Automatic synthesis of supervisory control systems

Najafi, Esmaeil

DOI
10.4233/uuid:c26ff0a0-366d-49e7-bfc7-2892e9a2e2a9

Publication date
2016

Document Version
Final published version

Citation (APA)
Najafi, E. (2016). *Automatic synthesis of supervisory control systems*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:c26ff0a0-366d-49e7-bfc7-2892e9a2e2a9

Automatic Synthesis of Supervisory Control Systems

Esmaeil Najafi

Cover illustration: The foreground presents the collection of learned controllers using the probabilistic learning trees algorithm. The stable equilibria are indicated by the numbers and each enclosing colored circle represents an estimate of the domain of attraction for a learned controller.

Cover design: *Dr. Massoud Tohidian*

AUTOMATIC SYNTHESIS OF SUPERVISORY CONTROL SYSTEMS

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op

maandag 30 mei 2016 om 10:00 uur

door

Esmaeil NAJAFI

Master of Science in Mechanical Engineering
K. N. Toosi University of Technology, Iran
geboren te Tehran, Iran

Dit proefschrift is goedgekeurd door de:

Promotor: Prof. dr. R. Babuška
Copromotor: Dr. G.A.D. Lopes

Samenstelling promotiecommisie:

Rector Magnificus,	voorzitter
Prof. dr. R. Babuška,	Technische Universiteit Delft, promotor
Dr. G.A.D. Lopes,	Technische Universiteit Delft, copromotor

Onafhankelijke leden:

Prof. dr. ir. J. Hellendoorn,	Technische Universiteit Delft
Prof. dr. ir. P.P. Jonker,	Technische Universiteit Delft
Prof. dr. A. Nowé,	Vrije Universiteit Brussel
Dr. M. Corno,	Politecnico di Milano
Dr. R. Carloni,	Universiteit Twente

This dissertation has been completed in fulfillment of the requirements of the Dutch Institute of Systems and Control (DISC) for graduate study.

The requirement of the TU Delft Graduate School for the Doctoral Education Program has been fulfilled.

ISBN: 978-94-6186-656-1

Printed in the Netherlands.

Dedicated to my wife for her patience,
support, and unconditional love

Acknowledgments

This thesis is the result of four and a half years of research and study at the Delft Center for Systems and Control (DCSC) of Delft University of Technology. This is most certainly not the result of individual work. Many people were directly or indirectly involved and have contributed to the final result. It is my great pleasure to dedicate these words of appreciation to them for their contributions throughout this endeavor.

I would like to thank my promoter Prof. dr. Robert Babuška for giving me the opportunity to do a PhD and for the trust and support he gave me during these years. Robert, I thank you very much for your commitment to grow your students scientifically. I would like to express my great gratitude to my supervisor Dr. Gabriel A.D. Lopes. I kindly appreciate his friendship, scientific advice, and many insightful discussions and suggestions. Gabriel, you are not only a supervisor, but also a great friend who cares about other aspects of your students' life.

I am grateful to all the colleagues in DCSC. I really enjoyed the time I spent with you all during these years. I want to thank Fankai Zhang for his help at the DCSC Robotics Lab, Mohsen Alirezaei and Sadegh Esmaeil Zadeh for their friendly and scientifically discussions at the beginning of my PhD, Subramanya P. Nageshrao for his collaboration during my PhD research, Anuj Shah for his great job during his MSc project working with me, and Cees Verdier for the translation of my thesis's summary. My special thanks is reserved for my best colleague and friend, Mohammad Shahbazi. We have been together for a very long time, since 2003. I kindly thank you for sharing your opinion and experience with me in these years.

My appreciation is extended to my Iranian friends in the Netherlands who have been a great support for me and my family. With the risk of forgetting someone who I definitely should have mentioned, I would like to thank families Shahbazi, Mehrara, Derakhshani, Madadi, Chahardowli, Latifi, Alemi, Bornaee, Mirzaei, Bakhshandeh, Ghaemi Nia, Tohidian, Zadpoor, Behdani, Hesan, Abbasi, Saeedi, Mianabadi, Boroumandzadeh, Abouhamzeh, Alirezaei, Fasihi, Rahimi, Kavian, Mohammadi, Ramazi, Hosseini Nasab, Ahmadi Mehr, Mir Mohammadi, Sedighi, Amani, Rostampour, Monadi, and all members of Hey'at Mohebban Al-Mahdi.

I would like to express my great appreciation to my parents. It is beyond my ability to express in only a few words how much both of you have inspired me throughout my life. I believe you are the best father and mother who have kindly and generously dedicated your life towards the success of your children. My special thanks to both of you for your support, kindness, love and blessing. I would like to extend my appreciation to both my supportive brothers and kind sister. I am grateful of you and your respected families. I kindly thank you for your dedication, support, and kindness. Thank you my great family.

