Undiscounted control policy generation for continuous-valued optimal control by approximate dynamic programming

Downloaded from: https://research.chalmers.se, 2026-11-14 02:39 UTC

Citation for the original published paper (version of record):
Lock, J., McKelvey, T. (2022). Undiscounted control policy generation for continuous-valued optimal control by approximate
dynamic programming. International Journal of Control, 95(10): 2854-2864.
http://dx.doi.org/10.1080/00207179.2021.1939892

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

International Journal of Control

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tcon20

Undiscounted control policy generation for continuous-valued optimal control by approximate dynamic programming

Jonathan Lock & Tomas McKelvey

To cite this article: Jonathan Lock & Tomas McKelvey (2021): Undiscounted control policy generation for continuous-valued optimal control by approximate dynamic programming, International Journal of Control, DOI: 10.1080/00207179.2021.1939892

To link to this article: https://doi.org/10.1080/00207179.2021.1939892

© 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

Published online: 17 Jun 2021.

Submit your article to this journal

Article views: 7

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tcon20

INTERNATIONAL JOURNAL OF CONTROL
https://doi.org/10.1080/00207179.2021.1939892

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

Undiscounted control policy generation for continuous-valued optimal control by approximate dynamic programming

Jonathan Lock and Tomas McKelvey

Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden

ABSTRACT

We present a numerical method for generating the state-feedback control policy associated with general undiscounted, constant-setpoint, infinite-horizon, nonlinear optimal control problems with continuous state variables. The method is based on approximate dynamic programming, and is closely related to approximate policy iteration. Existing methods typically terminate based on the convergence of the control policy and either require a discounted problem formulation or demand the cost function to lie in a specific subclass of functions. The presented method extends on existing termination criteria by requiring both the control policy and the resulting system state to converge, allowing for use with undiscounted cost functions that are bounded and continuous. This paper defines the numerical method, derives the relevant underlying mathematical properties, and validates the numerical method with representative examples. A MATLAB implementation with the shown examples is freely available.

ARTICLE HISTORY
Received 1 July 2020
Accepted 29 May 2021

KEYWORDS
Approximate dynamic programming; control policy; undiscounted infinite-horizon; optimal control

1. Introduction

Practical methods for generating the optimal control policy (i.e. the state feedback function) for general non-linear optimal control problems are useful tools for control engineers. If the optimal control policy is known, a real-time optimal controller can be implemented on very computationally limited hardware as the optimal control signal can be generated simply by interpolating the pre-computed optimal control based on the current system state. However, one practical difficultly lies in pre-computing the optimal control policy, which can be very computationally expensive. Although several methods for solving this class of problem are well-studied, dynamic programming (DP) variants being one example, they all have associated limitations or drawbacks. Policy iteration is one extensively studied variant of DP (e.g. Bertsekas 2017, p. 246; Puterman 1994, p. 295; Puterman and Brumelle 1979) that has been used for over 40 years for finding the optimal control policy for discrete-valued, non-linear, infinite-horizon problems, i.e. where the state and control variables are taken from discrete sets.

Approximate dynamic programming (ADP) is another well-known extension of DP (see for instance Powell (2009) for a general introduction) that approximates the cost function using a prescribed set of basis functions. One group of ADP methods approximate the cost function by interpolating costs and optimal controls between discrete gridded points (e.g. Munos & Moore, 2002; Santos & Vigo-Aguiar, 1998). This approach allows for extending DP to applications with continuous state variables.

Assuming the problem of finding the approximately-optimal control policy for continuous-valued, non-linear, infinite-horizon problems, one might attempt to use traditional policy iteration in concert with ADP. However, this is problematic as traditional policy iteration requires the set of states and controls to be discrete (i.e. finite) to terminate, while the interpolation performed with ADP leads to a continuous (i.e. infinite) number of possible states and controls. This has led to the development of several methods that can be broadly classified as approximate policy iteration (API) methods, where the termination criterion of conventional policy iteration is altered in order to terminate in finite time and generate an approximately optimal solution.

There are several excellent papers that consider different variants of API. However, the vast majority of these are limited to the case where the cost function is discounted, i.e. where future costs are successively weighted less and less (Bertsekas, 2011; Santos & Rust, 2004; Scherrer, 2014; Stachurski, 2008). Though a discounted cost function may be relevant for some problems and allows for more easily determining a termination criterion, a sizeable portion of optimal control problems are better formulated as undiscounted problems (e.g. minimum fuel/energy/time problems, or yield maximisation for chemical plants and cultivation). Guo et al. (2017) introduce one API method for the undiscounted case from a reinforcement learning perspective, but this method is limited both in that the cost function must be a sum of a positive definite function of the state and a quadratically weighted function of the controls, and that the state and control cannot be arbitrarily constrained.

CONTACT Tomas McKelvey tomas.mckelvey@chalmers.se Department of Electrical Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden

© 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this paper we will introduce a method similar to API schemes that approximates the solution to the infinite-horizon problem by instead solving a finite-horizon problem. More specifically, the method uses conventional interpolating ADP to approximate the undiscounted, infinite-horizon, non-linear, optimal control problem where the state is constrained to converge to a unique equilibrium. The primary contribution of this paper is a termination criterion that terminates at a suitable horizon without requiring the presence of a discount factor, while also allowing for (nearly) arbitrary cost, constraints, and problem dynamics – a combination that is novel to the best of the authors knowledge. The method's sole tuning parameter allows for controlling the trade off between memory consumption, computational time, and accuracy. This allows for the method to be used without in-depth knowledge of the method. Furthermore, as the method's output is the optimal control policy (i.e. the optimal control tabulated by the system state) subsequent on-line control can be implemented using a computationally fast interpolation operation.

The structure of this paper follows; in Section 2 we will define the problem studied in this paper and the structure of the interpolating ADP method we subsequently base our presented method on. We will assume a working knowledge of ADP methods for optimal control. Sundstrom and Guzzella (2009) gives a straightforward introduction while Bertsekas (2017); Puterman (1994) go into more detail. This is followed by Section 3, where we derive relevant properties of the studied problem. Though these properties are mostly already known, by deriving them we can both highlight some important details, as well as use a language and notation more commonly seen by control engineers as compared to existing API literature. In Section 4 we present our method of generating an approximation of the optimal control policy, as well as highlight how existing API methods compare with our method. Finally, in Section 5 we use two representative examples to show the results generated by our method. For ease of reference, a list of the symbols and notation used in this paper is shown in Table 1.

Table 1. List of used notation, symbols, and first definition.

DP		Dynamic programming
ADP		Approximate dynamic programming
API		Approximate policy iteration
d_x	(8a)	Distance between neighbouring points in $\mathcal{X}$
d_u	(8b)	Distance between neighbouring points in $\mathcal{U}$
f_c	(3a)	Cost function
$f_{c,R}$	(17)	Relaxed cost function
f_d	(1)	System dynamics function
f_α	(3f)	Average constraint function
$\mathcal{F}$	(5)	Set of initial conditions with feasible initial condition
$\mathcal{F}'_k$	(29)	Set of feasible gridded initial conditions after k samples
g	(3e)	Inequality constraint function
J	(3a)	Cost
J^*	(3b)	Optimal cost
J^*_{eq}	(11a)	Optimal equilibrium cost
J_R	(18a)	Relaxed cost
J^*_R	(18b)	Optimal relaxed cost
J^{*N}_R	(25a)	Optimal relaxed N-horizon cost
N_M	(32a)	Finite minimum horizon
N'_M	(33b)	Finite UCPADP horizon
$\mathcal{S}$	(3e)	Set of trajectories with feasible dynamics and inequality constraints
u_k	(1)	Control signal at sample k
$\bar{u}$	(2b)	Control trajectory
$\bar{u}^*$	(3c)	Optimal control trajectory
$u_{\text{eq}}, u^*_{\text{eq}}$	(7)	Optimal equilibrium control (identical by Theorem 3.1)
$\bar{u}_R$	(18a)	Relaxed control trajectory
$\bar{u}^*_R$	(18c)	Optimal relaxed control trajectory
$\bar{u}^{*N}_R$	(25b)	Optimal relaxed N-horizon control trajectory
$\mathcal{U}$	(9b)	Cartesian grid of sampled controls for ADP routine
$\mathcal{V}_\alpha$	(3f)	Set of trajectories satisfying average equality constraint
x_k	(1)	System state at sample k
$\bar{x}$	(2a)	State trajectory
$\bar{x}^*$	(3c)	Optimal state trajectory
$x_{\text{eq}}, x^*_{\text{eq}}$	(7)	Optimal equilibrium state (identical by Theorem 3.1)
$\bar{x}_R$	(18a)	Relaxed state trajectory
$\bar{x}^*_R$	(18c)	Optimal relaxed state trajectory
$\bar{x}^{*N}_R$	(25b)	Optimal relaxed N-horizon state trajectory
$x_{k,CL}$	(27)	Closed-loop state after applying a control policy k times
$\mathcal{X}$	(9a)	Cartesian grid of sampled states for ADP routine
α	(3f)	Average constraint
Δ^k_μ	(30)	Control policy deviation at sample k
Δ^k_x	(31)	State deviation at sample k
ε_x	(32c)	State tolerance
ε_μ	(32b)	Control policy tolerance
λ	(17)	Relaxation parameter
μ^*	(6)	Optimal stationary control law
$\bar{\mu}^{*N}_R$	(26)	Optimal relaxed N-horizon control policies

