Learning Sequential Composition Control

Esmaeil Najafi, *Member, IEEE*, Robert Babuška, *Member, IEEE*,
and Gabriel A. D. Lopes, *Member, IEEE*

***Abstract*—Sequential composition is an effective supervisory control method for addressing control problems in nonlinear dynamical systems. It executes a set of controllers sequentially to achieve a control specification that cannot be realized by a single controller. As these controllers are designed offline, sequential composition cannot address unmodeled situations that might occur during runtime. This paper proposes a learning approach to augment the standard sequential composition framework by using online learning to handle unforeseen situations. New controllers are acquired via learning and added to the existing supervisory control structure. In the proposed setting, learning experiments are restricted to take place within the domain of attraction (DOA) of the existing controllers. This guarantees that the learning process is safe (i.e., the closed loop system is always stable). In addition, the DOA of the new learned controller is approximated after each learning trial. This keeps the learning process short as learning is terminated as soon as the DOA of the learned controller is sufficiently large. The proposed approach has been implemented on two nonlinear systems: 1) a nonlinear mass-damper system and 2) an inverted pendulum. The results show that in both cases a new controller can be rapidly learned and added to the supervisory control structure.**

***Index Terms*—Domain of attraction (DOA), passivity-based control (PBC), reinforcement learning (RL), sequential composition, supervisory control, switching control.**

I. Introduction

SEQUENTIAL composition [1] is a supervisory control methodology that can address complex control specifications in nonlinear dynamical systems. It focuses on the interaction between a collection of predesigned controllers, each endowed with a DOA and a goal set. If the goal set of one controller lies in the DOA of another controller, which is defined as the prepare relation [1], the supervisor can instantly switch from the first controller to the second. This approach decomposes complex tasks into smaller and simpler tasks such that each can be solved using traditional stabilization and tracking control methods. The composition is usually represented by a supervisory finite-state machine that we call the control automaton, with each state (mode) associated with the specific controller, a corresponding DOA and a goal set.

Applications of sequential composition include, for instance, balancing of an underactuated system [2], navigation of an autonomous mobile robot [3], [4], navigation of fully actuated dynamical systems through cluttered environments [5], control of cooperative systems [6], etc. The standard sequential composition framework has been extended in several ways. In [7], robust controller specifications are composed sequentially. Linear quadratic regulators trees (LQR-trees) is a feedback motion planning algorithm, designed based on the sequential composition approach, that uses computed stability regions to construct a tree of LQR-stabilized trajectories through the state space [8].

Although sequential composition provides a well-structured supervisory architecture, it cannot handle situations for which no controller was designed *a priori*. In this paper, we propose a learning sequential composition control approach to handle unmodeled situations by means of online learning. This paper addresses three main questions. First, how to learn a new controller online that can be added to the existing supervisory control architecture. Second, how to guarantee that the learning process is safe. Third, what is a suitable criterion for stopping the learning process.

The proposed learning sequential composition method works as follows. When a desired state is given, the supervisor computes a sequence of controllers over the control automaton. This sequence steers the system from an initial state to the desired state by switching between the local controllers. However, if the supervisor does not succeed in finding a sequence of controllers that drive the system to the desired state with its current set of controllers, a learning mode is activated to learn a new controller. Once the controller is learned, it is added to the control automaton by interconnecting it with the associated controllers such that it respects the prepare relation.

We employ reinforcement learning (RL) in which the control law is computed by interaction with the system, without the need of a model [9]. The learning experiments only explore the regions located within the union of the existing DOAs. This form of learning guarantees that exploration is always safe, because the supervisor can activate a stabilizing controller if the learning process reaches the boundary of the overall DOA. After each learning trial, the DOA of the learned controller is approximated by solving an optimization problem using sum of squares (SOS) programming [10] or by applying a sampling method [11] developed by the authors.

Manuscript received January 26, 2015; revised May 9, 2015; accepted August 21, 2015. The work of E. Najafi was supported by the Ministry of Science, Research and Technology of the Islamic Republic of Iran through his scholarship. This paper was recommended by Associate Editor P. P. Angelov.

E. Najafi is with the Delft Center for Systems and Control, Delft University of Technology, Delft 2628 CD, The Netherlands, and also with the Department of Mechatronics and Systems Engineering, Faculty of New Sciences and Technologies, University of Tehran, Tehran 14399-57131, Iran (e-mail: e.najafi@tudelft.nl).

R. Babuška and G. A. D. Lopes are with the Delft Center for Systems and Control, Delft University of Technology, Delft 2628 CD, The Netherlands (e-mail: r.babuska@tudelft.nl; g.a.delgadolopes@tudelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2481081

While learning is in progress, the DOA of the controller typically enlarges around its goal set. Once the DOA gets large enough to cover other DOAs and relevant goal sets to provide the necessary connections between the controllers, the learning process is terminated and the learned controller is added to the control automaton.

This paper is organized as follows. Section II reviews briefly the key concepts of sequential composition. Section III presents the main contributions of this paper, which is the use of learning in sequential composition. Section IV discusses rapid learning by exploiting the DOAs of the learning controllers and using passivity theory. In Section V, simulation and experimental results are presented for the application of the proposed method on two nonlinear systems. Finally, Section VI provides a brief discussion and then concludes this paper with some research lines for future work.

II. Sequential Composition

Consider the dynamical system

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

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state vector, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input, and $f\colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is the system dynamics. For a particular state-feedback controller $\Phi_i(x)$, indexed by i, the closed-loop system is

$$\dot{x} = f(x, \Phi_i(x)) = f_i(x). \tag{2}$$

Let x_i^* be a stable equilibrium of the closed-loop system (2). The goal set of controller $\Phi_i(x)$, denoted $\mathcal{G}(\Phi_i) \subseteq \mathcal{X}$, is described[1] by

$$\mathcal{G}(\Phi_i) = \{x_i^*\}. \tag{3}$$

Each control law is valid in a subset of the state space, called the DOA and denoted $\mathcal{D}(\Phi_i) \subseteq \mathcal{X}$. If $x(t, x_0)$ denotes the solution of (2) at time t, subject to the initial condition, the DOA of controller Φ_i is defined by the set

$$\mathcal{D}(\Phi_i) = \left\{x_0 \in \mathcal{X}\colon \lim_{t\to\infty} x(t, x_0) = \mathcal{G}(\Phi_i)\right\}. \tag{4}$$

It is assumed that system (1) is controllable throughout the union of all existing DOAs and each controller can stabilize the system at its goal set. Moreover, switching strategies and transitions between controllers are defined based on the prepare relation. According to this relation, controller Φ_i prepares controller Φ_j, denoted $\Phi_i \succeq \Phi_j$, if $\mathcal{G}(\Phi_i) \subset \mathcal{D}(\Phi_j)$ [1]. In other words, once the system enters $\mathcal{D}(\Phi_j)$ while en route to $\mathcal{G}(\Phi_i)$ the supervisor can instantly switch from controller Φ_i to Φ_j. Backchaining away from the controller that stabilizes the system at the desired state to the controller whose DOA contains the initial state results in a converging switching control law that ensures the stability of the closed-loop system through the overall DOA. This is an important property of sequential composition as a switching control methodology [12].

Each mode of the control automaton, indexed by i, describes a tuple $s_i \in \mathcal{S}$ as

$$s_i = \{\Phi_i, \mathcal{D}(\Phi_i), \mathcal{G}(\Phi_i)\} \tag{5}$$

[1]Note that in general the goal sets of controlled systems can be complex. For the purpose of this paper we assume only stabilizing controllers to a point in the state space.

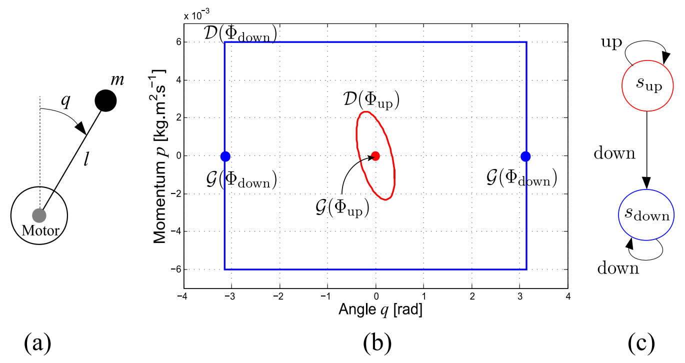

q
m
l
Motor
x 10^-3
$\mathcal{D}(\Phi_{\text{down}})$
$\mathcal{D}(\Phi_{\text{up}})$
$\mathcal{G}(\Phi_{\text{down}})$
$\mathcal{G}(\Phi_{\text{up}})$
$\mathcal{G}(\Phi_{\text{down}})$
Momentum p [kg.m^{2}s^{-1}]
Angle q [rad]
up
s_{up}
down
s_{down}
down
(a)
(b)
(c)

Fig. 1. Sequential composition controller designed for an inverted pendulum. (a) Schematic of the inverted pendulum. (b) Approximated DOAs and goal sets. (c) Induced control automaton.

where $\mathcal{S}$ is a finite set of modes. When a new controller is defined, first its relevant interactions with other controllers are computed based on the prepare relation. Then, its representative mode together with the associated arcs (events) are added to the control automaton.