Last but not least, I would like to express my deepest thanks and appreciation to my lovely wife, Maryam, for her patience, support, kindness, and unconditional love. Whatever we have achieved during these years could not happened without your help. To be honest, I cannot express my highest appreciation in only a few words. Thank you very much for everything. I would like to thank your respected family for their support and blessing always backing me up. Hossein, my lovely son, you are the best gift from God to me and your mother. I thank you very much for all the great time you have provided for us.

My dear God, I thank you so much for whatever you have given to me in my life.

Esmaeil Najafi
Delft, May 2016

Contents

1 CHAPTER

Introduction

This thesis addresses automatic synthesis of supervisory control systems. This chapter describes the thesis focus, research goals, and the main contributions. It provides the thesis outline as well as the corresponding publications.

1.1 Motivation

One practical approach to controller synthesis for nonlinear dynamical systems is that instead of designing a single nonlinear controller, one constructs a set of simpler, possibly linear, controllers, each tuned for a specific region of the state space. In the closed loop, as the state follows a specified trajectory, a supervisory mechanism switches sequentially from one controller to another. This approach is termed sequential composition [13].

Sequential composition is a supervisory control methodology that focuses on the interaction between a collection of pre-designed controllers. Each controller has a domain of attraction (DoA), a region of the state space in which the controller is active [23], and a goal set. The supervisor can instantly switch from one controller to the another controller if the goal set of the first controller is within the DoA of the second, called the prepare relation [13]. If the local controllers were properly coordinated with respect to the prepare relation, the union set of their DoAs would be significantly larger than the DoA for any one of the pre-existing feedback controllers [71].

Sequential composition uses the set of DoAs backchaining to generate a "path" to a desired goal. Once the path, a sequence of controllers, is computed on the symbolic level, the supervisor executes the task by triggering corresponding controllers following the sequence. If the prepare relation is satisfied when the supervisor switches between the controllers, switching will be safe and the system will

be stable with no chattering phenomena [25]. In sequential composition, the relation between controllers is represented by a supervisory finite-state machine [89].

Applications of sequential composition include, for instance, balancing of an underactuated system [79], navigation of an autonomous mobile robot [54, 131], navigation of fully actuated dynamical systems through cluttered environments [26], etc. The standard sequential composition framework has been extended in several ways. In [64], robust controller specifications are composed sequentially. Additionally, linear quadratic regulator trees (LQR-trees) [122] 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.

Sequential composition has some resemblances with other supervisory techniques such as gain scheduling, which traditionally was one of the most common systematic approaches to control of nonlinear dynamical systems in practice [5, 111]. A typical gain scheduling control system comprises of two main components: a set of controllers, and a supervisor (scheduler) that assigns a controller to the system at every time step. The supervisor design consists of two steps: first to define the scheduling variables such that the nonlinearities are captured [78], and second to select a supervisory algorithm for choosing the local controllers on the basis of variables defined a priori [78]. However, the supervisor design in sequential composition differs from gain scheduling by offering the prepare relation as a switching rule. Using this relation not only guarantees the switching safety and system stability, but also automates synthesize of the supervisor [85].

Although sequential composition provides an effective supervisory architecture, it cannot address the task for which no controller was defined a priori. This arise a question whether it is possible to automatically augment an existing control system with new controllers on demand, without changing the supervisory structure. Moreover, sequential composition controllers are typically designed for isolated systems. However, when the collaboration of multiple systems is required to fulfill a control specification, an extra mechanism is needed.

This thesis studies automatic synthesis of supervisory control systems using the paradigm of sequential composition. First, a learning sequential composition control technique is developed to learn new controllers by means of reinforcement learning (RL) on demand. Once learning is complete the supervisory control structure is augmented with the new learned controllers. As a consequence, the overall area of the state space in which the supervisor can be active gets incrementally larger upon request. Second, a cooperative sequential composition control algorithm is proposed to enable the coordination between a set of sequential composition controllers without any change in their low-level structures. Finally, the described supervisory architecture is applied to a robotic language, designed for the manipulation of multiple objects by multiple robots.

1.2 Research Goals and Contributions

The main research goal of this thesis, synthesize supervisory controllers automatically, is translated to the following research questions.

- How to estimate the domain of attraction of a controller in real-time?
- How to augment a control system with a new online controller learned?
- How to cooperate between multiple supervisory control systems?
- How to synthesize a supervisory controller for robotic manipulation?

Estimating the domain of attraction. The DoA of a stable equilibrium is a region of the system's state space from which each trajectory starts, eventually converges to the equilibrium. Several techniques have been introduced in the literature to compute an inner approximation for the DoA [23]. However, most of the existing methods are limited to polynomial systems [46, 124]. They are computationally costly and time-consuming which make them unsuitable for real-time implementation [22]. This thesis proposes a fast sampling method for estimating the DoAs of nonlinear systems [84]. This method is computationally effective, compared with the existing optimization-based techniques, and is beneficial for real-time applications. Estimating the DoA is a tool required for the synthesis of controllers in the context of sequential composition.