2. Problem formulation

Assume a dynamic system $f_d : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ whose associated state evolution is recursively given by

$$x_{k+1} = f_d(x_k, u_k) \tag{1}$$

for the system state $x_k \in \mathbb{R}^n$ and control input $u_k \in \mathbb{R}^m$ at samples $k \in [0, 1, 2, \ldots]$. Define the infinite sequences

$$\bar{x} \triangleq [x_0, x_1, x_2, \ldots] \tag{2a}$$

$$\bar{u} \triangleq [u_0, u_1, u_2, \ldots] \tag{2b}$$

as the *state trajectory* and *control trajectory*, respectively. Similarly, define the finite sequences $\bar{x}^N \triangleq [x_0, x_1, \ldots, x_{N-1}]$ and $\bar{u}^N \triangleq [u_0, u_1, \ldots, u_{N-1}]$. In particular, for both $\bar{x}$ and $\bar{x}^N$ we, respectively, define x_0 as the *initial condition*.

2.1 The infinite-horizon problem

Given x_0, introduce

$$J(\bar{x}, \bar{u}) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} f_c(x_k, u_k) \tag{3a}$$

$$J^* = \min_{\bar{x},\bar{u}} J(\bar{x}, \bar{u}) \tag{3b}$$

$$(\bar{x}^*, \bar{u}^*) = \operatorname*{argmin}_{\bar{x},\bar{u}} J(\bar{x}, \bar{u}) \tag{3c}$$

subject to

$$(\bar{x}, \bar{u}) \in \mathcal{S} \cap \mathcal{V}_\alpha \tag{3d}$$

for

$$\mathcal{S} = \left\{ (\bar{x}, \bar{u}) : \lim_{N\to\infty} \begin{array}{c} g(x_k, u_k) \le 0 \\ x_{k+1} = f_d(x_k, u_k) \end{array}, \ \forall\, k \in [0, N-1] \right\} \tag{3e}$$

$$\mathcal{V}_\alpha = \left\{ (\bar{x}, \bar{u}) : \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} f_a(x_k, u_k) = \alpha \right\} \tag{3f}$$

as the problem we study in this paper. Here, we denote $f_c : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ the *cost function*, $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^l$ the *inequality constraint(s)*, a scalar parameter $\alpha \in \mathbb{R}$ the *average constraint*, and $f_a : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ the *average constraint function*. We define a *feasible trajectory* as any trajectory $(\bar{x}, \bar{u})$ that satisfies (3d). The set $\mathcal{S}$ gives a convenient notation for demanding that the 'textbook' problem dynamics and inequality constraints hold, while the set $\mathcal{V}_\alpha$ denotes an additional average constraint.

Crucially, as none of the functions in (3) are explicitly dependent on k, its solution satisfies the *principle of optimality* (Bertsekas 2017, p. 20; Bellman 1954). Bertsekas (2017, p. 15) shows that this in turn implies that the optimal control trajectory $\bar{u}^*$ can equivalently be formulated as the control policy (i.e. state-feedback)

$$\bar{u}^* = \left[\mu_0^*(x_0), \mu_1^*(x_1), \dots \right], \tag{4}$$

where $\mu_k^* : \mathbb{R}^n \to \mathbb{R}^m$ are functions that are *independent of the initial condition* x_0. Note that while $\bar{x}$ and $\bar{u}$ (with various sub- and super-scripts) are sequences of vectors of scalars, $\bar{\mu}$ (with various sub- and super-scripts) are instead *sequences of functions*. We will refer to $\bar{\mu}^*$ as the *optimal control policy*.

Definition 2.1: Define $\mathcal{F} \subseteq \mathbb{R}^n$ as the set of initial conditions with feasible solutions, i.e.

$$\mathcal{F} \triangleq \{ x_0 : \exists\, (\bar{x}, \bar{u}) \in \mathcal{S} \cap \mathcal{V}_\alpha \}. \tag{5}$$

Assumption 2.1: *For the remainder this paper we assume:*

(A.1) *f_c, f_d, g, and f_a are continuous and bounded.*

(A.2) *The optimal solution $(\bar{x}^*, \bar{u}^*)$ associated with x_0 is unique.*

(A.3) *The optimal control policy associated with* (3) *exists, and can be expressed as*

$$\bar{u}^* = \left[\mu^*(x_0), \mu^*(x_1), \dots \right], \tag{6}$$

i.e. it is not only independent of the initial condition x_0, but also independent of the sample index k. We will refer to this as a stationary control policy (Bertsekas & Shreve, 1979).

(A.4) *$\mathcal{F}$ is nonempty, $\lim_{k\to\infty}(x_k^*)$ exists and is independent of x_0 for all $x_0 \in \mathcal{F}$, and x_k^* is asymptotically stable in the sense of Lyapunov for x_0 near $\lim_{k\to\infty}(x_k^*)$.*

Note that (A.1) implies that $J(\bar{x}, \bar{u})$ is finite for any feasible trajectory, and by (A.4) we can furthermore view J^* as the *average (mean) cost*.

Definition 2.2: Assuming (A.4) holds, define

$$(x_{\text{eq}}, u_{\text{eq}}) \triangleq \lim_{k\to\infty} (x_k^*, u_k^*) \tag{7}$$

as the problem's equilibrium point.

Note that (A.2), (A.3), and (A.4) may be difficult to determine a priori for a given problem. We will briefly discuss the possible effects of them not holding in Section 4.

2.2 Interpolating ADP

The method we introduce in this paper uses a conventional interpolating ADP scheme, and we will here use the standard method of gridding x and u into finite Cartesian sets. We define

$$d_x \in \mathbb{R}^n \tag{8a}$$

$$d_u \in \mathbb{R}^m \tag{8b}$$

as the distance between neighbouring grid points for each dimension of the states and controls, respectively. We also define

$$\mathcal{X} \subset \mathbb{R}^n \tag{9a}$$

$$\mathcal{U} \subset \mathbb{R}^m \tag{9b}$$

as the discrete set of state and control grid points resolved by ADP, respectively, separated by d_x and d_u, respectively, and bounded by the region(s) where $g(x, u) \leq 0$. We then use conventional multilinear interpolation to approximate the cost J and optimal control policy μ for the real-valued states that do not lie in the discrete set $\mathcal{X}$. For example, assuming $x \in \mathbb{R}^2$, $u \in \mathbb{R}^1$, and $g(x, u) = |x|_1 \leq 1 \wedge |u| \leq 1$, choosing the very coarse (but illustrative) $d_x = [2, 2]^T$ and $d_u = 0.5$ gives the sets

$$\mathcal{X} = \left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \tag{10a}$$

$$\mathcal{U} = \{-1, -0.5, 0, 0.5, 1\}. \tag{10b}$$

3. Infinite-horizon, average-constrained problem properties

In this section we introduce properties of the undiscounted, infinite-horizon, average-constrained problem that will later be utilised by the method we introduce in Section 4.