In standard sequential composition, it is assumed that the set of controllers are composable, the resulting graph is fully reachable [13], and the union of DOAs covers the entire state space, that is

$$\mathcal{D}(\Phi) = \bigcup_{\Phi_i} \mathcal{D}(\Phi_i) = \mathcal{X}. \tag{6}$$

If these assumptions are satisfied, the sequential composition controller can stabilize the system at a given state in the union of DOAs. However, these assumptions are typically not satisfied in practice. Therefore, we propose the use of learning in the structure of sequential composition to be able to handle these situations.

As an example, consider a sequential composition controller designed for an input-saturated inverted pendulum [see Fig. 1(a)]. The state vector is $x = [q\ p]^T$ with q the angle of the pendulum measured from the upright position and $p = J\dot{q}$ the angular momentum. The control system consists of two controllers Φ_{up} and Φ_{down}. Controller Φ_{up} stabilizes the pendulum at the "up" equilibrium point ($q = 0$) and controller Φ_{down} at the "down" equilibrium point ($q = \pi$). Fig. 1(b) illustrates the state space of the pendulum with the approximated DOAs and goal sets. Since the control input is saturated, controller Φ_{up} cannot swing the pendulum up from any initial state. Hence, $\mathcal{D}(\Phi_{\text{up}})$ is represented by a conservative ellipsoid centered at point $(0, 0)$. Controller Φ_{down} is globally stabilizing and its DOA is the entire state space, hence $\mathcal{D}(\Phi_{\text{down}})$ is illustrated by a rectangle covering the whole state space. The goal sets of the up and down controllers are the points $\mathcal{G}(\Phi_{\text{up}}) = \{(0, 0)\}$ and $\mathcal{G}(\Phi_{\text{down}}) = \{(\pi, 0)\}$, respectively. Fig. 1(c) depicts the control automaton, in which every mode s_i is associated with controller Φ_i. There is a prepare relation between controller Φ_{up} and Φ_{down} since $\mathcal{G}(\Phi_{\text{up}}) \subset \mathcal{D}(\Phi_{\text{down}})$, i.e., event down connects mode s_{up} to s_{down}. The supervisor is automatically synthesized based on the prepare relation described between the two controllers.

If the system starts in mode s_{down}, the feasible string of events for the control automaton are $S_{\text{down}} = \{\text{down}^*\}$, where the operator "$*$" denotes the Kleene closure [14], i.e., $S_{\text{down}} = \{\epsilon, \text{down}, \text{down down}, \ldots\}$. However, if the system starts at mode s_{up}, the available strings are $S_{\text{up}} = \{\text{up}^*\text{down}^*\}$. Thus, if the reference event signal is given as $S_{\text{ref}} = \text{down up}$, the supervisory controller will block, because there is no arc connecting mode s_{down} to s_{up}. In such a case, the supervisor needs a new controller to construct the required connections through the control automaton.

III. Learning Sequential Composition

In [15], we have proposed the use of RL in sequential composition for online learning new controllers, called learning sequential composition control. In this method, if the supervisor cannot find a sequence among the predesigned controllers to drive the system to the desired state, a learning mode is activated to learn new controllers using RL methods. The learning objective is defined such that the goal set of the learned controller lies inside one of the back-reachable DOAs of the desired state and its DOA is sufficiently large to cover a reachable goal set of the initial state. Once the controller is learned, it is added to the control system together with its corresponding connections with other controllers.

To formalize the framework for learning sequential composition control, we adapt the standard formulation of a hybrid automaton [16] with the supervisory control structure to construct a learning control automaton. The new element of this supervisory finite-state machine is a learning mode that activates learning on a need basis to generate new controllers online.

Definition 1: The learning control automaton is a tuple $A_\ell = (\mathcal{X}, \mathcal{S}, E, (s_0, x_0), \Phi, F, D, G, g, \Gamma)$, where the following holds.

1) $\mathcal{X} \subseteq \mathbb{R}^n$ is the state space of a continuous-time system.
2) $\mathcal{S} = \{\epsilon, s_\ell, s_0, s_1, \ldots, s_q\}$ is a finite set of discrete states or modes. We include an "empty" mode ϵ and a learning mode s_ℓ. The hybrid state of the system is represented by the pair $(s_i, x) \in \mathcal{S} \times \mathcal{X}$.
3) $E = \{e_\ell, e_1, \ldots, e_p\}$ is a finite set of events, where the learning event e_ℓ triggers the learning mode s_ℓ.
4) (s_0, x_0) is the initial state.
5) $\Phi = \{\Phi_\ell, \Phi_0, \Phi_1, \ldots, \Phi_q\}$ is a set of controllers, where Φ_ℓ is an overall learning controller.
6) $F : \mathcal{S} \times \mathcal{X} \times \Phi \rightarrow \mathbb{R}^n$ is a vector field that constrains the evolution of the continuous-time system to the differential equation $\dot{x} = f(x, \Phi_i(x))$, with mode $s_i \in \mathcal{S}/\{\epsilon\}$.
7) $D : \mathcal{S} \rightarrow 2^{\mathcal{X}}$ assigns to each mode s_i the DOA of its associated controller, hence $D(s_i) = \mathcal{D}(\Phi_i)$, $D(\epsilon) = \emptyset$, and $D(s_\ell) = \mathcal{D}(\Phi)$.
8) $G : \mathcal{S} \rightarrow 2^{\mathcal{X}}$ assigns to each mode s_i the goal set of its associated controller, hence $G(s_i) = \mathcal{G}(\Phi_i)$, $\mathcal{G}(\epsilon) = \emptyset$, and the goal set of the learning mode s_ℓ is defined at each learning instance.
9) $g : \mathcal{S} \times E \rightarrow \mathcal{S}$ is a discrete-event transition.
10) $\Gamma : \mathcal{S} \rightarrow 2^E$ is an active event function.

Note that the DOA of the learning controller Φ_ℓ, associated with mode s_ℓ, is the union of the DOAs of the existing controllers, i.e., $\mathcal{D}(\Phi_\ell) = \mathcal{D}(\Phi)$. Hence, controller Φ_ℓ can be activated from any point in the overall DOA. The goal set $\mathcal{G}(\Phi_\ell)$ is defined at each instance when the learning controller is activated. The learning objective is described such that the DOA of the learned controller covers the goal set of a specific controller. This can generate the required connections through the learning control automaton. To detect when the learning mode s_ℓ needs to be activated, we define a binary function $P : \mathcal{X} \times \mathcal{X} \rightarrow \{\text{true, false}\}$ as follows:

$$P(x_0, x_d) = \begin{cases} \text{true} & \text{if } \exists \text{ a path in the control automaton} \\ & s_i \rightarrow s_{i+1} \rightarrow \cdots \rightarrow s_{i+k} \text{ such that} \\ & x_0 \in D(s_i) \text{ and } G(s_{i+k}) = \{x_d\} \\ \text{false} & \text{otherwise.} \end{cases}$$

A. Properties

In sequential composition, a reference signal is given either as a string (sequence) of events or as a desired continuous-time state. If a desired sequence of events $S_{\text{ref}} = e_1 e_2 \ldots e_r$ is available, the supervisor executes the associated controllers from the set $\{\Phi_1, \Phi_2, \ldots, \Phi_r\}$, sequentially. Each controller Φ_i needs an inherent time to evolve state $x_0^i \in \mathcal{D}(\Phi_i)$ to $x_d^i \in \mathcal{G}(\Phi_i)$ with respect to the differential equation $\dot{x} = f(x, \Phi_i(x))$, where x_0^i and x_d^i are the initial state and the goal set of controller Φ_i, respectively. Note that the goal set of each controller (except for the desired state) should be in the DOA of the next controller to enable the supervisor to switch between the controllers.

If the reference signal is given as a desired continuous-time state $x_d \in \mathcal{X}$, an extra process is required to first find a path through the control automaton. For a given desired state x_d, the supervisor searches for a feasible sequence of controllers that drives the system from its current state to the desired state. The binary function P summarizes the result of exploring through the learning control automaton. The process of making a path toward the desired state relies on the conditions outlined below [15].