Learning sequential composition control. Sequential composition constructs a supervisory finite-state machine for a set of pre-designed controllers, each endowed with a DoA and a goal set [25]. By design, if the goal set of one controller lies in the DoA of another controller, the supervisor can instantly switch from the first controller to the second without affecting the stability and convergence of the system. As these controllers are designed offline, sequential composition cannot address the tasks for which no controller is available for the supervisor. This thesis develops a learning sequential composition control approach that augments the given pre-designed control system by learning new controllers online, using the actor-critic RL method [85, 89]. The learning process is always safe since the exploration for new controller can only takes place within the DoAs of the existing controllers. This learning control technique is also extended for situations where no controller exist initially and all controllers have to sequentially be synthesized so as to achieve the control objective [71].

Cooperative sequential composition control. The standard sequential composition is typically designed to control isolated systems [26, 91]. However, for tasks that require collaboration of multiple systems extra mechanisms are required. This thesis describes a cooperative sequential composition control algorithm that composes multiple sequential composition controllers to accomplish collaborative

behavior [83]. In this approach, the sequential composition controllers communicate with each other to share their system dynamics and low-level structures. Using this data together with the interaction dynamics enables computation of the DoAs of the resulting composed controllers. Based on the prepare relation defined between the DoAs, the original supervisors are augmented with new connections through their low-level controllers. Applying these events, the cooperative control system can fulfill the tasks which are not possible to satisfy with the original controllers individually.

Robot contact language. Dexterous manipulation tasks involve decision-making at various stages of planning and execution [41, 16]. This thesis studies the synthesis of supervisory control systems for robotic planning and manipulation [110]. The problem of dividing a manipulation task is addressed to obtain an appropriate sequence of sub-tasks with regards to the contact-based task division. A robot contact language is defined for robotic manipulation based on making and breaking contact between the involved components, namely robots, objects, and surfaces. This planner is modular enough to deploy geometrical and physical information of the components and translate supervisory planning to low-level robot controllers.

1.3 Outline of the Thesis

This thesis starts with a brief review on the background and preliminaries required for the proposed control approaches and then presents the original contributions. The thesis is organized as follows.

- **Chapter 2** describes sequential composition approach with a quick review of its application and other supervisory techniques. This chapter continues with a brief description on RL methods and discusses the main concepts of passivity-based learning control.
- **Chapter 3** proposes a fast and computationally effective sampling method to approximate the DoAs of nonlinear systems in real-time. This method is validated to estimate the DoAs of stable equilibria in several nonlinear systems. In addition, it is deployed for the passivity-based learning controller designed for a magnetic levitation system.
- **Chapter 4** proposes a learning control algorithm that augments the standard sequential composition with a learning module to cope with unmodeled situations that might occur during runtime. The proposed approach is implemented on two nonlinear systems: nonlinear mass-damper system and inverted pendulum. This control approach is extended for situations where there is no controller in the supervisory structure initially. This algorithm is simulated for the navigation of a simple mobile robot through a landscape.

- **Chapter 5** extends the standard sequential composition by introducing a novel control approach to compose multiple sequential composition controllers towards cooperative control systems. This control methodology is implemented on collaboration of an inverted pendulum with two second-order DC motors for cooperative maneuvers.
- **Chapter 6** describes a contact language for robot manipulation planning. When contact between the involved components are made or broken, the system's dynamics change. Using this paradigm the robot manipulation planner is developed. This robot language is validated for three different case studies, each with a specific control objective.
- **Chapter 7** concludes that the control approaches proposed throughout this thesis together enable automatic synthesis of a class of supervisory control systems that employ the paradigm of sequential composition. This chapter closes the thesis with some recommendations for future research.

The diagram in Figure 1.1 illustrates the connection between the chapters. It gives an overview on the structure of this thesis.

1.4 Publications by the Author

The material presented in Chapters 3, 4, 5, and 6 has been published as peer-reviewed articles in international journals, a chapter in a robotic control book, and papers in the proceedings of international conferences. There are also some manuscripts that will be ready for submission in the near future. The relation between every chapter and the corresponding publications is outlined as follows.

- Chapter 3 is based on [84]. The interested reader may refer to [90] for the application.
- Chapter 4 is based on [85, 71] and the manuscript [87]. The interested reader may refer to [89, 91] for more discussion.
- Chapter 5 is based on [83, 88] and the manuscript [86].
- Chapter 6 is based on [109, 110].