3.1 Solution convergence

Definition 3.1: For $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, using the same functions as in (3a), define

$$J_{\text{eq}}^* = \min_{x,u} f_c(x, u) \tag{11a}$$

$$\left(x_{\text{eq}}^*, u_{\text{eq}}^*\right) = \operatorname*{argmin}_{x,u} f_c(x, u) \tag{11b}$$

subject to

$$x = f_d(x, u) \tag{11c}$$

$$g(x, u) \leq 0 \tag{11d}$$

$$f_a(x, u) = \alpha \tag{11e}$$

$$\forall\, x_0 \in \mathcal{F}, \exists\, (\bar{x}, \bar{u}) \text{ s.t. } \lim_{k\to\infty} (x_k, u_k) = (x, u) \tag{11f}$$

as the *optimal reachable equilibrium operating point* $(x_{\text{eq}}^*, u_{\text{eq}}^*)$. (Note that we have identical states on both the left- and right-hand side of (11c), i.e. an equilibrium state.) We can view this as the unique stationary point of the system with lowest cost that we can reach for any initial condition in the feasible set $\mathcal{F}$.

Theorem 3.1: *Given* (A.1) *and* (A.4),

$$J^* = J^*_{\text{eq}} \tag{12}$$

$$\left(x_{\text{eq}}, u_{\text{eq}}\right) = \left(x^*_{\text{eq}}, u^*_{\text{eq}}\right), \tag{13}$$

i.e. the equilibrium we reach will be optimal in the sense of (11).

Proof: For $0 \leq i < j$, define

$$J_{i\to j}\left(\bar{x}, \bar{u}\right) \triangleq \sum_{k=i}^{j} f_c\left(x_k, u_k\right). \tag{14}$$

We can then formulate (3b) as

$$J^* = \min_{\bar{x},\bar{u}} \lim_{N\to\infty} \frac{1}{N} J_{0\to i-1}\left(\bar{x}, \bar{u}\right) + \frac{1}{N} J_{i\to N-1}\left(\bar{x}, \bar{u}\right). \tag{15}$$

As $N \to \infty$, we are guaranteed that $\frac{1}{N} J_{0\to i-1} = 0$ for any fixed $i > 0$ per our assumption that f_c is bounded. This implies that J^* is only dependent on $J_{i\to N-1}$. By (A.4), we can make (x^*_i, u^*_i) arbitrarily close to $(x_{\text{eq}}, u_{\text{eq}})$ for sufficiently large i.

Suppose that

$$\left(x_{\text{eq}}, u_{\text{eq}}\right) \neq \left(x^*_{\text{eq}}, u^*_{\text{eq}}\right). \tag{16}$$

By (A.4) $(x_{\text{eq}}, u_{\text{eq}})$ is unique, implying that $J^* > J^*_{\text{eq}}$. However, by (11f) there exists trajectories $\bar{x}'$ and $\bar{u}'$ such that $\lim_{k\to\infty}(x'_k, u'_k) = (x^*_{\text{eq}}, u^*_{\text{eq}})$, with corresponding cost $J' < J^*$, contradicting (16).

For an alternate view of the same proof, see Bertsekas (2012, p. 298). ■

By Theorem 3.1 we can intuitively view the infinite-horizon problem's solution as ignoring any (finite) costs during the transient phase and driving the state to the reachable stationary point with lowest cost. This is a special case of the *turnpike property* (Trélat & Zuazua, 2015; Zaslavski, 2014), which states that the solution to problems with a sufficiently long (finite) horizon tends to display transient dynamic initial and terminal phases, with a middle stationary phase that is independent of the initial and terminal conditions. Of course, the infinite-horizon problem does not have a terminal phase, and we can thus view the solution to our problem (3a) as consisting of an initial transient followed by stationary operation at the optimal reachable equilibrium point.

From a notation perspective, by Theorem 3.1 we do not need to make the distinction between x_{eq} and x^*_{eq}. For consistency, we will use x^*_{eq} from here on out.

3.2 Average-constraint relaxation

Definition 3.2: For a fixed, bounded, scalar relaxation parameter $\lambda \in \mathbb{R}$, define the *relaxed cost* as

$$f_{c,R}\left(x, u\right) \triangleq f_c\left(x, u\right) + \lambda f_a\left(x, u\right). \tag{17}$$

Now we can introduce the *relaxed problem* as

$$J_R\left(\bar{x}_R, \bar{u}_R\right) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} f_{c,R}\left(x_k, u_k\right) \tag{18a}$$

$$J^*_R = \min_{\bar{x}_R,\bar{u}_R} J_R\left(\bar{x}_R, \bar{u}_R\right) \tag{18b}$$

$$\left(\bar{x}^*_R, \bar{u}^*_R\right) = \operatorname*{argmin}_{\bar{x}_R,\bar{u}_R} J_R\left(\bar{x}_R, \bar{u}_R\right) \tag{18c}$$

subject to

$$\left(\bar{x}_R, \bar{u}_R\right) \in \mathcal{S}, \tag{18d}$$

where we view $J_R(\bar{x}_R, \bar{u}_R)$ as the *relaxed representation* of $J(\bar{x}, \bar{u})$, and J^*_R and $(\bar{x}^*_R, \bar{u}^*_R)$ as the *optimal relaxed cost* and *optimal relaxed trajectories*, respectively. Note that $(\bar{x}_R, \bar{u}_R)$, and therefore also $(\bar{x}^*_R, \bar{u}^*_R)$, are not formally constrained to lie in $\mathcal{V}_\alpha$.

For clarity, we will use the notation $\bar{x}_R$ and $\bar{u}_R$ when referring to trajectories associated with the relaxed problem. We will for ease of notation assume that $(\bar{x}^*_R, \bar{u}^*_R)$ is unique (much as (3c)), though we can in principle use DP (and in turn the method to be presented) to solve problems with non-unique solutions.

Lemma 3.1: *For a given* α*, assume for some* λ *we have* $(\bar{x}^*_R, \bar{u}^*_R) \in \mathcal{V}_\alpha$*. Then* $(\bar{x}^*_R, \bar{u}^*_R) = (\bar{x}^*, \bar{u}^*)$.

Proof: For convenience, introduce $\zeta^* \triangleq (\bar{x}^*, \bar{u}^*)$, $\zeta^*_R \triangleq (\bar{x}^*_R, \bar{u}^*_R)$, $\zeta \triangleq (\bar{x}, \bar{u})$, $\zeta_R \triangleq (\bar{x}_R, \bar{u}_R)$, and

$$h\left(\zeta\right) \triangleq \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} f_a\left(x_k, u_k\right) - \alpha. \tag{19}$$

Note that $h(\zeta) = 0 \Leftrightarrow \zeta \in \mathcal{V}_\alpha$.

The weak duality theorem (Andréasson et al., 2016) ensures that

$$J\left(\zeta^*\right) \geq J\left(\zeta^*_R\right) + \lambda h\left(\zeta^*_R\right) = J_R\left(\zeta^*_R\right) - \lambda\alpha. \tag{20}$$

In (20), by our assumption $\zeta^*_R \in \mathcal{V}_\alpha$ we are ensured that $h(\zeta^*_R) = 0$, giving

$$J\left(\zeta^*\right) \geq J\left(\zeta^*_R\right) = J_R\left(\zeta^*_R\right) - \lambda\alpha. \tag{21}$$

As $\zeta^*_R \in \mathcal{S} \cap \mathcal{V}_\alpha$, ζ^*_R also minimises (3a), allowing us to replace the inequality in (21) with strict equality. By (A.2) ζ^* and ζ^*_R are unique, ensuring that that $\zeta^* = \zeta^*_R$. Finally, as ζ^*_R is independent of constant terms we have that

$$\zeta^* = \zeta^*_R = \operatorname*{argmin}_{\zeta} J_R\left(\zeta\right). \tag{22}$$

■

Theorem 3.2: *Given* (3c) *and its relaxed counterpart* (18c),

(R.1) *If* (18c) *is infeasible (i.e. a solution does not exist), then* (3c) *is also infeasible (i.e.* (A.4) *is violated).*

(R.2) *For a given* λ *and feasible* (18c)*, there exists an* α *where*

$$\left(\bar{x}^*_R, \bar{u}^*_R\right) = \left(\bar{x}^*, \bar{u}^*\right). \tag{23}$$

Proof: (R.1): Trivial, as $\mathcal{S} \supseteq \mathcal{S} \cap \mathcal{V}_\alpha$. ■

Proof: (R.2): As λ is given and (18c) is feasible, we can thus find $(\bar{x}_R^*, \bar{u}_R^*)$. Let us now define

$$\alpha' \triangleq \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} f_a\left(\bar{x}_{R,k}^* \bar{u}_{R,k}^*\right). \tag{24}$$

For $\alpha = \alpha'$ we (by construction) have $(\bar{x}_R^*, \bar{u}_R^*) \in \mathcal{S} \cap \mathcal{V}_\alpha$, trivially satisfying the requirements of Lemma 3.1. ■

In essence, for a given λ (R.2) ensures us that $(\bar{x}_R^*, \bar{u}_R^*) = (\bar{x}^*, \bar{u}^*)$ for some value of α. We can intuitively view λ as a tuning parameter, where different values of λ are associated with different solutions, each of which (trivially) have an associated average that we can compute by means of (24).