1) If $P(x_0, x_d)$ is true, the standard sequential composition can drive the system from the initial state x_0 to the desired state x_d.
2) If $\neg P(x_0, x_d)$, $\exists i : x_0 \in D(s_i)$, and $\exists j : x_d \in G(s_j)$, the modes that cover the initial and desired states in the state space are in disconnected sections of the control automaton. This problem can be addressed by learning new controllers to connect the two sections. The DOA of the learned controller has to overlap with one of the reachable goal sets of the initial condition, and its goal set needs to overlap with one of the back-reachable DOAs of the desired state.
3) If $\neg P(x_0, x_d)$, $\exists i : x_0 \in D(s_i)$, $\nexists j : x_d \in G(s_j)$, but $\exists j : x_d \in D(s_j)$, a new controller is needed such that its goal set be the desired state.
4) If $\neg P(x_0, x_d)$ and $\nexists j : x_d \in D(s_j)$, the desired state x_d is not in the DOA of any controller. This requires a new controller to cover unknown regions of the state space.

5) If $\neg P(x_0, x_d)$ and the initial hybrid state is (ϵ, x_0), the initial state x_0 is not in the DOA of any controller, i.e., $\nexists i : x_0 \in D(s_i)$. This situation corresponds to a lack of an initialization routine of the control system that we do not consider in this paper.

The traditional sequential composition is not designed to cope with situation $\neg P(x_0, x_d)$ (i.e., conditions 2–5 or their combinations). As such, the learning mode is introduced to create the required connections between the controllers. In this paper, we use the actor-critic RL method.

The stability of the learning sequential composition controller can be addressed by three stability subproblems. The first is the stability of the predesigned controllers. Based on the properties of sequential composition, it is assumed that every predesigned controller can stabilize the system at its goal set. The second is the stability of the new learned controllers. Using actor-critic RL generates new controllers that stabilize the system in their computed DOAs. The third is the overall stability of the composed controlled system, which is handled by the prepare relation.

B. Actor-Critic Reinforcement Learning

Reinforcement learning is an optimization method in which an optimal controller is learned by interacting with the system [9]. An RL problem can be defined by a Markov decision process with the tuple $M(\mathcal{X}, \mathcal{U}, \bar{f}, \rho)$, where $\mathcal{X}$ is the state space, $\mathcal{U}$ is the action space, $\bar{f} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the state transition function that returns state[2] x_{k+1} after applying action u_k in state x_k, and $\rho : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function that gives the scalar reward $r_{k+1} \in \mathbb{R}$ to the controller after each transition. The learning objective is to find an optimal policy $\pi : \mathcal{X} \to \mathcal{U}$ to maximize a discounted sum of expected instantaneous rewards, which is stored as the value function

$$V^{\pi}(x_k) = \sum_{j=0}^{\infty} \gamma^j r^{\pi}_{k+j+1} = \sum_{j=0}^{\infty} \gamma^j \rho\big(x_{k+j+1}, \pi(x_{k+j})\big) \qquad (7)$$

with $\gamma \in (0, 1)$ a discount factor.

The actor-critic RL method is convenient for problems with continuous state and action spaces, where both the critic (value function) and the actor (control policy) are approximated via basis function parameterizations [17]. The critic is approximated as $\hat{V}(x, \theta) = \theta^T \Psi_c(x)$ with a parameter vector $\theta \in \mathbb{R}^{n_c}$ and a user-defined basis function vector $\Psi_c(x) \in \mathbb{R}^{n_c}$. Similarly, the actor is approximated as $\hat{\pi}(x, \mu) = \mu^T \Psi_a(x)$, where $\mu \in \mathbb{R}^{n_a}$ is a parameter vector and $\Psi_a(x) \in \mathbb{R}^{n_a}$ is a user-defined basis function vector. We use the temporal-difference (TD) method [9] to update the critic parameters, with the TD defined as

$$\delta_{k+1} = r_{k+1} + \gamma \hat{V}(x_{k+1}, \theta_k) - \hat{V}(x_k, \theta_k). \qquad (8)$$

The critic parameters are updated using the gradient ascent rule

$$\theta_{k+1} = \theta_k + \alpha_c \delta_{k+1} \nabla_{\theta} \hat{V}(x_k, \theta_k) \qquad (9)$$

[2]Note that here we consider a discrete system with $x_k = x(T_s k)$ for a defined sampling time T_s.

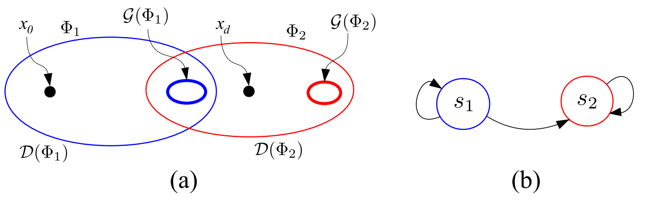


Fig. 2. (a) Pictorial sketch of the DOAs and goal sets of a sequential composition controller, where $x_0 \in \mathcal{D}(\Phi_1)$, $x_d \in \mathcal{D}(\Phi_2)$, and controller Φ_1 prepares controller Φ_2. (b) Induced control automaton, in which the learning mode can make the required connection s_2 to s_1 using bounded learning.

where $\alpha_c > 0$ is a learning rate. To find an optimal policy, the learning algorithm needs to explore new regions in the state-action space. Hence, a zero-mean random exploration term Δu_k is added to the control policy as

$$u_k = \hat{\pi}(x_k, \mu_k) + \Delta u_k. \qquad (10)$$

Finally, the actor parameters are updated by

$$\mu_{k+1} = \mu_k + \alpha_a \delta_{k+1} \Delta u_k \nabla_{\mu_k} \hat{\pi}(x_k, \mu_k) \qquad (11)$$

with $\alpha_a > 0$ the actor learning rate. For more details about actor-critic RL refer to [18].

C. Safe Learning

One of the main concerns of learning is safety. To guarantee the safety of the learning process, the learner is restricted to only explore regions of the state space that lie in the DOAs of all of the existing controllers. This type of exploration is safe since the supervisor can always execute a stabilizing controller once the learner reaches the boundary of the union of the existing DOAs. We call this restricted learning process bounded learning. Consider the condition $\neg P(x_0, x_d)$, with $\exists i : x_0 \in D(s_i)$ and $\exists j : x_d \in D(s_j)$. Two situations are possible for this condition. In the first situation, the DOAs of the existing controllers cover the state space such that the learner does not necessarily need to leave the union of the DOAs to achieve the learning objective. Fig. 2 illustrates a sequential composition controller with two controllers Φ_1 and Φ_2, where $x_0 \in \mathcal{D}(\Phi_1)$ and $x_d \in \mathcal{D}(\Phi_2)$, but $x_d \notin \mathcal{G}(\Phi_2)$. To attain the desired state x_d, the learner starts exploring from the goal set $\mathcal{G}(\Phi_2)$ and searches for the possible trajectories to the desired state. Once a learning experiment reaches the boundary of the union set $\mathcal{D}(\Phi_1) \cup \mathcal{D}(\Phi_2)$, the learning process is reset to $\mathcal{G}(\Phi_2)$. This is an example of bounded learning.

The second situation is when there is no connection between the controllers and the union of the DOAs is not a simply connected set. Here, the learner may need to leave the existing DOAs to achieve the learning goal, as depicted in Fig. 3. This type of learning can be dangerous, because there is no guarantee that the learning experiments can be reset when the learner is exploring new regions of the state space for which no controller was designed *a priori*. We call this unbounded learning in the sense that the learner is not restricted to just explore within the overall DOA. In this paper, we only consider bounded learning.

When the learning mode s_ℓ is activated in a bounded learning process, the learner explores within the existing DOAs.

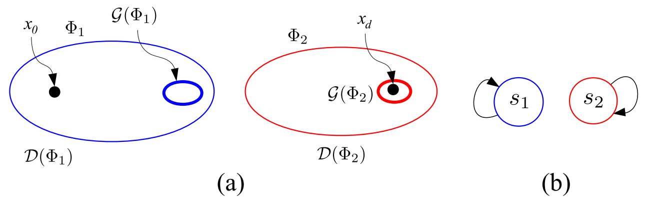


Fig. 3. (a) Pictorial sketch of the DOAs and goal sets of a sequential composition controller, where $x_0 \in \mathcal{D}(\Phi_1)$ and $x_d \in \mathcal{G}(\Phi_2)$, but controller Φ_1 does not prepare controller Φ_2. (b) Induced control automaton, in which the learning mode can only make the required connections using unbounded learning.

Once the learning goal is attained, a new controller Φ_ℓ with the DOA $\mathcal{D}(\Phi_\ell)$ and goal set $\mathcal{G}(\Phi_\ell)$ are stored in the control system. Moreover, the learned mode s_ℓ with its corresponding arcs are added to the learning control automaton based on the prepare relation.