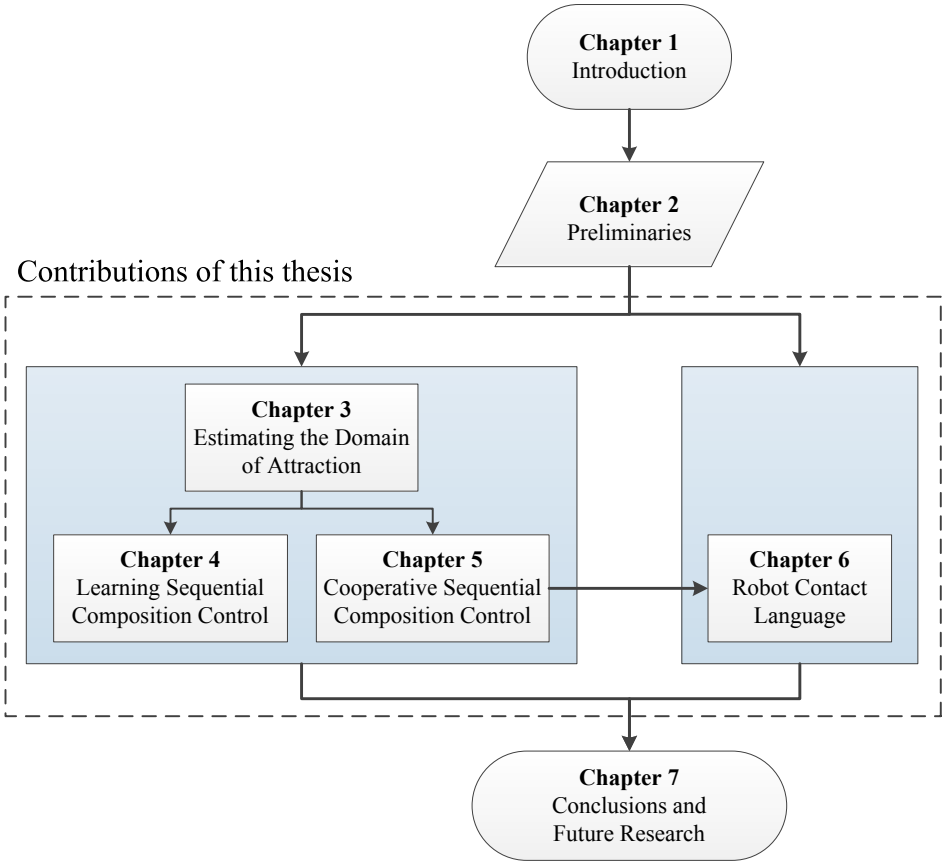

Figure 1.1: Structure of this thesis.

2 CHAPTER

Preliminaries

This chapter discusses the preliminaries for the control techniques proposed in the thesis. It describes sequential composition control and reviews the main concepts of RL methods. Then, it discusses passivity-based learning control.

2.1 Introduction

Control synthesis for dynamical systems spans the fields of systems and control and computer science, ranging from model-based state-feedback control to motion planning techniques. Systems and control theory provides tools for analyzing stability and synthesizing controllers for systems with complex dynamics, but typically simple control specifications [90]. On the other side, computer science tools address complex control specifications for simple dynamical systems [62, 56]. For example, the design of a controller for an autonomous humanoid robot consists of low-level controllers for dynamic balancing, designed using tools from systems and control, and high-level controllers for task-oriented control, such as grasping or navigation in a cluttered room, designed using motion planning techniques from computer science.

Sequential composition [13], emerging from the systems and control field, aims to cope with rich control specifications on dynamical systems. It offers a natural framework for control design since it decomposes a given task into smaller problems, each solved in a traditional control systems manner, taking advantage of all the available tools such as feedback/feedforward design, optimal control, robust control, and etc. Sequential composition typically results in a simple supervisory finite-state machine, with each node consisting of specially crafted controllers that can have large DoAs. Although sequential composition accomplish pre-defined tasks well, it cannot fulfill situations for which no controller was designed a priori.

The use of learning in the context of sequential composition is proposed to enable automatic synthesis of supervisory controllers. As such, RL methods [116] are briefly reviewed, namely actor-critic algorithm [44], which is convenient for problems with continuous state and action spaces. Passivity-based learning controllers described in [112] are another element that are used for the proposed control approaches. The dynamic equations together with the total energy of the system are deployed for estimating the DoAs of learning controllers.

This chapter is organized as follows. Section 2.2 describes sequential composition control and enumerates its application in robotics. A discussion about alternatives to sequential composition is presented at the end of this section. Section 2.3 reviews the main concepts of RL methods and Section 2.4 discusses passivity-based learning controllers. Finally, Section 2.5 provides a brief discussion on these elements and concludes the chapter.

