-map Φ^t ($t = 0.1$ in our case) of (38), with the control input held constant during this sampling period. We use the quadratic instantaneous cost function

$$g(x, u) = q_P(\Phi_1^t(x) - V_{\text{ref}})^2 + q_D(\Phi_2^t(x) - I_{load})^2 + q_I\Phi_3^t(x)^3.$$

The third component in (38) is only used in order to penalize a large L^1-error of the output voltage. We slightly simplify the problem (over its original formulation in [15]) by using $x_3 = 0$ as initial value in each evaluation of the discrete map. Correspondingly, the map reduces to a two-dimensional one on the x_1, x_2-plane.

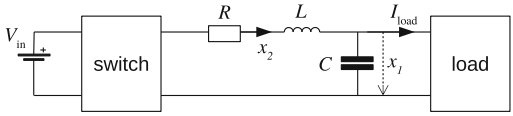

Fig. 12. A switched DC/DC converter (cf. [15]).

In the following numerical experiment we use the same parameter values as given in [15], i.e. $V_{\text{in}} = 1V$, $V_{\text{ref}} = 0.5$, $R = 1\Omega$, $L = 0.1H$, $C = 4F$, $I_{\text{load}} = 0.3\ A$, $q_P = 1$, $q_D = 0.3$ and $q_I = 1$. Confining our domain of interest to the rectangle $X = [0, 1] \times [-1, 1]$, our target set is given by $T = \{V_{\text{ref}}\} \times [-1, 1]$. For the

construction of the finite graph, we employ a partition of X into 64×64 equally sized boxes. We use 4 test points in each box, namely their vertices, in order to construct the edges of the graph.

Using the resulting discrete value function (associated to a nominal $I_{load} = 0.3$ A) and the associated feedback, we repeated the stabilization experiment from [15], where the load current is changed after every 100 iterations. Figure 13 shows the result of this simulation, proving that our controller stabilizes the system as requested.

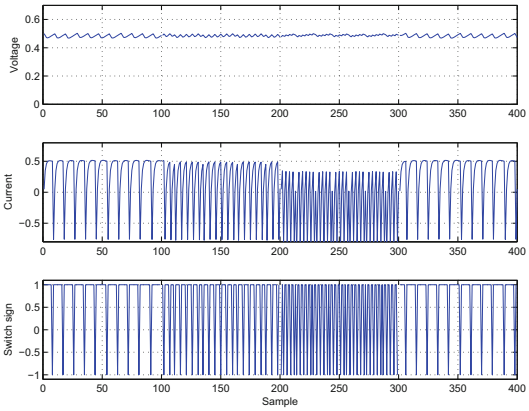

Fig. 13. Simulation of the controlled switched power converter.

Event Systems. In many cases, the discrete-time system (1) is given by time-sampling an underlying continuous time control system (an ordinary differential equation with inputs u and w), i.e. by the time-t-map of the flow of the continuous time system. In some cases, instead of fixing the time step t in each evaluation of f, it might be more appropriate to chosen t in dependence of the dynamics. Formally, based on the discrete time model (1) of the plant, we are dealing with the discrete time system

$$x_{\ell+1} = \tilde{f}(x_\ell, u_\ell), \quad \ell = 0, 1, \ldots, \tag{39}$$

where

$$\tilde{f}(x, u) = f^{r(x,u)}(x, u), \tag{40}$$

$r : X \times U \to \mathbb{N}_0$ is a given *event function* and the iterate f^r is defined by $f^0(x, u) = x$ and $f^r(x, u) = f(f^{r-1}(x, u), u)$, cf. [10]. The associated instantaneous cost $\tilde{g} : X \times U \to [0, \infty)$ is given by

$$\tilde{g}(x, u) = \sum_{k=0}^{r(x,u)-1} g(f^k(x, u), u). \tag{41}$$

The time k of the underlying system (1) can be recovered from the event time ℓ through

$$k_{\ell+1} = k_\ell + r(x_\ell, u_\ell).$$

Note that this model comprises an *event-triggered* scenario where the event function is constructed from a comparison of the state of (1) with the state of (39), as well as the scenario of *self-triggered control* (cf. [1]) where the event function is computed from the state of (1) alone.