One element that is not addressed in this paper is the automatic choice of the parameters that are required for the learning experiment such as the basis functions for the approximated value function and policy, the reward function, and the learning rates. Currently, much experience goes into designing RL experiments that can run efficiently.

IV. Rapid Learning

Typically, the learning process might be time-consuming and computationally costly. In our approach, we make use of partial prior knowledge to speed up the learning process by combining the learning methods with passivity-based control (PBC) techniques [19]. In addition, the DOA of the new learned controller is approximated after each learning trial. This enables the supervisor to terminate learning as soon as the learned controller's DOA is sufficiently large to satisfy the learning objective [20]. This section provides a brief review on passivity-based learning control and DOA estimation, which are the tools required for our proposed control approach. Next, the proposed control algorithm is described.

A. Passivity-Based Learning Control

PBC has been extensively used for regulation problems in port-Hamiltonian (PH) systems [21]. The PH systems are a general form of Euler–Lagrangian systems, in which the standard input-state-output form of a time-invariant system is given by

$$\begin{cases} \dot{x} = \big(J(x) - R(x)\big)\nabla_x H(x) + g(x)u \\ y = g^T(x)\nabla_x H(x) \end{cases} \tag{12}$$

where $J(x) = -J^T(x) \in \mathbb{R}^{n\times n}$ is a skew-symmetric interconnection matrix, $R(x) = R^T(x) \in \mathbb{R}^{n\times n}$ is a symmetric dissipation matrix, $H(x)$ is the system Hamiltonian, and y is a collocated output with $g(x) \in \mathbb{R}^{n\times m}$. In this paper, we apply the energy-balancing actor-critic (EB-AC) [22] and interconnection and damping assignment actor-critic (A-IDA-AC) [23] techniques to learn new controllers.

1) Energy-Balancing Actor-Critic: In this method, the goal is to find a feedback control law such that the desired closed-loop Hamiltonian $H_d(x)$ has a local minimum at the equilibrium x^*, that is

$$x^* = \arg\min H_d(x). \tag{13}$$

The control law combines an energy-shaping (ES) term with a damping-injection (DI) term

$$\begin{aligned} u(x) &= u_{\text{es}} + u_{\text{di}} \\ &= \big(g^T(x)g(x)\big)^{-1} g^T(x)(J(x) - R(x))\nabla_x H_a(x) \\ &\quad - K(x)g^T(x)\nabla_x H_d(x) \end{aligned} \tag{14}$$

where $K(x) = K^T(x)$ is a symmetric positive semi-definite DI matrix and $H_a(x)$ is an added energy term that satisfies the energy balancing equation

$$H_a(x) = H_d(x) - H(x). \tag{15}$$

The supplied energy function $H_a(x)$ is found by solving a set of partial differential equations, called "matching condition" and given by

$$\begin{bmatrix} g^\perp(x)(J(x) - R(x)) \\ g^T(x) \end{bmatrix} \nabla_x H_a(x) = 0 \tag{16}$$

with $g^\perp(x) \in \mathbb{R}^{(n-m)\times n}$ the left annihilator matrix of the input matrix $g(x)$ [i.e., $g^\perp(x)g(x) = 0$]. A solution of (16) that satisfies the equilibrium condition (13) is selected as $H_a(x)$. This can be solved using the EB-AC method.

The approximated desired Hamiltonian in the EB-AC method is given by

$$\hat{H}_d(x, \xi) = H_{\text{di}} + \xi^T \Psi_{\text{es}}(x) \tag{17}$$

where $\xi \in \mathbb{R}^{n_{\text{es}}}$ is an unknown parameter vector and $\Psi_{\text{es}}(x) \in \mathbb{R}^{n_{\text{es}}}$ is a user-defined basis function vector. The "hat" symbol represents the approximated terms. The DI matrix $K(x)$ is parameterized as

$$\hat{K}(x, \psi) = \psi^T \Psi_{\text{di}}(x) \tag{18}$$

with an unknown parameter vector $\psi \in \mathbb{R}^{n_{\text{di}}}$ and a user-defined basis function vector $\Psi_{\text{di}}(x) \in \mathbb{R}^{n_{\text{di}}}$. Substituting (15), (17), and (18) in (14) results in the control policy

$$\begin{aligned} \hat{\pi}(x, \xi, \psi) &= g^\dagger(x)(J(x) - R(x))\big(\xi^T \nabla_x \Psi_{\text{es}}(x) - \nabla_x H(x)\big) \\ &\quad - \psi^T \Psi_{\text{di}}(x) g^T(x) \nabla_x \hat{H}_d(x) \end{aligned} \tag{19}$$

where $g^\dagger(x) = (g^T(x)g(x))^{-1} g^T(x)$ is the pseudo inverse of matrix $g(x)$. The parameter vectors ξ and ψ are learned using the actor-critic RL method, following the same procedure described in [22]. Consequently, the control input of the EB-AC method is computed at each time step by

$$u_k(x, \xi, \psi) = \text{sat}\big(\hat{\pi}(x_k, \xi_k, \psi_k) + \Delta u_k\big) \tag{20}$$

where Δu_k is a zero-mean Gaussian noise, as an exploration term, and sat is a saturation function.

2) Algebraic Interconnection and Damping Assignment Actor-Critic: Interconnection and damping assignment PBC (IDA-PBC) is an effective nonlinear state-feedback control approach that can be used for stabilizing and tracking problems [24]. The control objective in IDA-PBC is to find a state-feedback law for the nonlinear system (12) such that the resulting closed-loop system can be written as

$$\dot{x} = (J_d(x) - R_d(x))\nabla_x H_d(x) \tag{21}$$

where $J_d(x)$ and $R_d(x)$ are the desired interconnection and dissipation matrices, respectively. The control law that fulfills the desired closed-loop system (21) is of the form

$$u(x) = g^{\dagger}(x)(F_d(x)\nabla_x H_d(x) - f(x)) \tag{22}$$

where $F_d(x) = J_d(x) - R_d(x)$ is the desired system matrix. The unknown desired matrices $F_d(x)$ and $H_d(x)$ are obtained from the matching condition

$$g^{\perp}(x)(F_d(x)\nabla_x H_d(x) - f(x)) = 0. \tag{23}$$

Depending on how this condition is solved, the type of the IDA-PBC controller can vary. We employ algebraic interconnection and damping assignment PBC (A-IDA-PBC), which can be used for tracking and regulation problems in various physical systems such as mechanical, electrical or electromechanical systems [25]. In this method, instead of explicitly solving the matching condition (23) for the unknown elements of the desired system matrix $F_d(x)$, we parameterize $F_d(x)$ as

$$\hat{F}_d(x, \vartheta) = \vartheta^T \Psi_{\text{al}}(x) \tag{24}$$

where $\vartheta \in \mathbb{R}^{n_{\text{al}}}$ is an unknown parameter vector and $\Psi_{\text{al}}(x) \in \mathbb{R}^{n_{\text{al}}}$ is a user-defined basis function vector. Applying the approximated parameterized $\hat{F}_d(x, \vartheta)$ yields the control law

$$\hat{\pi}(x, \vartheta) = g^{\dagger}\big(\vartheta^T \Psi_{\text{al}}(x)\nabla_x H_d(x) - f(x)\big). \tag{25}$$

The parameter vector ϑ is learned using the actor-critic RL method, following the same procedure described in [23]. Consequently, the control input of the A-IDA-AC method is computed at each time step by

$$u_k(x, \vartheta) = \text{sat}\big(\hat{\pi}(x_k, \vartheta_k) + \Delta u_k\big) \tag{26}$$

where Δu_k is again a zero-mean Gaussian noise as an exploration term.

B. Monitoring the Domain of Attraction

While learning is in progress, the DOA of the new learned controller typically enlarges around its goal set, but not necessarily monotonically. If the DOA of the learned controller is monitored, the supervisor can terminate learning as soon as it gets large enough to contain the goal set of other controllers, allowing the creation of a new arc in the learning control automaton. This strategy allows the supervisor to learn a new controller in a short amount of time.

Methods for computing the DOAs for stable equilibrium points of nonlinear systems have been studied for several years [10]. Since the DOAs of nonlinear systems are typically very complex, there are no general analytic methods for their exact computation, but approximations can be found using parametric shapes [26], [27]. An analytical conservative method to approximate the DOA has been proposed via Lyapunov stability theory, where a Lyapunov function that satisfies the locally asymptotic stability of the controller's goal set (equilibrium point) is considered for approximating the DOA [28]. Any sublevel set of this Lyapunov function in which its time derivative is negative is an estimate for the DOA. Since the largest sublevel gives a better estimate, approximating the controller's DOA is transferred to the problem of finding the largest sublevel set of a candidate Lyapunov function [29].