2.2 Sequential Composition Control

Sequential composition is a supervisory control approach that address complex dynamical systems. It focuses on the interaction between a collection of predesigned controllers, each endowed with a DoA and a goal set [23]. Sequential composition uses a set of DoAs backchaining to generate a path to a desired goal. Once the path, a sequence of controllers, is computed on the symbolic level, the controller executes the task by triggering corresponding controllers following a particular sequence. Consider the dynamical system

$$\dot{x} = f(x, u) \tag{2.1}$$

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

$$\dot{x} = f\big(x, \Phi_i(x)\big) = f_i\big(x\big). \tag{2.2}$$

Let x_i^* be a stable equilibrium of the closed-loop system (2.2). The goal set of controller $\Phi_i(x)$, denoted $\mathcal{G}(\Phi_i) \subseteq \mathcal{X}$, is described by

$$\mathcal{G}(\Phi_i) = \{x_i^*\}. \tag{2.3}$$

Note that in general the goal sets of controlled systems can have oddly shapes. For the purpose of this chapter we assume only stabilizing controllers to a point in the state space. 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 (2.2) at time t,

subject to the initial condition, the DoA of controller Φ_i is defined by the set

$$\mathcal{D}(\Phi_i) = \{x_0 \in \mathcal{X} : \lim_{t\to\infty} x(t, x_0) = \mathcal{G}(\Phi_i)\}. \tag{2.4}$$

It is assumed that every controller can be illustrated by a funnel [13], as shown in Figure 2.1, where the funnel's height determines the value of the candidate Lyapunov function $L_i(x)$, the set $\mathcal{D}(\Phi_i)$ represents the DoA of controller $\Phi_i(x)$, and $\mathcal{G}(\Phi_i)$ illustrates its goal set. When a controller is executed, the value of $L_i(x)$ decreases and the system trajectory converges to the controller's goal set.

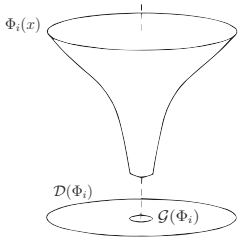

Figure 2.1: Representation of controller $\Phi_i(x)$ based on its candidate Lyapunov function as a funnel. The sets $\mathcal{D}(\Phi_i)$ and $\mathcal{G}(\Phi_i)$ illustrate the DoA and goal set of controller $\Phi_i(x)$ respectively, adopted from [25].

It is assumed that system (2.1) 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 if $\mathcal{G}(\Phi_i)$ is the subset of $\mathcal{D}(\Phi_j)$, that is

$$\Phi_i \succeq \Phi_j \;\text{ if }\; \mathcal{G}(\Phi_i) \subset \mathcal{D}(\Phi_j). \tag{2.5}$$

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 [66]. Consider a sequential composition controller with three control laws Φ_1, Φ_2, and Φ_3, such that each drives the system trajectories that lies in its DoA to its goal set, as shown in Figure 2.2. Based on the prepare relation, the final goal is attained by composing the controllers in a proper sequence. Figure 2.2 on the left side illustrates the controllers' DoAs by their representative funnels and on the right side, represents the induced supervisory finite-state machine.

Sequential composition is beneficial for planning multiple tasks in the space of

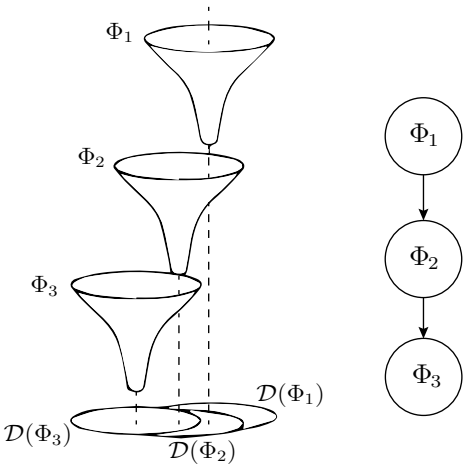

Figure 2.2: Prepare relation between three local controllers: the DoAs of the controller with their corresponding Lyapunov functions, and the induced supervisory finite-state machine, adopted from [25].

control laws and execute them based on the prepare relation. Typically, planning over the discrete space of the policies is easier than planning over the continuous space and more flexible with respect to the high-level control specifications [27]. Consider the navigation of a mobile robot through a structured environment with some obstacles. A sequential composition controller is designed, as shown at the top of Figure 2.3 with its local control laws to navigate the robot to the final goal. Figure 2.3 at the bottom presents the induced finite-state machine with transitions between the controllers based on the prepare relation.

As illustrated in Figure 2.4, to obtain the desired state G from different initial states S_1, S_2, S_3, and S_4 the sequential composition controller executes a specific sequence of controllers to derive the system trajectory from the initial state to the desired state G.