Using the relaxed problem formulation allows us to avoid the explicit average constraint (3f), which is primarily of use in the sense that the problem becomes more numerically tractable. At its core, the method we will introduce in this paper approximates the solution to (3a) by instead solving a finite-horizon problem of sufficient length. One naive method of satisfying the average constraint would then be to introduce an additional state variable that stores the accumulated average, i.e. $z_N = \sum_{k=0}^{N-1} f_\alpha(x_k, u_k)$. We could then add an equality constraint demanding $z_n/N = \alpha$. However, this is computationally demanding (as we need to introduce an additional state variable, which DP schemes scale poorly with) and introduces a bias in the achieved average (as the average z_n/N is taken over both the initial transient and the stationary phase, we therefore only achieve the desired average as $N \to \infty$). Using the relaxed formulation thus avoids these issues entirely.

3.3 Convergence of finite-horizon problem

We will in this section introduce notation for the finite-horizon problem, which will then be used for constructing the method presented in this paper.

Definition 3.3: For a given finite horizon N, bounded λ, and initial condition x_0, define

$$J_R^{*N} = \min_{\bar{x}_R^N, \bar{u}_R^N} \frac{1}{N} \sum_{k=0}^{N-1} f_{c,R}\left(x_k, u_k\right) \tag{25a}$$

$$\left(\bar{x}_R^{*N}, \bar{u}_R^{*N}\right) = \operatorname*{argmin}_{\bar{x}_R^N, \bar{u}_R^N} \frac{1}{N} \sum_{k=0}^{N-1} f_{c,R}\left(x_k, u_k\right) \tag{25b}$$

subject to

$$\left(\bar{x}_R^N, \bar{u}_R^N\right) \in \mathcal{S} \tag{25c}$$

as the *N-horizon relaxed problem* with average cost J_R^{*N} and associated (finite-length) state and control trajectories $(\bar{x}_R^{*N}, \bar{u}_R^{*N})$. Furthermore, define

$$\bar{\mu}_R^{*N} = \left[\mu_{R,0}^{*N}, \mu_{R,1}^{*N}, \ldots, \mu_{R,N-1}^{*N}\right], \tag{26}$$

where $\mu_{R,k}^{*N} : \mathbb{R}^n \to \mathbb{R}^m$ is the k'th state-feedback control policy, as the N-horizon sequence of control policies associated with (25a).

Definition 3.4: Define

$$x_{k,CL}\left(\mu, x_0\right) \tag{27}$$

as the (not necessarily optimal) k'th closed-loop state given by repeatedly applying a (sample-independent) control policy μ k times from an initial state x_0, e.g.

$$x_{0,CL}\left(\mu, x_0\right) \triangleq x_0$$
$$x_{1,CL}\left(\mu, x_0\right) \triangleq f_d\left(x_0, \mu\left(x_0\right)\right)$$
$$x_{2,CL}\left(\mu, x_0\right) \triangleq f_d\left(x_{1,CL}\left(\mu, x_0\right), \mu\left(x_{1,CL}\left(\mu, x_0\right)\right)\right).$$

Note that the method of generating $x_{k,CL}$ is very similar to the forward-calculation stage of ADP, and differs only in that the control policy is kept constant.

Definition 3.5: For a given control policy μ, define

$$\mathcal{F}_k'(\mu) \triangleq \left\{x_0 \in \mathcal{X} : g(x_{k',CL}, \mu(x_{k',CL})) \leq 0 \; \forall \, k' \in [0, k]\right\}. \tag{29}$$

We can thus view $\mathcal{F}_k'(\mu)$ as the set of initial conditions in $\mathcal{X}$ that satisfies the problem constraints and dynamics (the latter trivially, as we use μ to apply a control and give the next state) after applying the control policy μ k times.

Definition 3.6: For $k > 0$, introduce the maximum control policy deviation $\Delta_\mu^k \in \mathbb{R}^m$ as

$$\left[\Delta_\mu^k\right]_i \triangleq \max_{\substack{x \in \mathcal{F}'_{\lceil k/2 \rceil}(\mu_{R,0}^{*k}) \\ k' \in [0, \lceil k/2 \rceil]}} \left|\left[\mu_{R,0}^{*k}(x) - \mu_{R,k'}^{*k}(x)\right]_i\right|, \tag{30}$$

where the notation $[a]_i$ refers to the i'th element of a vector a and $\lceil \ldots \rceil$ refers to the ceiling function. We can view Δ_μ^k as indicating the convergence of $\mu_{R,0}^{*k}$ to μ^*, evaluated at the gridded state points $\mathcal{X}$ whose associated state evolution remains feasible after $k/2$ iterations.

Definition 3.7: Introduce the maximum state deviation $\Delta_x^k \in \mathbb{R}^n$ as

$$\left[\Delta_x^k\right]_i \triangleq \max_{x \in \mathcal{F}'_{k\lceil k/2 \rceil}(\mu_{R,0}^{*k})} \left| \left[x_{\lceil k/2 \rceil, CL}\left(\mu_{R,0}^{*k}, x\right) - \sum_{x' \in \mathcal{F}'_{\lceil k/2 \rceil}(\mu_{R,0}^{*k})} x_{\lceil k/2 \rceil, CL}\left(\mu_{R,0}^{*k}, x'\right) \frac{1}{|\mathcal{F}'_{\lceil k/2 \rceil}|} \right]_i \right|. \tag{31}$$

Note that the notationally heavy second line of (31) is equivalent to the mean feasible state after $\lceil k/2 \rceil$ iterations. Similarly to Definition 3.6, we can thus view Δ_x^k as indicating the convergence of $[x_{0,CL}, x_{1,CL}, \ldots, x_{\lceil k/2 \rceil, CL}]$ to $\bar{x}^*$, evaluated at the points where $x_{\lceil k/2 \rceil, CL}$ remains feasible.

Trivially, using Definitions 3.6 and 3.7 gives:

Proposition 3.1: *By* (A.3) $\lim_{k\to\infty} \Delta_\mu^k = 0$, *and by* (A.4) $\lim_{k\to\infty} \Delta_x^k = 0$.

Definition 3.8: Given a *control policy tolerance* $\varepsilon_\mu \in \mathbb{R}^m$ and *state convergence tolerance* $\varepsilon_x \in \mathbb{R}^n$, define

$$N_M \triangleq \min_k k \tag{32a}$$

such that

$$\left[\Delta_\mu^k\right]_i < \left[\varepsilon_\mu\right]_i \forall i \in [1, m] \tag{32b}$$

$$\left[\Delta_x^k\right]_i < [\varepsilon_x]_i \forall i \in [1, n], \tag{32c}$$

as the *minimum horizon*. Proposition 3.1 ensures us that that for any ε_μ and ε_x there exists an associated finite horizon N_M, which we view as the shortest finite-horizon approximation of the infinite-horizon problem.