Theorem 1 [30]: A closed set $\mathcal{M} \subset \mathbb{R}^n$, including the origin as an equilibrium point of the autonomous nonlinear system (2), approximates the DOA of the origin if:

1) $\mathcal{M}$ is an invariant set for system (2);
2) a Lyapunov function $L(x)$ can be found such that:
   a) $L(x)$ is positive definite over $\mathcal{M}$;
   b) $\dot{L}(x)$ is negative definite over $\mathcal{M}$.

Note that $\mathcal{M}$ is invariant under the flow (2) if $x(t, x_0) \in \mathcal{M}$ for all $t \geq 0$ and $x_0 \in \mathcal{M}$. In the case of nonzero equilibrium point, without loss of generality, we can replace the variable x by $z = x - \bar{x}^*$, where $\bar{x}^*$ is the nonzero equilibrium point. Hence, the study of the stability of $\bar{x}^*$ is transferred to investigating the stability of its associated zero equilibrium point [31].

Although the conditions of Theorem 1 are quite conservative, they ensure that the approximated set $\mathcal{M}$ is contained in the DOA. To compute the largest DOA estimate, one needs to find the maximum value of $c \in \mathbb{R}$ for the sublevel set $\mathcal{V}(c)$, described for the Lyapunov function $L(x)$ by

$$\mathcal{V}(c) = \{x \in \mathbb{R}^n : L(x) \leq c\} \tag{27}$$

such that the computed sublevel set respects the conditions defined in Theorem 1.

Theorem 2 [10]: The invariant set $\mathcal{V}(c_{\text{m}})$ provides the largest estimate for the DOA of the origin of system (2) with respect to the Lyapunov function $L(x)$ if

$$\begin{cases} c_{\text{m}} = \max\ c \\ \quad \text{s.t.}\ \mathcal{V}(c) \subseteq \mathcal{H}(x) \\ \quad\quad \mathcal{H}(x) = \{0\} \cup \{x \in \mathbb{R}^n : \dot{L}(x) < 0\}. \end{cases} \tag{28}$$

Consequently, the controller's DOA approximation is obtained by finding the maximum value of c in the optimization problem (28), which is typically solved via SOS programming [32], [33]. The SOS methods are usually used to approximate the DOAs of polynomial systems [34]. In addition to SOS programming, we use a sampling method that can compute approximations of the DOAs of both polynomial and nonpolynomial systems [11]. In this method, a sampling algorithm searches for the largest sublevel set of a candidate Lyapunov function such that the computed sublevel set satisfies the conditions of Theorem 1. Using PBC provides a system's total energy function (Hamiltonian, locally positive definite) that can be used as a candidate Lyapunov function for approximating the DOA.

Algorithm 1 Learning Sequential Composition Control Using the EB-AC Algorithm

```
Require: system (12), $A_\ell$, $x_0$, $x_d$, $\lambda$, $\gamma$, $\alpha_a$, $\alpha_c$, $n_s$, $n_t$
 1: if $P(x_0, x_d)$ is true then
 2:     Execute: sequential composition controller
 3: else
 4:     Execute: learning mode $s_\ell$
 5:     $w \leftarrow 0$
 6:     Initialize $\xi_0$, $\psi_0$
 7:     repeat
 8:         $w \leftarrow w + 1$
 9:         Initialize $x_0$
10:         $k \leftarrow 1$
11:         for $n_s$ do
12:             Energy-balancing actor-critic:
13:                 $u_k(x, \xi, \psi) = \text{sat}\big(\hat{\pi}(x_k, \xi_k, \psi_k) + \Delta u_k\big)$
14:                 Apply actor-critic RL to update $\xi_k$ and $\psi_k$
15:         end for
16:         EB-AC controller $\Phi_\ell$
17:         Estimating the DOA of the learned controller:
18:             Find the largest sub-level set $\mathcal{D}(\Phi_\ell)$
19:     until $\mathcal{G}(\Phi_i) \subset \mathcal{D}(\Phi_\ell)$
20:     Add controller $\Phi_\ell$ to the learning control automaton
21: end if
```

C. Control Algorithm

Algorithm 1 summarizes the procedure of the proposed learning sequential composition control. In this algorithm, the loop counter w counts the number of learning trials after which the DOA of the learned controller Φ_ℓ is sufficiently large to cover the goal set $\mathcal{G}(\Phi_i)$, which is reachable from the initial state. In addition, k counts the number of samples in a learning trial, n_s denotes the number of samples defined for the learning trials, and n_t represents the scheduled number of trials for the learning process.

V. Simulation and Experimental Results

We implement the proposed learning sequential composition on two nonlinear dynamical systems. In the first, we address a positioning problem in a simulated second-order system consisting of a mass with a nonlinear damper, where the control input is saturated. In the second, we study the stabilization of a physical inverted pendulum with saturated control input.

A. System 1: Nonlinear Mass-Damper

Consider a mass with a nonlinear damper, as illustrated in Fig. 4. The dynamics are given by

$$m\ddot{q} = -B(q)\dot{q} + u \qquad (29)$$

where q is the mass position measured from the origin, $B(q) = (1 - q^2)$ is the nonlinear damping coefficient and u is the control input, which is saturated at ± 3 N. The state vector of the system is described by $x = [q\ p]^T$, where $p = m\dot{q}$ is the momentum with $m = 1$ kg.

The sequential composition controller consists of two LQR controllers Φ_1 and Φ_2. Controller Φ_1 steers the mass to point

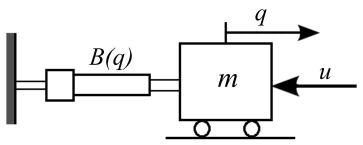


Fig. 4. Schematic of a nonlinear mass-damper system with damping coefficient $B(q)$.

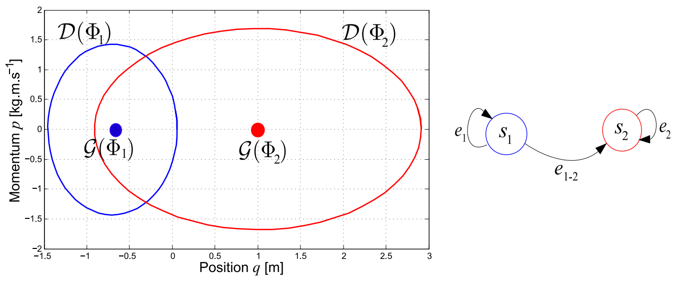


Fig. 5. Sequential composition controller designed for the nonlinear mass-damper system. (a) Approximated DOAs and goal sets. (b) Induced control automaton.

$(q, p) = (-0.7, 0)$ and controller Φ_2 to point $(q, p) = (1, 0)$. To design these controllers, the equation of motion (29) is linearized around the operating points. Then, the gain matrices for the linearized system are computed as $K_1 = [0.447\ 0.654]$ and $K_2 = [2.000\ \ 0.663]$. Fig. 5(a) presents the approximated DOAs and goal sets of controllers Φ_1 and Φ_2 and Fig. 5(b) illustrates the induced control automaton. As $\mathcal{G}(\Phi_1) \subset \mathcal{D}(\Phi_2)$, controller Φ_1 prepares controller Φ_2 via event e_{1-2}.

Suppose the controller has to drive mass m to the origin $(q, p) = (0, 0)$ from an initial state within the existing DOAs. Since the origin is not the goal set of either $\mathcal{G}(\Phi_1)$ or $\mathcal{G}(\Phi_2)$, the supervisor cannot construct a sequence of controllers to attain the desired state. Hence, the binary function P is false and the supervisor executes the learning mode s_ℓ. In this example, we apply the A-IDA-AC controller for the learning mode, which is defined by (26). The reward function is described as

$$\rho(x_{k+1}, u_k) = -\alpha_1 q_{k+1}^2 - \alpha_2 \dot{q}_{k+1}^2 - \alpha_3 u_k^2 \qquad (30)$$

which gives higher rewards to the transitions that fulfill the learning objective. The learning experiment starts exploring from the goal set $\mathcal{G}(\Phi_2)$. Since the learning process is bounded, if the learner cannot reach the origin after a number of samples, the experiment is safely reset to the goal set $\mathcal{G}(\Phi_2)$. The learning process is scheduled to run for 60 trials, each lasting 1 s. Fig. 6 presents the sum of rewards that a learning controller receives per trial over a simulated experiment with 60 trials.