Quantized Systems. The approach for discretizing the optimality principle described in Sects. 4–6 is based on a discretization of state space in form of a finite partition. While in general the geometry of the partition elements is arbitrary (except from reasonable regularity assumptions), in many cases (e.g. in our implementation in GAIO) cubical partitions are a convenient choice. In this case, the discretization can be interpreted as a *quantization* of (1), where the quantized system is given by the finite state system

$$P_{k+1} = F(P_k, u_k, \gamma_k), \quad k = 0, 1, \ldots, \tag{42}$$

with

$$F(P, u, \gamma) = \pi(f(\gamma(P), u)), \qquad P \in \mathcal{P}, u \in U,$$

where $\gamma : \mathcal{P} \to X$ is a function which chooses a point x from some partition element $P \in \mathcal{P}$, i.e. it satisfies $\pi(\gamma(P)) = P$ for all $\mathcal{P} \in P$ [10]. The choice function models the fact that it is unknown to the controller from which exact state x_k the system transits to the next cell P_{k+1}. It may be viewed as a perturbation which might prevent us from reaching the target set – in this sense, (42) constitutes a *dynamic game* in the sense of Sect. 6.

9 Lazy Feedbacks

In some applications, e.g. when data needs to be transmitted between the system and the controller over a channel with limited bandwidth, it might be desirable to minimize the amount of transmitted data. More specifically, the question might be how to minimize the number of times that a new control value has to be transmitted from the controller to the system. In this section, we show how this can be achieved in an optimization based feedback construction by defining a suitable instantaneous cost function.

In order to detect a change in the control value we need to be able to compare its current value to the one in the previous time step. Based on the setting from Sect. 2, we consider the discrete-time control system

$$z_{k+1} = \bar{f}(z_k, u_k), \quad k = 0, 1, 2, \ldots \tag{43}$$

with $z_k = (x_k, w_k) \in Z := X \times U$, $u_k \in U$ and

$$\bar{f}(z, u) = \bar{f}((x, w), u) := \begin{bmatrix} f(x, u) \\ u \end{bmatrix}.$$

Given some target set $T \subset X$, we define $\bar{T} := T \times U$ as the target set in the extended state space Z. The instantaneous cost function $\bar{g} : Z \times U \to [0, \infty)$, which penalizes control value changes is given by

$$\bar{g}_\lambda(z, u) = \bar{g}_\lambda((x, w), u) := (1 - \lambda)g(x, u) + \lambda\delta(u - w) \tag{44}$$

with

$$\delta(u) = \begin{cases} 0 & \text{if } u = 0, \\ 1 & \text{else.} \end{cases} \tag{45}$$

Here, $\lambda \in [0, 1)$ (in particular, $\lambda < 1$ in order to guarantee that $\bar{g}(z, u) = 0$ iff $z \in \bar{T}$).

In order to apply the construction from Sect. 7, we choose a finite partition $\mathcal{P}$ of X. Let $\hat{V}_\mathcal{P}$ denote the associated discrete upper value function, $\hat{S} = \{x \in X : \hat{V}_\mathcal{P}(x) < \infty\}$ the stabilizable set, and $\hat{u}_\mathcal{P}$ the associated feedback for the original system (f, g). For simplicity, we assume that U is finite and use $\mathcal{P} \times U$ as the partition of the extended state space Z. We denote the discrete upper value function of $(\bar{f}, \bar{g}_\lambda)$ by $\bar{V}_\lambda : Z \to [0, \infty]$, the stabilizable subset by $\bar{S}_\lambda := \{z \in Z : \bar{V}_\lambda(z) < \infty\}$ and the associated feedback by $\bar{u}_\lambda : \bar{S}_\lambda \to U$.

For some arbitrary feedback $u_\lambda : \bar{S}_\lambda \to U$, consider the closed loop system

$$z_{k+1} = \bar{f}(z_k, u_\lambda(z_k)), \quad k = 0, 1, 2, \ldots. \tag{46}$$

We will show that for any sufficiently large $\lambda < 1$ the closed loop system with $u_\lambda = \bar{u}_\lambda$ is asymptotically stable on $\bar{S}_\lambda$, more precisely that for $z_0 \in \bar{S}_\lambda$ the trajectory of (46) enters $\bar{T}$ in finitely many steps and that the number of control value changes along this trajectory is minimal.