4. The UCPADP method

In this section we introduce the primary contribution of this paper: *Undiscounted Control Policy generation by Approximate Dynamic Programming* (UCPADP), a method that generates an approximation of μ^*. At its core, in UCPADP we generate an approximation of the optimal control policy by iteratively testing successively larger horizons until the termination criteria (32a) are satisfied. For computational efficiency reasons we will return to, UCPADP will approximate the control policy as

$$\mu^* \approx \mu_{R,0}^{*N'_M} \tag{33a}$$

$$\text{where } N_M \leq N'_M \leq 2N_M, \tag{33b}$$

i.e. the generated horizon will lie in a range between N_M and $2N_M$.

We can at this stage highlight one of the more significant differences between UCPADP and conventional API: the choice of termination conditions. Conventional API generates improved control policies analogous to $\mu_{R,1}^{*1}, \mu_{R,2}^{*2}, \mu_{R,3}^{*3}, \ldots$ with an associated cost $J_R^{*1}, J_R^{*2}, J_R^{*3}, \ldots$, and eventually terminates when the difference between either successive policies or cost is below a given threshold, for instance as in Stachurski (2008) and Santos and Rust (2004, CPI, PSDP). This is similar to the test performed in (32b), which requires the control policy to be near-stationary. However, in conventional API the termination tolerance (analogous to ε_μ) is sized based on the discount factor, and depending on the specific method chosen the tolerance is either undefined or tends towards zero when the discount factor tends towards one (i.e. becomes the undiscounted case we study here). Scherrer (2014) and Bertsekas (2011) review other methods that do not terminate based on the change in the control policy, but instead use some other termination criterion. However, these methods also assume a discounted problem formulation. Guo et al. (2017) is one example of a method that considers the undiscounted case, however their method imposes fairly significant limits on the class of cost and constraint functions (as discussed previously).

The state convergence condition (32c) is to the best of our knowledge novel, and serves a crucial purpose in that it demands the horizon be long enough for *all gridded feasible initial conditions to converge to a region near the equilibrium*. Recall that by (A.4) x_k^* (the true optimal state trajectory) is stable in the sense of Lyapunov for initial conditions near the equilibrium, and in concert with Theorem 3.1 we are thus ensured that an initial condition near the equilibrium will also remain in its vicinity. As we apply test (32c) to all feasible elements in $\mathcal{X}$, at least one initial condition $x_0 \in \mathcal{X}$ will therefore start and then *remain in the nearby vicinity of the equilibrium*. Ultimately, by combining (32b) and (32c) we are ensured that $\mu_{R,0}^{*N'_M}$ is nearly constant during the interval needed for all feasible gridded points in $\mathcal{X}$ to reach the vicinity of the equilibrium.

In UCPADP, we determine $\mu_{R,0}^{*N'_M}$ numerically efficiently in a manner similar to API implemented with ADP. We do this using a nested scheme that repeatedly switches between backward-calculation phases (successively generating control policies with longer associated horizons) and forward-calculation phases (applying tests (32b) and (32c), and eventually terminating when both tests pass). A description of the phases in UCPADP follows, see Figure 1 for an illustration. For now, assume ε_μ and ε_x are given (fixed) vectors.

First, we arbitrarily choose a small initial horizon N and perform N backward-calculation iterations, giving us (among other data) μ_R^{*N}. We can then perform test (32b) and, by performing $N/2$ forward-calculation steps, test (32c). If both tests pass we terminate and return $\mu_{R,0}^{*N}$ as our approximation of μ^*. Conversely, if either of these tests fail by (A.4) we are ensured that increasing the horizon sufficiently will give a control policy that satisfies the tests. In UCPADP we chose to proceed by increasing the horizon to $3N$. Fortunately, in our DP scheme we can compute μ_R^{*3N} using only $2N$ additional backward-calculation iterations by resuming the backward-calculation from μ_R^{*N}. This is possible as each successive backward-calculation step is independent of the total horizon. After generating μ_R^{*3N} we can now again test (32b) and (32c). Should both tests pass we can return $\mu_{R,0}^{*3N}$ as our approximation of μ^*, and otherwise recursively repeat this procedure of doubling the number of backcalculation steps until the tests pass (i.e. generating and testing horizons $N, 3N, 9N, 27N, \ldots$). A pseudocode implementation of the UCPADP method is listed in Algorithm 1.

Up to this point we have assumed that the problem solution is unique (A.2), converges to a stationary control policy (A.3), and all states converge to a unique equilibrium (A.4). Let us now briefly consider the case where we do not know if these assumptions hold beforehand. Beginning with (A.2), recall that we can determine whether or not this assumption holds during the backward-calculation phase by checking if the minimum cost is unique, and in the case of a non-unique cost we can resolve this by simply returning one arbitrarily selected optimal

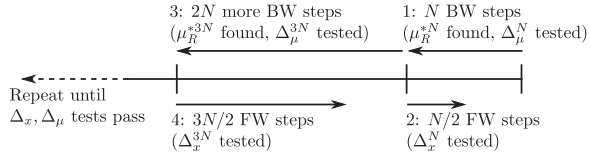

Figure 1. UCPADP steps, successively switching between generating more accurate control policies (backward calculation, steps 1, 3, . . .), and evaluating whether the control policy is constant over the time needed for the state evolution to converge (forward calculation, steps 2, 4, . . .).

solution. Let us now focus on the case where (A.3) and (A.4) are unverified. Applying the UCPADP method gives one of two possible outcomes: UCPADP either never terminates (i.e. (32b) and (32c) never pass), or it terminates after a finite number of back-calculation iterations. If UCPADP never terminates, then one possible cause is that (A.3) and/or (A.4) do not hold (i.e. the termination criteria (32b) and (32c) correctly detected a non-stationary control policy and/or detected that the system states do not converge to a single equilibrium). Alternatively, it is possible that the problem's discretisation and/or tolerances were poorly chosen. Regardless, should UCPADP never terminate it is clear that no valid solution could be generated. If UCPADP *does* terminate, we are assured that either: *(i)* (A.3) and (A.4) do hold and a near-optimal control policy is generated, or *(ii)* the problem is maliciously nonlinear and (A.3) and/or (A.4) do not hold (which went undetected by (32b) and (32c)), ultimately giving a control policy without any clear optimality guarantees. As the class of problems we can attempt to solve with UCPADP covers general non-linear systems it is not surprising that there exist pathological problems that lead UCPADP (and ADP in general) to generate erroneous solutions. Ultimately it is up to the user of UCPADP to determine whether or not the studied problem is of a class that satisfies the (arguably mild) assumptions (A.3) and (A.4).

In Algorithm 1, we extend the notion of termination used thus far by adding a parameter $N_{\max}$ that allows for configuring a maximum horizon that terminates UCPADP if $N > N_{\max}$. This acts as a safety and guarantees that UCPADP terminates after a finite number of iterations. In the event that this limit triggers UCPADP to terminate we can conclude that either the minimum horizon is larger than $N_{\max}$, that (A.3) and/or (A.4) do not hold, or the discretisation and/or tolerances were poorly chosen. Of course, should this happen then we can not say anything about the stability (let alone the optimality) of the returned control policy.

Tests (32b) and (32c) are straightforward to compute exhaustively, as the initial conditions x_0 come from the discrete set $\mathcal{X}$. Furthermore, in UCPADP we have chosen to double the number of additional back-calculation steps to perform between each test evaluation. This attempts to balance the time spent on backward-calculation iterations and the horizon length sufficiency tests, though we may ultimately solve for problem horizons up to $2N_M$, as indicated by (33b). Ultimately this choice is arbitrary, and it is possible for some problems to use another scheme for selecting a new length.

From a practical perspective, we have found that setting $\varepsilon_\mu \approx 2d_u$ and $\varepsilon_x \approx 2d_x$ (the distance between points in $\mathcal{U}$ and $\mathcal{X}$, respectively) is a good design choice for well-behaved problems. Smaller values raise the risk of never terminating, e.g. due to residual state trajectory jitter caused by approximation inherent to interpolation, while larger values give an unnecessarily large approximation of the true control policy $\mu^*_{R,0}$. Ultimately, this implies that UCPADP has to some degree only one tuning parameter: the ADP discretisation, which trades off accuracy with computational time and memory demands.