The desired Hamiltonian of the system is chosen to be quadratic as $H_d(x) = x^T \Lambda x$ with $\Lambda = [1\ 0.5; 0.5\ 1]$ a symmetric positive definite matrix. We use the desired Hamiltonian as a candidate Lyapunov function for approximating the DOA of the learned controller at every trial. Since the equations of motion and basis functions are polynomial, we use SOS programming to approximate the DOAs by following the same procedure described in [10]. Fig. 7 shows the DOA of the new learned controller Φ_ℓ after seven specific trials

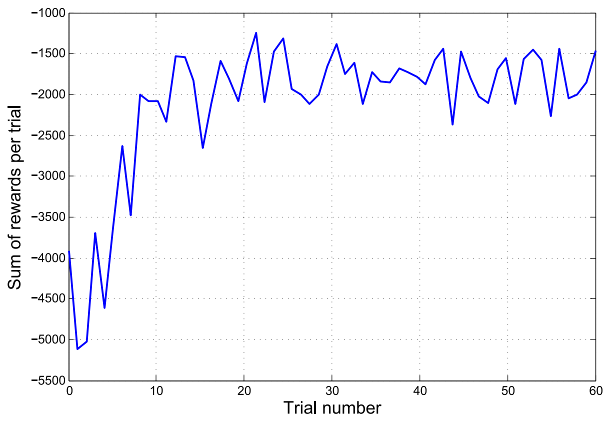


Fig. 6. Sum of rewards that a learning controller receives per trial over a simulation with 60 trials for the nonlinear mass-damper system.

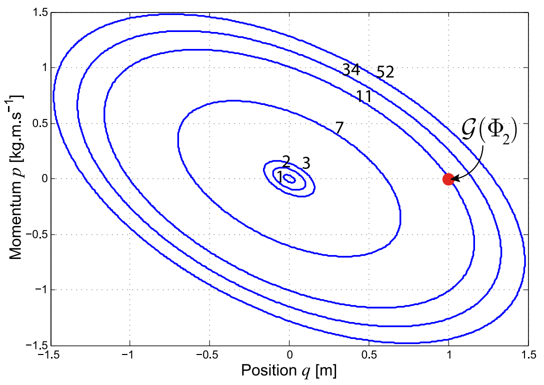


Fig. 7. Approximated DOAs of the learned controllers after seven specific trials for the nonlinear mass-damper system. The trial numbers are also indicated.

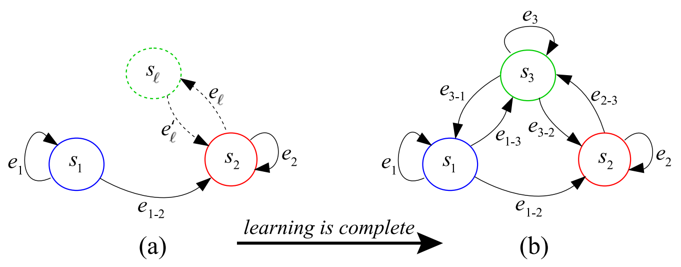


Fig. 8. Learning control automaton for the nonlinear mass-damper system. (a) During learning. (b) After learning.

(within 60 simulated trials), where the trial numbers are indicated as well. As long as learning is in progress, the approximated DOA typically enlarges, but not necessarily monotonically. Approximately after 11 trials, the DOA $\mathcal{D}(\Phi_\ell)$ is large enough to cover the goal set $\mathcal{G}(\Phi_2)$. Hence, the learning process achieves the control objective after a short amount of time while it was scheduled to run for 60 trials.

Fig. 8 illustrates the learning control automaton during and after learning. In Fig. 8(a), event e_ℓ connects mode s_2 to s_ℓ and executes the learning mode s_ℓ. Conversely, event e'_ℓ connects mode s_ℓ to s_2 and enables the supervisor to execute controller Φ_2 if the learner reaches the boundary of the union set $\mathcal{D}(\Phi_1) \cup \mathcal{D}(\Phi_2)$. When the learning objective is obtained and the goal set $\mathcal{G}(\Phi_2)$ is covered by the DOA of

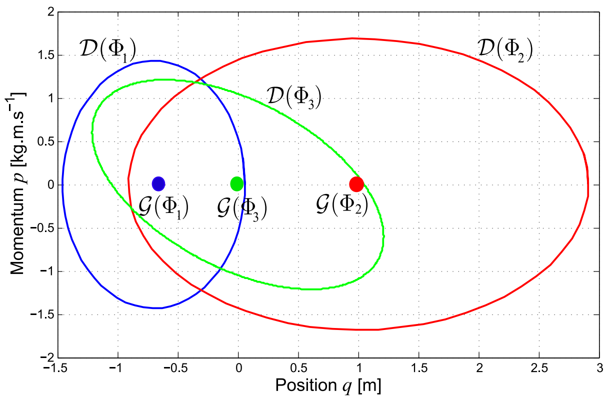


Fig. 9. Approximated DOAs and goal sets of the controllers for the nonlinear mass-damper system after learning.

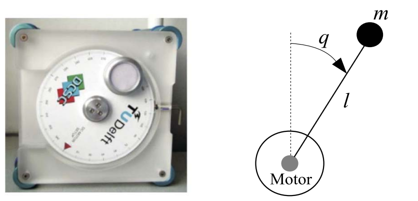


Fig. 10. Inverted pendulum and its schematic.

the learned controller, the learning process can be terminated. Fig. 9 depicts the approximated DOA of the learned controller Φ_ℓ together with the DOAs of controllers Φ_1 and Φ_2. Once learning is completed, controller Φ_ℓ is appended to the control system by introducing a new node s_3 with corresponding events "e_{2-3}", "e_{3-2}", "e_{1-3}", and "e_{3-1}". These are added to the learning control automaton with respect to the prepare relation, as shown in Fig. 8(b). Consequently, the resulting learning sequential composition controller is able to switch from controller Φ_2 to Φ_1 via the new learned controller.

B. System 2: Inverted Pendulum

The inverted pendulum, as shown in Fig. 10, is modeled by the nonlinear equation of motion

$$J\ddot{q} = mgl\sin(q) - \left(b + \frac{K^2}{R}\right)\dot{q} + \frac{K}{R}u \tag{31}$$

with q the angle of the pendulum measured from the upright position, J the inertia, m the mass, l the length of the pendulum, and b the viscous mechanical friction. Moreover, K is the motor constant, R is the electrical motor resistance, and u is the control input in Volts, which is saturated at ± 3 V. Table I presents the physical parameters of the pendulum. The values are found partly by measuring and partly estimated using nonlinear system identification.

The control system comprises two LQR controllers Φ_{up} and Φ_{down} to stabilize the pendulum at the up and down equilibria, respectively. To design these controllers, first the equation of motion (31) is linearized around the operating points $q = 0$ and $q = \pi$ with respect to the state vector $x = [q\ p]^T$. Then, the gain matrices are computed as $K_{\text{down}} = [0.025\ 72.850]$ and

TABLE I
PHYSICAL PARAMETERS OF THE INVERTED PENDULUM

Physical parameter	**Symbol**	**Value**	**Unit**
Pendulum inertia	J	1.91×10^{-4}	$\text{kg}\cdot\text{m}^2$
Pendulum mass	m	6.8×10^{-2}	kg
Gravity	g	9.81	$\text{m}\cdot\text{s}^{-2}$
Pendulum length	l	4.20×10^{-2}	m
Damping in joint	b	3×10^{-6}	Nm·s
Torque constant	K	5.36×10^{-2}	$\text{Nm}\cdot\text{A}^{-1}$
Rotor resistance	R	9.5	Ω

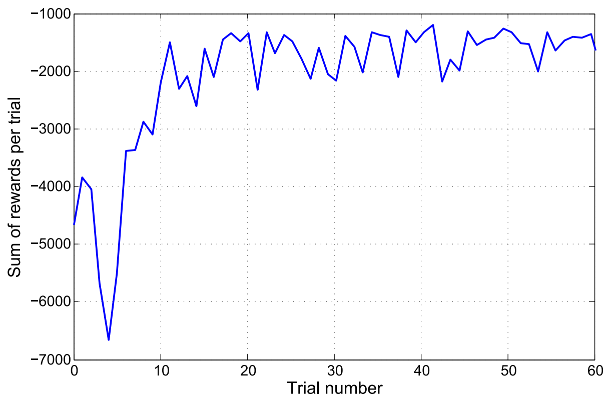


Fig. 11. Sum of rewards that a learning controller receives per trial over a simulation with 60 trials for the inverted pendulum.

$K_{\text{up}} = [8.486\ 3.439 \times 10^3]$, which generate the approximated DOAs represented in Fig. 1(b).