To this end, for some initial state $z_0 \in \bar{S}_\lambda$, let $(z_k)_k \in Z^\mathbb{N}$, $z_k = (x_k, w_k)$, be the trajectory of (46). Let $\kappa(z_0, u_\lambda) = \min\{k \geq 0 : z_k \in \bar{T}\}$ be the minimal number of time steps until the trajectory reaches the target set $\bar{T}$,

$$E(z_0, u_\lambda) = \sum_{k=0}^{\kappa(z_0,u_\lambda)} \delta\big(u_\lambda(z_k) - w_k\big)$$

the number of control value changes along the corresponding trajectory as well as

$$J(z_0, u_\lambda) = \sum_{k=0}^{\kappa(z_0,u_\lambda)} g(x_k, u(z_k)), \quad \text{resp.} \quad \bar{J}(z_0, u_\lambda) = \sum_{k=0}^{\kappa(z_0,u_\lambda)} \bar{g}(z_k, u(z_k))$$

the associated accumulated costs. Note that

$$\bar{J}(z_0, u_\lambda) = (1 - \lambda)J(z_0, u_\lambda) + \lambda E(z_0, u_\lambda).$$

Theorem 6. *For all $\lambda \in [0, 1)$, $\hat{S} \times U \subset \bar{S}_\lambda$. Using the optimal feedback $\bar{u}_\lambda$ in (46) and for $z_0 \in \bar{S}_\lambda$, $z_k \to \bar{T}$ as $k \to \infty$. Further, there exists $\lambda < 1$ such that for any feedback $u_\lambda : \bar{S}_\lambda \to U$ and $z_0 \in \bar{S}_\lambda$ with $\kappa(z_0, u_\lambda) < K$ for some arbitrary $K \in \mathbb{N}$, we have $E(z_0, u_\lambda) \geq E(z_0, \bar{u}_\lambda)$.*

Proof. By construction, the system (43, 44) fulfills the assumptions of Theorem 5, so we have asymptotic stability of the closed loop system (46) with $u_\lambda = \bar{u}_\lambda$ for all $z_0 \in \bar{S}_\lambda$.

In order to show that $\hat{S} \times U \subset \bar{S}_\lambda$ for all $\lambda \in [0,1)$, choose $\lambda \in [0,1)$ and some initial value $z_0 = (x_0, u_0) \in \hat{S} \times U$. Consider the feedback

$$u(z) = u((x,u)) := \hat{u}_{\mathcal{P}}(x)$$

for system (43). This leads to a trajectory $(x_k, u_k)_k$ of the extended system with $(x_k)_k$ being a trajectory of the the closed loop system for f with feedback $\hat{u}_{\mathcal{P}}$. Since $x_0 \in \hat{S}$, $\hat{V}_{\mathcal{P}}(x_0)$ is finite and the accumulated cost $\bar{J}(z_0, u)$ for this trajectory does not exceed $(1-\lambda)\hat{V}_{\mathcal{P}}(x_0) + \lambda\kappa(z_0, u)$ which is finite. According to the optimality of V_λ,

$$V_\lambda(z_0) \leq (1-\lambda)\hat{V}_{\mathcal{P}}(x_0) + \lambda\kappa(z_0, u) < \infty$$

follows, i.e. $z_0 \in \bar{S}_\lambda$.

To show the optimality of $\bar{u}_\lambda$ with respect to the functional E, assume there exists a feedback $u_\lambda : \bar{S}_\lambda \to U$ with $E(z_0, u_\lambda) \leq E(z_0, \bar{u}_\lambda) - 1$ for some $z_0 \in \bar{S}_\lambda$. Since $\bar{u}_\lambda$ is optimal, the following inequality holds:

$$\begin{aligned}(1-\lambda)J(z_0, u_\lambda) + \lambda E(z_0, u_\lambda) &= \bar{J}(z_0, u_\lambda) \\ &\geq \bar{J}(z_0, \bar{u}_\lambda) \\ &= (1-\lambda)J(z_0, \bar{u}_\lambda) + \lambda E(z_0, \bar{u}_\lambda) \\ &\geq (1-\lambda)J(z_0, \bar{u}_\lambda) + \lambda E(z_0, u_\lambda) + \lambda\end{aligned}$$

and thus

$$(1-\lambda)J(z_0, u_\lambda) \geq (1-\lambda)J(z_0, \bar{u}_\lambda) + \lambda. \tag{47}$$