In sequential composition, the set of controllers and their interactions are represented by a supervisory finite-state machine that we call control automaton. Each mode of the control automaton, indexed by i, describes a tuple $s_i \in \mathcal{S}$ as

$$s_i = \{\Phi_i, \mathcal{D}(\Phi_i), \mathcal{G}(\Phi_i)\} \tag{2.6}$$

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 [132], and the union of DoAs

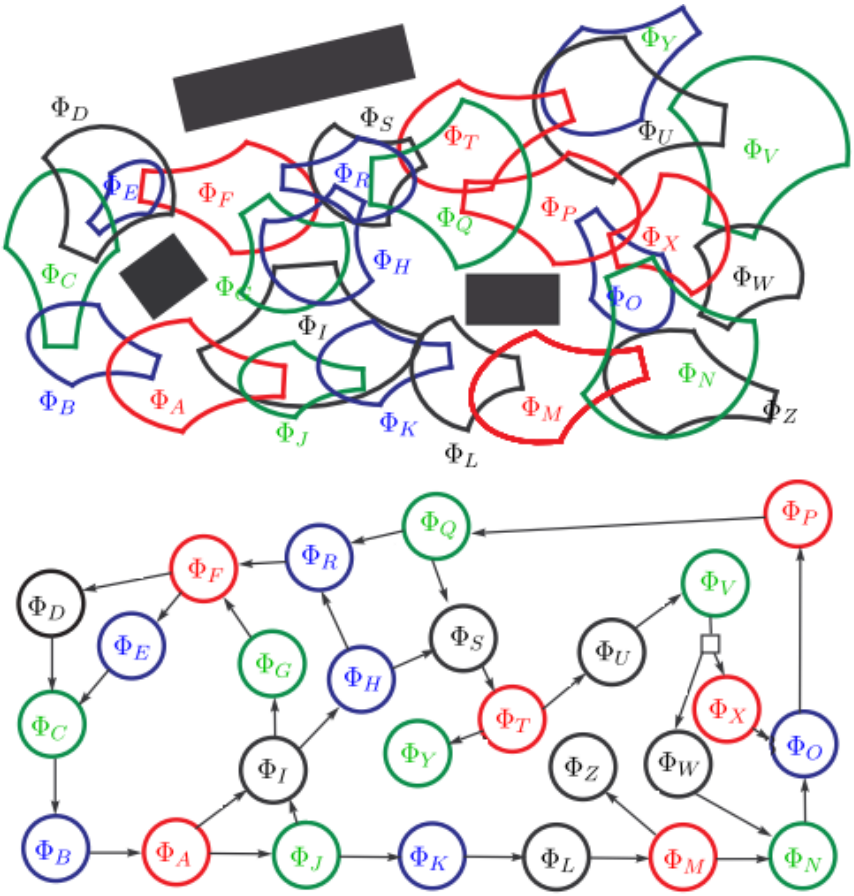

Figure 2.3: Sequential composition controller for navigation of a mobile robot through a structured environment in presence of obstacles. The top figure illustrates the DoAs of controllers in the state space and the bottom graph depicts the induced finite-state machine, adopted from [25].

covers the entire state space, i.e.,

$$\mathcal{D}(\Phi) = \bigcup_{\Phi_i} \mathcal{D}(\Phi_i) = \mathcal{X}. \tag{2.7}$$

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. The idea of sequential composition has been successfully implemented on several robotic systems. Some examples are described in the following.

Burridge et al. [13] implemented sequential composition on a robot juggling a ball by repeatedly batting the ball with a paddle. They defined a notion of generic

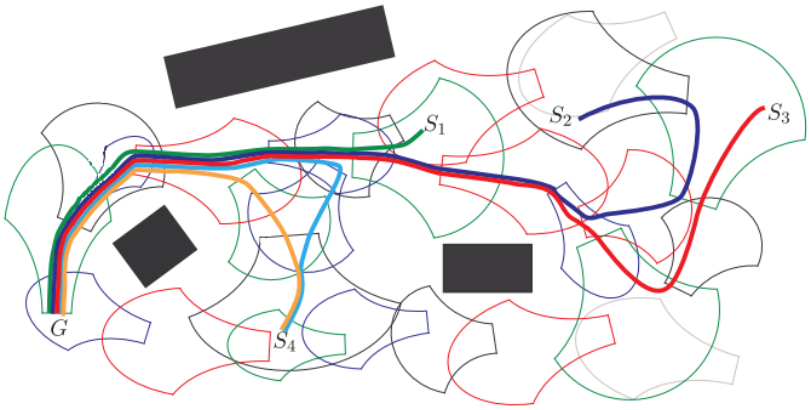

Figure 2.4: Induced paths (sequence of controllers) from different initial states to the desired state G through the state space in the example of mobile robot navigation, adopted from [25].

control policy which is indeed a control law with free parameters. A set of generic control policies generates a "palette". The experimental results illustrate that if the policies are composed properly, the robot can juggle the ball through its work space, while avoiding the obstacles. According to the obtained results, it is concluded that sequential composition is inherently robust even in the presence of perturbation since the designed controller repeatedly brought the ball into its desired state.