The control task is defined as tracking a reference string of events $S_{\text{ref}} = (\text{up down})^*$, where event up triggers mode s_{up} and event down triggers mode s_{down} in the control automaton. The initial hybrid state of the system is given by (up, 0, 0), meaning that the system starts from the continuous state (0, 0) with controller Φ_{up} is activated. According to the reference string, the initial event $S_{\text{ref}}(0) = \text{up}$ can be executed since mode s_{up} has a self triggered event up [35]. If $x \in \mathcal{G}(\Phi_{\text{up}})$, the supervisor fires the next event $S_{\text{ref}}(1) = \text{down}$. Since this event (transition from mode s_{up} to s_{down}) is feasible, the pendulum can switch to the down position by simply activating controller Φ_{down}. The next event in the reference string is $S_{\text{ref}}(2) = \text{up}$, but there is no connection from mode s_{down} to s_{up}. The supervisor triggers the learning mode s_ℓ to learn a new controller online and create the required connections through the learning control automaton. In this example, we apply the EB-AC controller for the learning mode, which is described by (20). The reward function for the learning method is defined as

$$\rho(x_{k+1}, u_k) = -\alpha_1(1 - \cos(q_{k+1})) - \alpha_2 \dot{q}_{k+1}^2 - \alpha_3 u_k^2 \quad (32)$$

which gives higher rewards to the transitions where the learning controller can stabilize the pendulum at the up equilibrium point.

Since the DOA of the down controller is the entire state space, the learning process is bounded. As such, controller Φ_{down} can safely reset each experiment to the down equilibrium point if the learner cannot reach $\mathcal{D}(\Phi_{\text{up}})$ after a number of samples. Fig. 11 represents the sum of rewards that a learning controller receives per trial over a simulated experiment with 60 trials, each lasting 1 s.

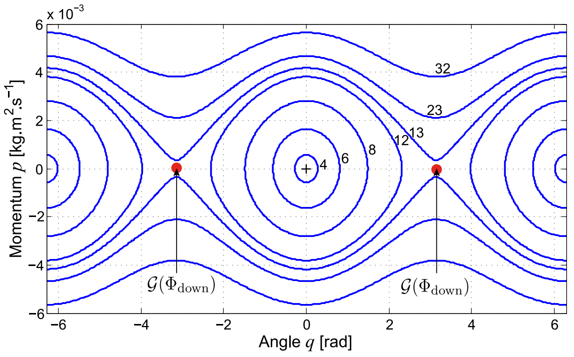


Fig. 12. Approximated DOAs of the learned controllers after seven specific trials for the inverted pendulum. The trial numbers are also indicated.

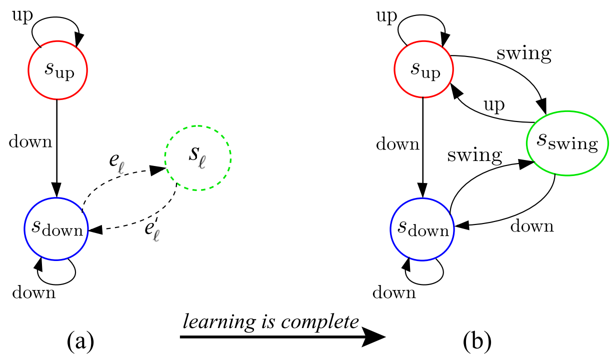


Fig. 13. Learning control automaton for the inverted pendulum. (a) During learning. (b) After learning.

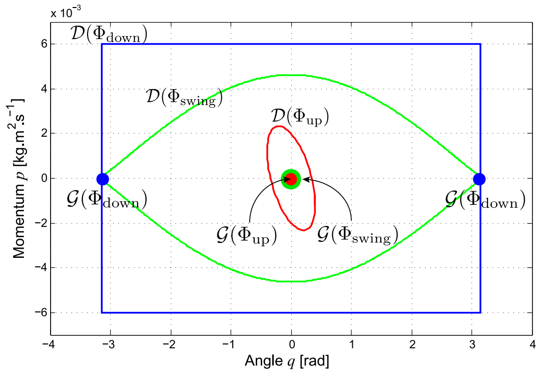


Fig. 14. Approximated DOAs and goal sets of the controllers for the inverted pendulum after learning.

Using the EB-AC method provides the Hamiltonian of the system that can be exploited as a candidate Lyapunov function for approximating the DOA of the learned controller at every trial. Since the equations of motion and desired Hamiltonian are non-polynomial, we use the sampling method to approximate the DOAs by following the same procedure described in [11]. Fig. 12 illustrates the DOA of the new learned controller Φ_{swing} after seven specific trials (among 60 trials), where the trial numbers are indicated as well. Approximately after 13 trials, the DOA $\mathcal{D}(\Phi_{\text{swing}})$ is large enough to cover the goal set $\mathcal{G}(\Phi_{\text{down}})$ and the learning process can be terminated.

Fig. 13 shows the learning control automaton during and after learning. In Fig. 13(a), event e_ℓ connects mode s_{down}

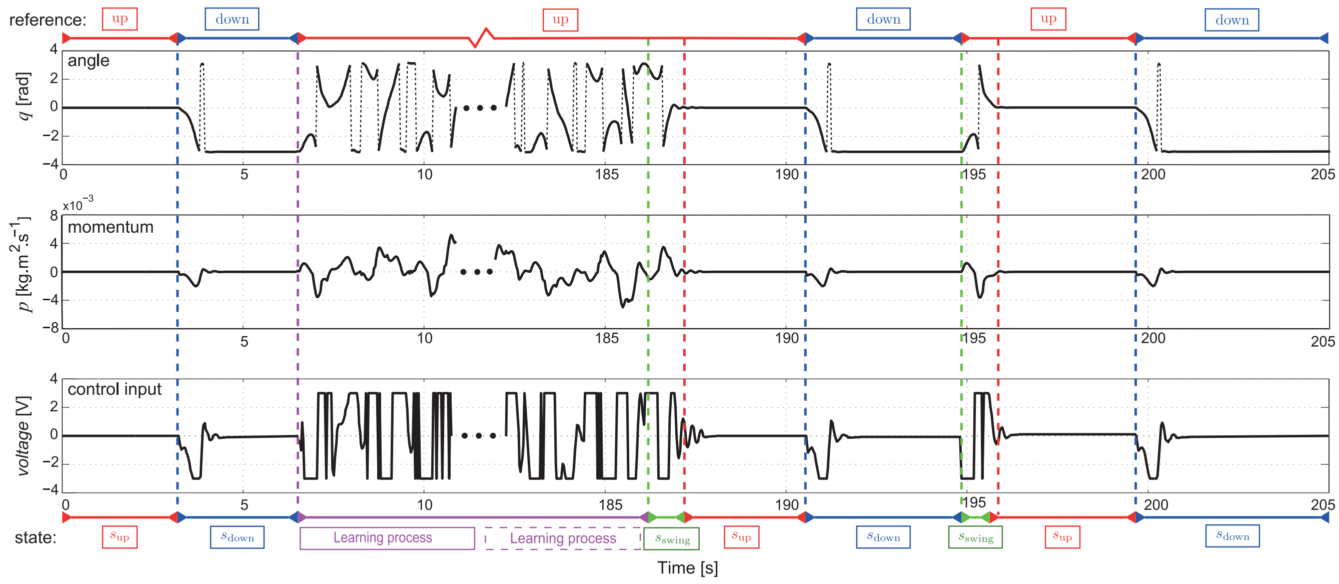


Fig. 15. Experimental results of the learning sequential composition controller for the inverted pendulum. The reference events are displayed at the top and the controllers at the bottom. When the pendulum is in the down mode and the reference event is up, the control system learns how to swing the pendulum up. The results of the learning process have been shown in part due to the space limitation. Once learning is completed, the new mode s_{swing} is added to the learning control automaton. Consequently, the resulted controller can track the reference event up by executing events swing and up, respectively. For a video that shows the implementation of the learning sequential composition control approach see: https://youtu.be/NF5ihL06SV4.

to s_ℓ and executes the learning mode s_ℓ. Conversely, event e'_ℓ connects mode s_ℓ to s_{down} and enables the supervisor to execute controller Φ_{down} if the learner reaches the boundary of $\mathcal{D}(\Phi_{\text{down}})$. When the learning goal is attained and $\mathcal{D}(\Phi_{\text{swing}})$ covers the goal set $\mathcal{G}(\Phi_{\text{down}})$, the learning process can be stopped. Fig. 14 represents the approximated DOA of the learned controller Φ_{swing} by a sublevel set of the system Hamiltonian, together with the DOAs of controllers Φ_{up} and Φ_{down}. Once learning is terminated, the new controller Φ_{swing} is appended to the control system and the new mode s_{swing} with the associated events "swing" and "up" are added to the learning control automaton with respect to the prepare relation, as shown in Fig. 13(b).

Fig. 15 illustrates the experimental results of the proposed learning sequential composition on the inverted pendulum (presented in a compressed form for the sake of space). The first time the pendulum is in mode s_{down} and the reference event is up, the learning mode s_ℓ is activated. After a number of trials (here after 13 trials), the DOA of controller Φ_{swing} is sufficiently large such that $\mathcal{G}(\Phi_{\text{down}}) \subset \mathcal{D}(\Phi_{\text{swing}})$. Although learning can be terminated at this stage, we let it to run for 60 trials for illustration purposes. Once learning is completed, the new mode s_{swing} is added to the learning control automaton. Afterward, the resulting learning sequential composition controller can track the reference event up by triggering events swing and up, respectively.