As UCPADP is based on interpolating ADP (and in turn DP) it is subject to the inherent limitations of DP methods, in particular its poor scaling with problem dimensionality (colloquially referred to as the 'curse of dimensionality') (Bellman, 1954; Bertsekas, 2017). This limits UCPADP to low- to moderate-dimensional problems. The examples shown in the following section (with two state variables and one control variable, giving a total of three independent variables) are easily solved using an ordinary desktop computer on the order of one minute to one hour (depending on the demanded solution accuracy). In practice, we expect UCPADP to be viable for up to 4–6 continuous-variable problems, depending on the discretisation of the state and control variables, the nature of the problem, and the available computational power.

A general implementation of the UCPADP method in the MATLAB language, including the numerical examples in the following section, is available at `https://gitlab.com/lerneaen_hydra/ucpadp`.

Algorithm 1 Pseudocode UCPADP algorithm. Here, $\mathrm{DP}_{1-\mathrm{back}}$ and $\mathrm{DP}_{1-\mathrm{fw}}$ are the one-step backward and forward ADP operations. $\mathcal{X}$ is the set of initial conditions tested in the ADP method. N_{init} is the initial problem horizon. C_N is the cost-to-go after N iterations. Note here that a reverse notation is used for the calculated control policy; μ_1 corresponds to the state-feedback control policy from the first back-calculation step (i.e. μ^{*N}_{N-1}) while μ_N corresponds to the last (i.e. μ^{*N}_0). We can view the index k as counting the number of back-calculation steps performed. Note the abuse of notation on line 14 that indicates the Δ^N_x and Δ^N_μ tests, respectively.

1: **function** UCPADP($\mathcal{X}$, $N_{\max}$, N_{init})
2: $N_b \leftarrow N_{\mathrm{init}}$ ▷ Batch back-calculation steps
3: $N \leftarrow 0$ ▷ Cumulative back-calculation steps
4: $C_0 \leftarrow 0$ ▷ Set initial cumulative cost to zero
5: **repeat**
6: **for** $N \leftarrow N, N + N_b$ **do**
7: $\mu_{N+1}, C_{N+1} \leftarrow \mathrm{DP}_{1-\mathrm{back}}(C_N)$
8: **end for**
9: $X_{CL} \leftarrow \mathcal{X}$
10: **for** $i \leftarrow 1, \lceil N/2 \rceil$ **do**
11: $X_{CL} \leftarrow \mathrm{DP}_{1-\mathrm{fw}}(X_{CL}, \mu_N)$
12: **end for**
13: $N_{\mathrm{b}} \leftarrow 2 \cdot N_{\mathrm{b}}$ ▷ Raise N_b by doubling
14: **until** $N > N_{\max}$ or $(|X_{CL} - \mathrm{mean}(X_{CL})| < \varepsilon_x$ and $|\mu_N - \mu_k| < \varepsilon_u \; \forall k \in [\lceil N/2 \rceil, N])$
15: **return** μ_N, X_{CL}, N
16: **end function**

5. Representative examples

We illustrate the UCPADP method, introduced in Section 4, by solving two simple problems. Though 'toy' problems in some sense, recall that Assumption 2.1 allows for significantly more difficult (and practically relevant) problems. First we consider the classical minimum-time inverted pendulum problem, where we highlight the stopping criterion of UCPADP. Afterwards, we consider the problem of maintaining an average pendulum angle with minimum control power, illustrating the average-constraint properties shown in Section 3.2.

We will consider the dynamical system given by a simple pendulum (Figure 2) for both problems. For a pendulum with length l, point mass m, gravitational force g, damping coefficient d, angle θ, and applied torque u, the dynamic equation for the system can be derived as

$$\ddot{\theta} + \frac{d}{m}\dot{\theta} + \frac{g}{l}\sin(\theta) = \frac{1}{ml^2}u. \tag{34}$$

In both the following examples we will assume a discrete-time control system with sample rate t_s, i.e. the control input u is piecewise constant over intervals of uniform time t_s. If the problem is reformulated as a set of coupled first-order ordinary differential equations with a state variable vector

$$x \triangleq \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{35}$$

then we can express the state at the next sample as

$$x_{k+1} = f_p(x_k, u_k), \tag{36}$$

where f_p is given by solving (34) over a time t_s with initial condition x_k and constant control input u_k.

5.1 The inverted pendulum

To illustrate the mechanics of UCPADP's termination criterion, consider the traditional minimum-time inverted pendulum problem (formulated here as an infinite-horizon problem)

$$J^* = \min_{\bar{x},\bar{u}} \lim_{N\to\infty} \frac{1}{N}\sum_{k=0}^{N-1} f_c(x_k) \tag{37a}$$

$$f_c(x) = \begin{cases} 0 & \text{if } |\theta_k - \pi| < 2d_x, |\dot{\theta}_k| < 2d_x \\ 1 & \text{else} \end{cases} \tag{37b}$$

subject to

$$x_{k+1} = f_p(x_k, u_k) \tag{37c}$$

$$|u_k| \le 1 \tag{37d}$$

$$-2 \le \theta_k \le 3.5 \tag{37e}$$

$$-1.5 \le \dot{\theta}_k \le 2. \tag{37f}$$

All the following results are shown for a sample time of $t_s = 0.2$, pendulum parameters set to give the system dynamics equation $\ddot{\theta} + \sin(\theta) = u$, state variables discretised by a Cartesian grid with separation $d_x = [0.05, 0.05]^T$ in the range allowed by (37e) and (37f), and the control variable discretised with even spacing $d_u = 0.01$ in the range allowed by (37d). Setting ε_x and ε_μ to the suggested value of twice the discretisation gives $\varepsilon_x = [0.1, 0.1]^T$ and $\varepsilon_\mu = 0.02$.

Note that the cost function (37b) equally penalises all pendulum configurations other than the single vertical zero-velocity state combination, and with an infinite horizon (and small enough d_x) gives a solution arbitrarily close to the traditional minimum-time formulation. The state bounds (37e) and (37f) have been chosen to give a reasonable range for the specific initial value we will study shortly.

For the above problem, UCPADP terminates after testing a horizon of $N'_M = 135$, indicating that $45 < N_M \le 135$. An illustration of termination criterion (32b) is shown in Figure 3, where we can verify the condition is satisfied as all values are above $\lceil N/2 \rceil = 68$. Furthermore, Δ_μ^k will by construction take values from $\mathcal{U} = \{0, \pm 0.01, \pm 0.02, \ldots, \pm 1\}$. For $\varepsilon_\mu = 0.02$ (32b) will thus only be satisfied for values $-0.01, 0, 0.01$. We can see this in Figure 3, where $\Delta_\mu^{135} = 0$. Similarly, criterion (32c) is illustrated in Figure 4, where we can verify that representative trajectories all converge to a region bounded by ε_x (shown by the yellow box). An illustration of the control policy ultimately generated by UCPADP is shown in Figure 5. Solving this specific problem took approximately 10 minutes using a standard desktop PC.

Figure 6 shows a comparison of the solution generated by UCPADP and a reference solution, generated by formulating a problem with an explicit horizon of $N = 10N'_M = 1350$ (i.e. one order of magnitude longer the UCPADP horizon), for $x_0 = [0, 0]^T$. Here the reference solution is generated using a traditional ADP scheme, configured with the same sample time and state/control grid discretisation. Note that we intentionally compare the UCPADP solution to a traditional ADP solution (in contrast to, for instance, an analytical solution) as we wish to highlight the accuracy of the automatically sized horizon, rather than the accuracy of an interpolating ADP scheme.

The average cost over the time interval shown in Figure 6 is 0.09664 for the UCPADP solution, while the cost associated with the reference solution is 0.09689, i.e. a deviation[1] of

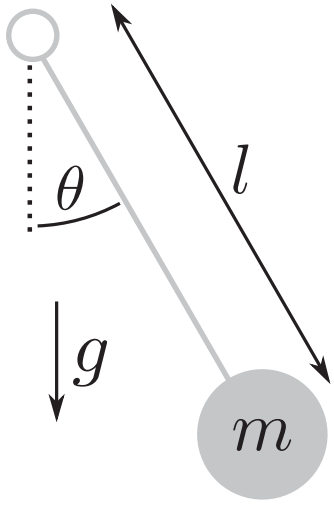

Figure 2. A simple pendulum.