Rizzi [104] used sequential composition to simplify motion planning for a holonomic second-order dynamical system with velocity and acceleration constraints. He specified a particular goal set for each control policy to lie within the overlapping convex polytopes. If a collection of polytopes are composed together with appropriate goal sets, the system will be derived to the overall desired state via composing a sequence of controllers sequentially. If the initial state lies in the DoA of a controller, it will finally converge to the desired state. Moreover, Yang and LaValle [134] developed a similar approach to address kinematic systems without considering the input constraints. They described a potential function over a ball in the configuration space. They showed that while there are a number of balls throughout the configuration space, the overlapping balls create a similar function to the polytopes.

Quaid and Rizzi [102] extended the standard sequential composition to a more sophisticated approach that takes into account the constraints described on acceleration and velocity. They applied their suggested approach on planar robots and improved the safety of control systems, specifically in multi-robots environments. Later, Kantor and Rizzi [55] implemented sequential composition to control underactuated wheeled mobile robots. They defined a set of visual control policies for a nonholonomic unicycle with constraint on the view field. They applied vari-

able constraint control to define each specific control policy. Patel et al. [99] used sequential composition to describe a set of control policies for a nonholonomic wheelchair for navigation through a doorway.

Weingarten et al. [131] implemented sequential composition for legged robots. They developed a supervisory finite-state machine as a high level control framework to properly switch between the controllers and obtain the control objective. Figure 2.5 at the top illustrates the workspace of a mobile robot with subdivisions: Servo home, Experiment, and Stabilizing. For each part of the workspace a specific controller is activated by the supervisor such that the control objective is finally achieved. Figure 2.5 at the bottom represents the finite-state machine of the control system with all designed controllers. The control law "Servo home" which is the initial controller can be executed throughout the state space, where the supervisor get stuck due to unforeseen situations.

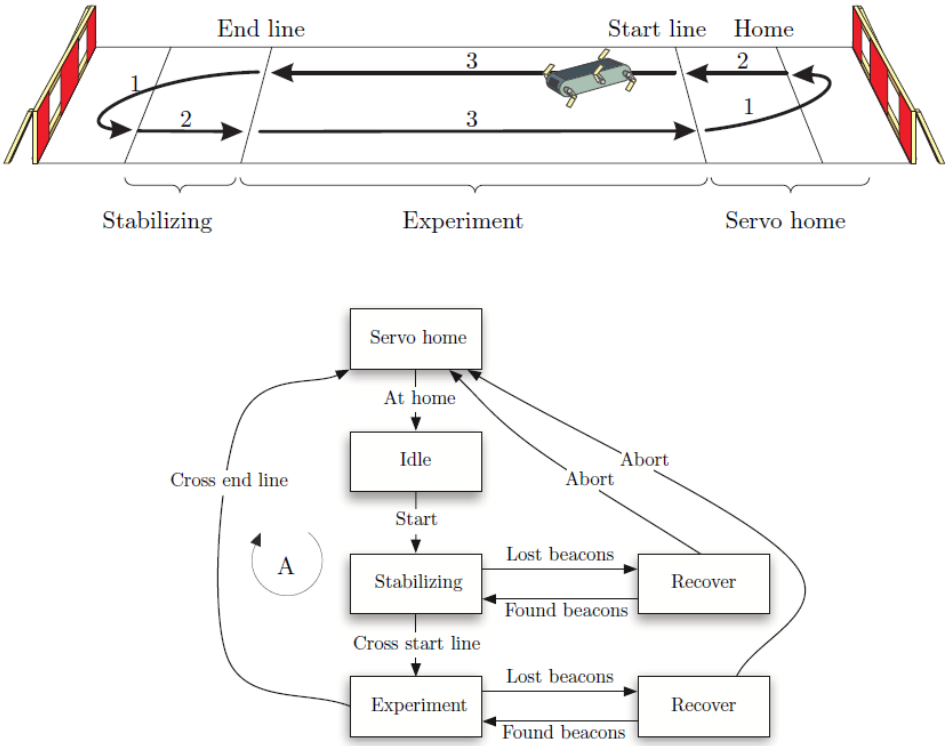

Figure 2.5: a) An example of control policy composition for a legged rescue robot, where a high-level task is addressed by a set of simple controllers. The finite-state machine represents transitions among the controllers, adopted from [131].

kallem et al. [54] used sequential composition for navigation of a nonholonomic robot in the presence of obstacles. They decomposed the free workspace of the

system into triangular tori and composed local feedback controllers, each associated to a particular torus, such that the obtained sequence of controllers is able to drive the robot from one cell to its neighbor cell and hence navigates the robot to its desired state consequently. They implemented the composition approach on a group of robots in cluttered environments [6]. Nagarajan et al. [79] implemented sequential composition for navigation of shape-accelerated underactuated balancing systems with dynamic constraints. They extended the concept of sequential composition to discrete state-based switching control approach and proposed a globally asymptotically convergent feedback policy. The motion policies are designed such that their composition produces an overall graceful motion [80]. In fact, an automatic control algorithm deploys motion policies and a supervisory framework switches between the policies.