VI. Conclusion

We have developed a learning sequential composition control approach that can handle unmodeled situations using learning online. In this approach, a learning mode is added to the standard sequential composition framework to learn new controllers on a need basis when the supervisor cannot attain the desired state with its predesigned controllers. The learning process is guaranteed to be safe since it is bounded to the union of all existing DOAs. After each learning trial, the DOA of the learned controller is approximated. Once it is sufficiently large, the learning process is stopped and the new controller is added to the learning control automaton. A stopping criterion is provided for learning that can effectively speed up the learning process. Simulation and experimental results of two nonlinear systems validate the capability of the proposed control approach to cope with unmodeled situations that might occur at runtime.

The design of a generic learning experiment to learn proper control laws in a short amount of time is a real challenge since for each system the reward function, the learning rates and the parametrization of the value function must be chosen carefully. This paper did not address the automatic choice of these parameters which can be a research line for future work.

References

[1] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *Int. J. Robot. Res.*, vol. 18, no. 6, pp. 534–555, 1999.

[2] U. Nagarajan, G. Kantor, and R. Hollis, "Hybrid control for navigation of shape-accelerated underactuated balancing systems," in *Proc. 49th IEEE Int. Conf. Decis. Control*, Atlanta, GA, USA, Dec. 2010, pp. 3566–3571.

[3] V. Kallem, A. T. Komoroski, and V. Kumar, "Sequential composition for navigating a nonholonomic cart in the presence of obstacles," *IEEE Trans. Robot.*, vol. 27, no. 6, pp. 1152–1159, Dec. 2011.

[4] J. D. Weingarten, G. A. Lopes, M. Buehler, R. E. Groff, and D. E. Koditschek, "Automated gait adaptation for legged robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, USA, Apr./May 2004, pp. 2153–2158.

[5] D. C. Conner, H. Choset, and A. A. Rizzi, "Flow-through policies for hybrid controller synthesis applied to fully actuated systems," *IEEE Trans. Robot.*, vol. 25, no. 1, pp. 136–146, Feb. 2009.

[6] E. Najafi, R. Babuska, and G. A. D. Lopes, "An application of sequential composition control to cooperative systems," in *Proc. 10th Int. Workshop Robot Motion Control*, Poznan, Poland, Jul. 2015, pp. 15–20.

[7] J. Le Ny and G. J. Pappas, "Sequential composition of robust controller specifications," in *Proc. IEEE Int. Conf. Robot. Autom.*, Saint Paul, MN, USA, May 2012, pp. 5190–5195.

[8] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1038–1052, Jul. 2010.

[9] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.

[10] G. Chesi, *Domain of Attraction: Analysis and Control via SOS Programming*. London, U.K.: Springer, 2011.

[11] G. A. Lopes, E. Najafi, S. P. Nageshrao, and R. Babuska, "Learning complex behaviors via sequential composition and passivity-based control," in *Handling Uncertainty and Networked Structure in Robot Control*, L. Busoniu and L. Tamas, Eds. Berlin, Germany: Springer, 2015.

[12] D. Liberzon, *Switching in Systems and Control*. Berlin, Germany: Springer, 2012.

[13] D. B. West, *Introduction to Graph Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[14] G. Rozenberg and A. Salomaa, *Handbook of Formal Languages: Beyonds Words*. Berlin, Germany: Springer Sci. Bus. Media, 1997.

[15] E. Najafi, G. A. Lopes, and R. Babuska, "Reinforcement learning for sequential composition control," in *Proc. 52nd IEEE Int. Conf. Decis. Control*, Firenze, Italy, Dec. 2013, pp. 7265–7270.

[16] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. New York, NY, USA: Springer Sci. Bus. Media, 2008.

[17] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema, "Efficient model learning methods for actor-critic control," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 591–602, Jun. 2012.

[18] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[19] S. P. Nageshrao, G. A. Lopes, D. Jeltsema, and R. Babuška, "Passivity-based reinforcement learning control of a 2-DOF manipulator arm," *Mechatronics*, vol. 24, no. 8, pp. 1001–1007, Dec. 2014.

[20] E. Najafi, G. A. Lopes, S. P. Nageshrao, and R. Babuska, "Rapid learning in sequential composition control," in *Proc. 53rd IEEE Int. Conf. Decis. Control*, Los Angeles, CA, USA, Dec. 2014, pp. 5171–5176.

[21] A. van der Schaft and D. Jeltsema, *Port-Hamiltonian Systems Theory: An Introductory Overview*. Norwell, MA, USA: Now, 2014.

[22] O. Sprangers, R. Babuska, S. Nageshrao, and G. A. D. Lopes, "Reinforcement learning for port-Hamiltonian systems," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1003–1013, May 2015.

[23] S. P. Nageshrao, G. A. Lopes, D. Jeltsema, and R. Babuska, "Interconnection and damping assignment control via reinforcement learning," in *Proc. 19th IFAC World Congr.*, Cape Town, South Africa, Aug. 2014, pp. 1760–1765.

[24] R. Ortega and E. Garcia-Canseco, "Interconnection and damping assignment passivity-based control: A survey," *Eur. J. Control*, vol. 10, no. 5, pp. 432–450, 2004.

[25] R. Ortega, A. van der Schaft, B. Maschke, and G. Escobar, "Interconnection and damping assignment passivity-based control of port-controlled Hamiltonian systems," *Automatica*, vol. 38, no. 4, pp. 585–596, Apr. 2002.

[26] O. Hachicho, "A novel LMI-based optimization algorithm for the guaranteed estimation of the domain of attraction using rational Lyapunov functions," *J. Franklin Inst.*, vol. 344, no. 5, pp. 535–552, Aug. 2007.

[27] E. Najafi, G. A. Lopes, and R. Babuska, "Balancing a legged robot using state-dependent Riccati equation control," in *Proc. 19th IFAC World Congr.*, Cape Town, South Africa, Aug. 2014, pp. 2177–2182.

[28] G. Chesi, "Estimating the domain of attraction via union of continuous families of Lyapunov estimates," *Syst. Control Lett.*, vol. 56, no. 4, pp. 326–333, Apr. 2007.

[29] W. Tan and A. Packard, "Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming," *IEEE Trans. Autom. Control*, vol. 53, no. 2, pp. 565–570, Mar. 2008.

[30] H. K. Khalil and J. Grizzle, *Nonlinear Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[31] F. Amato, C. Cosentino, and A. Merola, "On the region of attraction of nonlinear quadratic systems," *Automatica*, vol. 43, no. 12, pp. 2119–2123, Dec. 2007.

[32] G. Chesi, "Rational Lyapunov functions for estimating and controlling the robust domain of attraction," *Automatica*, vol. 49, no. 4, pp. 1051–1057, Apr. 2013.

[33] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, May 2013, pp. 4054–4061.

[34] G. Chesi, "Estimating the domain of attraction for non-polynomial systems via LMI optimizations," *Automatica*, vol. 45, no. 6, pp. 1536–1541, 2009.

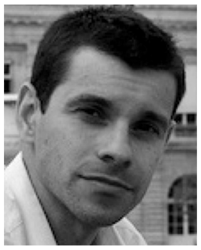
[35] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, Apr. 1994.

Esmaeil Najafi (M'13) received the B.Sc. and M.Sc. degrees in mechanical engineering from the K. N. Toosi University of Technology, Tehran, Iran, in 2007 and 2010, respectively. He is currently pursuing the Ph.D. degree with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands.

His current research interests include switching control systems, supervisory control, robotics, and machine learning.

Robert Babuška (M'15) received the M.Sc. degree (*cum laude*) in control engineering from Czech Technical University, Prague, Czech Republic, and the Ph.D. degree (*cum laude*) in control from the Delft University of Technology, Delft, The Netherlands, in 1990 and 1997, respectively.

He has had faculty appointments with the Department of Technical Cybernetics, Czech Technical University, and the Electrical Engineering Faculty, Delft University of Technology. He is currently a Professor of Intelligent Control and Robotics with the Delft Center for Systems and Control, Delft University of Technology. His current research interests include reinforcement learning, neural and fuzzy systems, identification and state-estimation, nonlinear model-based and adaptive control, and dynamic multiagent systems.



Gabriel A. D. Lopes (M'03) received the M.Sc. and Ph.D. degrees in electric engineering and computer science from the University of Michigan, Ann Arbor, MI, USA, in 2003 and 2007, respectively.

He was with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA, from 2005 to 2008. He is currently an Assistant Professor with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. His current research interests include nonlinear control, robotics, discrete-event systems, and machine learning.