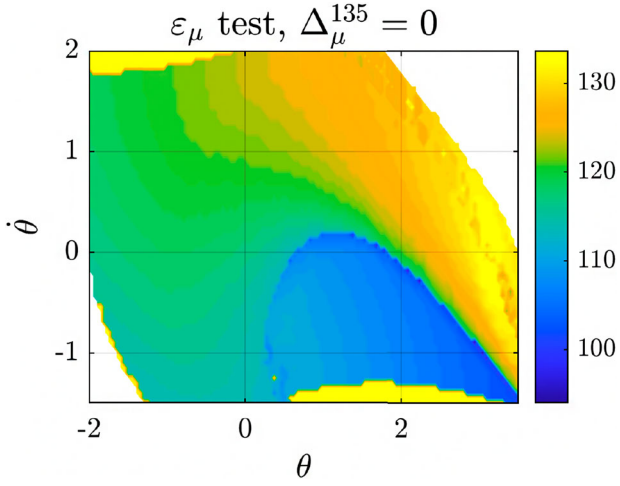

Figure 3. Visualisation of $\varepsilon_\mu = 0.02$ test for $N'_M = 135$. The shaded regions indicate the number of samples that the control policy varies less than ε_μ, while white regions indicate a feasible solution could not be found, i.e. white regions lie outside of $\mathcal{F}'_{135}$. Note that as $\mathcal{U}$ is discrete then Δ_μ^{135} is also, i.e. we here have Δ_μ^{135} identically equal to zero.

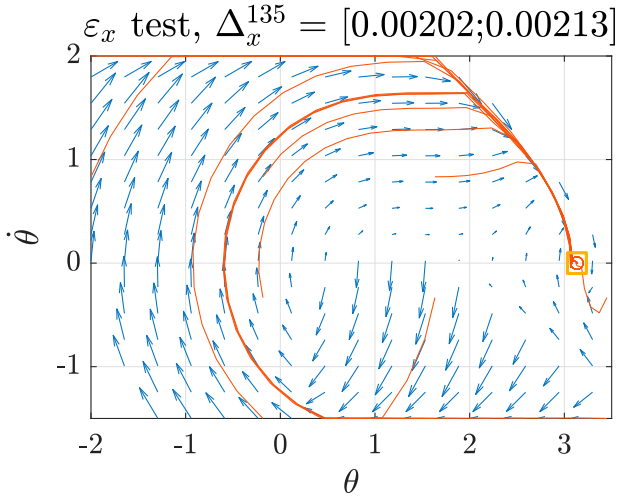

Figure 4. Visualisation of $\varepsilon_x = [0.1, 0.1]^T$ test for $N'_M = 135$. Arrows indicate the motion of the system through its phase space. Representative closed-loop trajectories are shown with solid lines. The closed-loop state at $N = 135$ is shown by (overlapping) small red circles near $\theta = \pi, \dot{\theta} = 0$. The box near $\theta = \pi, \dot{\theta} = 0$ indicates the Δ_x^{135} termination criterion, which is satisfied as all states at $N = 135$ lie inside the box.

0.25%. We can conclude (for this specific problem) that the cost associated with the UCPADP solution is virtually identical to a conventional ADP solution, indicating that the identified horizon $N'_M = 135$ was sufficient.

5.2 The constant-angle pendulum

Let us now consider a problem that illustrates the properties of the average constraint introduced in Section 3.2. Assume we wish to solve

$$J^* = \min_{\bar{x},\bar{u}} \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} u_k^2 \tag{38a}$$

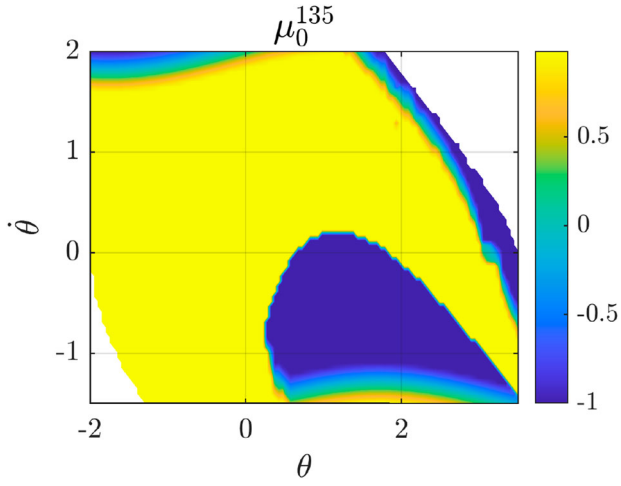

Figure 5. Control policy associated with (37a) for horizon $N = 135$. The shaded region shows the optimal control to apply for any given state, while white regions indicate infeasible states, i.e. outside of $\mathcal{F}'_{135}$.

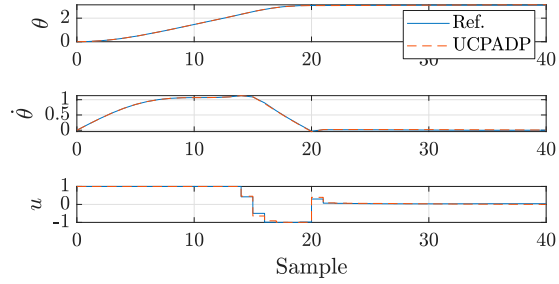

Figure 6. Comparison of the solutions given by UCPADP and an open-loop ADP reference method for $x_0 = [0, 0]^T$.

subject to

$$x_{k+1} = f_p(x_k, u_k) \tag{38b}$$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \theta_k = \theta_{\text{ref}} \tag{38c}$$

$$|u_k| \leq 1, |\theta_k| \leq 1, |\dot{\theta}_k| \leq 1, \tag{38d}$$

i.e. the problem of keeping the average pendulum angle at a setpoint θ_{ref} while minimising the quadratic control input u_k^2.

By Theorem 3.2 we can avoid including the average constraint (38c) by augmenting the cost functional (38a) as

$$J_R^* = \min_{\bar{x}_R,\bar{u}_R} \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} u_k^2 + \lambda\theta_k \tag{39}$$

for a constant scalar λ. Assuming the problem reaches an equilibrium with control u_{eq} and states $\theta = \theta_{\text{ref}}, \dot{\theta} = 0$, by (34) we have $u_{\text{eq}} = mgl\sin(\theta_{\text{ref}})$. We can thus express the equilibrium cost as

$$c_{\text{eq}} = (mgl\sin(\theta_{\text{ref}}))^2 + \lambda\theta_{\text{ref}} \tag{40}$$

which is a function of one variable. Equation (40) has one unique stationary point (a minimum) in the permissible range $|\theta| < 1$, and we can thus find the specific value λ that gives

the lowest equilibrium cost at the desired setpoint by setting $\frac{dc_{\text{eq}}}{d\theta_{\text{ref}}} = 0$ and solving for λ, giving

$$\lambda_0 = -2m^2g^2l^2 \sin(\theta_{\text{ref}}) \cos(\theta_{\text{ref}}) . \tag{41}$$

We can now reformulate (38a) as the equivalent problem

$$J_R^* = \min_{\bar{x}_R, \bar{u}_R} \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} u_k^2 - \theta_k\lambda_0 \tag{42a}$$

subject to

$$x_{k+1} = f_p(x_k, u_k), \quad |u_k| \leq 1, \ |\theta_k| \leq 1, \ |\dot{\theta}_k| \leq 1. \tag{42b}$$

As in the previous example, we discretise the state and control variables evenly in the permissible space, here with separation $d_x = [0.02, 0.02]^T$ and $d_u = 0.01$, respectively. Solving (42a) for pendulum parameters resulting in a system dynamics equation $\ddot{\theta} + \dot{\theta} + \sin(\theta) = u$ and $\theta_{\text{ref}} = 0.5$ gives the results shown in Figure 7 (again compared with a reference solution given by explicitly choosing a large horizon, one order of magnitude larger than the horizon given by UCPADP).