Let $C(u_\lambda) = \max_{z_0}\{J(z_0, u_\lambda) \mid \kappa(z_0, u_\lambda) < K\}$ which is finite. From (47) we get

$$(1-\lambda)C(u_\lambda) \geq (1-\lambda)C(\bar{u}_\lambda) + \lambda. \tag{48}$$

so that $\lambda \to 1$ leads to a contradiction. □

Acknowledgements. OJ thanks Michael Dellnitz for being his mentor, colleague and friend since more than 25 years. OJ and LG gratefully acknowledge the support through the Priority Programme SPP 1305 Control Theory of Digitally Networked Dynamic Systems of the German Research Foundation. OJ additionally acknowledges support through a travel grant by DAAD.

References

1. Anta, A., Tabuada, P.: To sample or not to sample: self-triggered control for nonlinear systems. IEEE Trans. Autom. Control **55**(9), 2030–2042 (2010)
2. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)

3. Dellnitz, M., Froyland, G., Junge, O.: The algorithms behind GAIO-set oriented numerical methods for dynamical systems. In: Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems, pp. 145–174, 805–807. Springer, Berlin (2001)
4. Dellnitz, M., Hohmann, A.: A subdivision algorithm for the computation of unstable manifolds and global attractors. Numerische Mathematik **75**(3), 293–317 (1997)
5. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
6. Fleming, W.H.: The convergence problem for differential games. J. Math. Anal. Appl. **3**, 102–116 (1961)
7. Grüne, L., Junge, O.: A set oriented approach to optimal feedback stabilization. Syst. Control Lett. **54**(2), 169–180 (2005)
8. Grüne, L., Junge, O.: Approximately optimal nonlinear stabilization with preservation of the Lyapunov function property. In: Proceedings of the 46th IEEE Conference on Decision and Control, pp. 702–707 (2007)
9. Grüne, L., Junge, O.: Global optimal control of perturbed systems. J. Optim. Theory Appl. **136**(3), 411–429 (2008)
10. Grüne, L., Müller, F.: An algorithm for event-based optimal feedback control. In: Proceedings of the 48th IEEE Conference on Decision and Control, Shanghai, China, pp. 5311–5316 (2009)
11. Grüne, L., Müller, F.: Global optimal control of quantized systems. In: Proceedings of the 18th International Symposium on Mathematical Theory of Networks and Systems — MTNS2010, Budapest, Hungary, pp. 1231–1237 (2010)
12. Grüne, L., Nešić, D.: Optimization-based stabilization of sampled-data nonlinear systems via their approximate discrete-time models. SIAM J. Control Optim. **42**(1), 98–122 (2003)
13. Junge, O.: Rigorous discretization of subdivision techniques. In: International Conference on Differential Equations, vol. 1, 2 (Berlin, 1999), pp. 916–918. World Scientific Publishing, River Edge (2000)
14. Junge, O., Osinga, H.M.: A set oriented approach to global optimal control. ESAIM Control Optim. Calc. Var. **10**(2), 259–270 (2004)
15. Lincoln, B., Rantzer, A.: Relaxing dynamic programming. IEEE Trans. Autom. Control **51**(8), 1249–1260 (2006)
16. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. U.S.A. **93**(4), 1591–1595 (1996)
17. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations. Proc. Natl. Acad. Sci. U.S.A. **98**(20), 11069–11074 (2001)
18. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Trans. Autom. Control **40**(9), 1528–1538 (1995)
19. Tucker, W.: Validated Numerics: A Short Introduction to Rigorous Computations. Princeton University Press, Princeton (2011)
20. von Lossow, M.: A min-man version of Dijkstra's algorithm with application to perturbed optimal control problems. In: Proceedings of the GAMM Annual Meeting, Zürich, Switzerland (2007)