For this problem, we find that the UCPADP solution generates a solution with control cost (i.e. $\sum u_k^2$) of 0.2321 over the horizon shown in Figure 7, while the control cost associated with the reference solution is 0.2323 (i.e. a deviation of 0.09%), again showing that the accuracy of the UCPADP solution is virtually identical to that of a reference ADP solution.

For comparison, in Figure 8 we also show the solution quality parameterised by different finite horizons. More specifically, we solve the finite-horizon counterpart of (42a), i.e. using the notation introduced in (25a), for varying finite horizons N (denoted the *problem horizon*), resulting in the associated control policies $\mu_{R,0}^{*N}$. We then apply the control policy to the set of initial conditions feasible with a long horizon $N = 1350$ (denoted the *trajectory horizon*). The plot shows the augmented cost of the trajectory horizons, i.e. J_R^*, parameterised by different problem horizons. We can identify that the average cost is higher for short problem horizons than for long problem horizons, and that the cost associated with problem horizons $\gtrsim 40$ is constant, indicating that for this problem a problem horizon $\gtrsim 40$ is sufficient. It may therefore seem like UCPADP is inefficient in its choice of problem horizon (135 samples). However, computing the average cost of any given problem horizon shown in Figure 8 is time consuming, with each individual problem

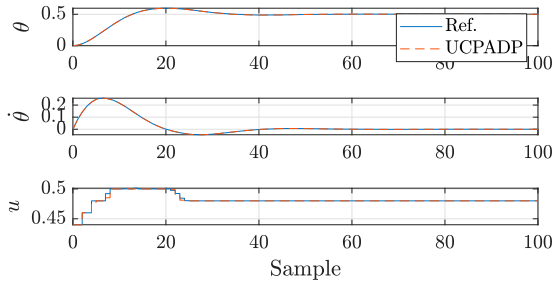

Figure 7. Comparison of the solutions given by UCPADP and an open-loop ADP reference method for $x_0 = [0, 0]^T$. The state trajectories are nearly identical and the control trajectory displays only very small differences.

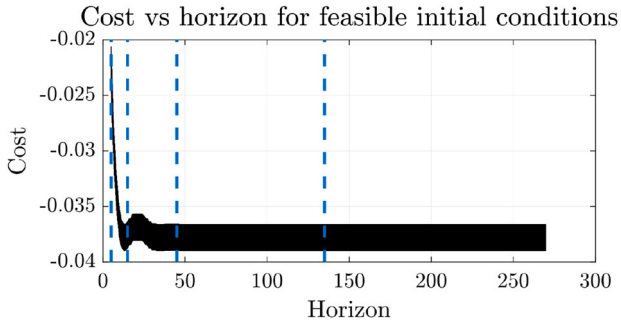

Figure 8. Cost of the applying control policy associated with varying problem horizon lengths. The cost is shown for all feasible initial conditions, resulting in a range of costs for any given horizon (e.g. $[-0.037, -0.039]$ for horizons ≥ 40). Problem horizons tested by UCPADP are shown with dashed lines.

horizon taking approximately the same time to compute as the *entire* UCPADP solution, as well as requiring problem-specific knowledge of the initial conditions and trajectory horizon to average over. The trade-off between spending time computing additional back-calculation steps and checking whether a given horizon is sufficiently large thus motivates a scheme like our proposed horizon-doubling method.

6. Conclusions

In this paper we have introduced UCPADP, a numerical method inspired by API. UCPADP can be used to generate a near-optimal control policy for general undiscounted continuous-valued infinite-horizon nonlinear optimal control problems. The problem can also optionally be constrained to converge to a given equilibrium. The primary contribution of UCPADP is the introduction of a termination criterion that is amenable to the undiscounted case, while still allowing for general costs and constraints. We have evaluated the method by solving two simple, but representative, problems. For both examples we showed that the generated control policy was on par with the accuracy of a reference ADP solution (whose accuracy is determined by the chosen discretisation of the problem).

UCPADP has several properties that render it useful as as one part of the process of constructing an on-line controller. Firstly, it shares a property with other API methods in that it does not require any a-priori information about a suitable horizon, instead performing an indefinite number of iterations and terminating when a suitable problem horizon is found. Secondly, the tuning parameters are simple to grasp, as they trade off solution accuracy with computational time and memory demands. Finally, the output from UCPADP, as with other API methods, is a control policy (i.e. a state feedback table). After this control policy is computed in an off-line phase it can in turn be used to construct a subsequent on-line controller with very low computational demand, only requiring a simple interpolation operation to determine the control signal.

Full source code of the implementation as well as the specific problems studied is available at `https://gitlab.com/lerneaen_hydra/ucpadp`.

Note

1. The fact that the UCPADP solution has a lower associated cost is likely due to the inherent approximation of interpolating ADP.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was performed within the Combustion Engine Research Center at Chalmers (CERC) with financial support from the Swedish Energy Agency.

ORCID

Jonathan Lock http://orcid.org/0000-0003-3677-8132
Tomas McKelvey http://orcid.org/0000-0003-2982-5535

References

Andréasson, N., Evgrafov, A., Patriksson, M., Gustavsson, E., Nedelkova, Z., Sou, K. C., & Önnheim, M. (2016). *An introduction to continuous optimization* (3rd ed.). Studentlitteratur.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, *60*(6), 503–515. https://doi.org/10.1090/S0002-9904-1954-09848-8

Bertsekas, D. P. (2011, August). Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, *9*(3), 310–335. https://doi.org/10.1007/s11768-011-1005-3

Bertsekas, D. P. (2012). *Dynamic programming and optimal control* (4th ed., Vol. 2). Athena Scientific.

Bertsekas, D. P. (2017). *Dynamic programming and optimal control* (4th ed., Vol. 1). Athena Scientific.

Bertsekas, D. P., & Shreve, S. E. (1979). Existence of optimal stationary policies in deterministic optimal control. *Journal of Mathematical Analysis and Applications*, *69*(2), 607–620. https://doi.org/10.1016/0022-247X(79)90171-9

Guo, W., Si, J., Liu, F., & Mei, S. (2017). Policy approximation in policy iteration approximate dynamic programming for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(7), 2794–2807. https://doi.org/10.1109/TNNLS.2017.2702566

Munos, R., & Moore, A. (2002). Variable resolution discretization in optimal control. *Machine Learning*, *49*(2/3), 291–323. https://doi.org/10.1023/A:1017992615625

Powell, W. B. (2009, April). What you should know about approximate dynamic programming: Approximate dynamic programming. *Naval Research Logistics (NRL)*, *56*(3), 239–249. https://doi.org/10.1002/nav.20347

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons.

Puterman, M. L., & Brumelle, S. L. (1979, February). On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, *4*(1), 60–69. https://doi.org/10.1287/moor.4.1.60

Santos, M. S., & Rust, J. (2004, January). Convergence properties of policy iteration. *SIAM Journal on Control and Optimization*, *42*(6), 2094–2115. https://doi.org/10.1137/S0363012902399824

Santos, M. S., & Vigo-Aguiar, J. (1998, March). Analysis of a numerical dynamic programming algorithm applied to economic models. *Econometrica*, *66*(2), 409. https://doi.org/10.2307/2998564

Scherrer, B. (2014, May). Approximate policy iteration schemes: A comparison. arXiv:1405.2878 [cs, stat]. http://arxiv.org/abs/1405.2878.

Stachurski, J. (2008, March). Continuous state dynamic programming via nonexpansive approximation. *Computational Economics*, *31*(2), 141–160. https://doi.org/10.1007/s10614-007-9111-5

Sundstrom, O., & Guzzella, L. (2009, July). *A generic dynamic programming Matlab function*. 2009 IEEE Control Applications, (CCA) Intelligent Control, (ISIC) (pp. 1625–1630). IEEE. https://ieeexplore.ieee.org/document/5281131

Trélat, E., & Zuazua, E. (2015). The turnpike property in finite-dimensional nonlinear optimal control. *Journal of Differential Equations*, *258*(1), 81–114. https://doi.org/10.1016/j.jde.2014.09.005

Zaslavski, A. J. (2014). *Turnpike phenomenon and infinite horizon optimal control*. Springer International Publishing.