Le and Pappas studied the composition of robust controllers and presented a general notion of robust controller specifications with a mechanism to compose them sequentially [64]. Conner et al. [28] defined the idea of flow-through policy, where each individual controller is activated once its previous controller with higher priority has been executed. This creates a flow of control policies. They developed a generic class of control policies that respects nonholonomic constraints [25]. The results show that the proposed method works safely for a convex-bodied mobile robot with respect to the obstacles since each local controller satisfies the system constraints over its associated region through the state space. They implemented the flow-through policies approach to synthesis a hybrid controller to be able to address the coupled navigation and control problems of fully actuated dynamical systems that operate in cluttered environments [26].

Lindemann and LaValle [68] extended the flow-through of policies approach and defined flow-through vector fields over disjoint regions over the work space. They focused on theoretical completeness and smoothness of simple dynamical systems and defined a different vector field technique to extend their approach into cylindrical algebraic decompositions [67]. They studied nonholonomic systems with bounded steering and unbounded control inputs [69] and presented an effective approach for computing feedback control laws in the presence of obstacles [70]. Instead of computing a trajectory between a pair of initial and goal states, their proposed algorithm computes a vector field over the entire state space such that all trajectories attain their desired states. By partitioning the state space into simple cells, a vector field is constructed. An appropriate interpolation between these local vector fields results in a global vector field that can solve navigation problem and provide robustness for the system with regards to disturbances.

In addition to the standard sequential composition and its extensions, there exist other mechanisms that can be classified as composition based approaches. Here, we review a few of these control schemes as alternatives to sequential composition that use the idea of composition to construct a supervisory control structure.

Minler [75] introduced the notion of bi-simulation equivalence, the relation be-

tween a system and a model that simulates it, to reduce the complexity of modeling dynamical systems. Bi-simulation is a supervisory control approach that arose with a computer science mindset. Due to the challenges in designing controllers for nonlinear systems, it is advantageous to look for symbolic models that can represent or approximate the dynamics of a continuous-time dynamical system [117]. This leads to transforming the control synthesis problem into a search on a graph [105]. Once the system is represented in the symbolic domain, rich control specifications can be implemented [101] and properties verified [42]. Such flexibility comes at a high cost. The accurate representation of even simple dynamical systems can at times require millions of nodes in a graph. Moreover, if the environment is dynamical it can be difficult to update the graph online. These challenges have limited the applicability of bi-simulation methods in robotics.

Konidaris and Barreto [59] introduced the skill discovery method where the state space is partitioned into a number of sub-domains, called options, to construct chains of skills, which is analogous to sequential composition. Tedrake [121] introduced LQR-trees as a feedback motion planning technique which is established based on the composition approach. This algorithm combines a set of local linear quadratic regulators to make a tree that can stabilize the planned trajectories computed by local optimizers and then cover the entire state space [103]. The LQR-trees operates by growing a tree of stabilized and verified trajectories backwards from a desired state. At each step, a random state is drawn from the state space. If the chosen state is inside the DoA of an existing trajectory it will be discarded, otherwise a local trajectory optimizer looks for a new trajectory that connects this random state to the generated tree and so to the desired state. After that, the new trajectory is stabilized and verified, and then the process repeats again to construct a comprehensive tree [122].

In the field of quantized control systems, Bicchi et al. [10] studied finite abstraction of a certain class of control systems with quantized inputs. Moreover, their research was continued in the field of digital control systems where the control signals are piecewise-constant. They showed that if a system is incrementally input-to-state stable, by using a proper quantization in the space of inputs, symbolic models can be generated for a system [101].

Besides these composition methods, symbolic planning techniques have been established to satisfy high-level control specifications. Linear temporal logic (LTL) combines the standard boolean operators such as “and”,“or”, and “not” with temporal operators such as “next” and “always” to develop an appropriate transition relation in symbolic models [58]. Fainekos et al. [35] developed an automaton that uses specifications of the LTL to describe the behaviors of a system with a prepare graph. This approach allows the system to use a set of discrete events to react to the environmental changes effectively.

2.3 Reinforcement Learning

Reinforcement learning is an optimization method in which an optimal controller is learned by interacting with the system [116]. A RL problem can be defined by a Markov decision process defined as 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 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. Note that here a discrete-time deterministic system is considered with $x_k = x(T_s k)$ for a given sampling time T_s. The learning objective is to find an optimal policy $\pi:\mathcal{X}\to\mathcal{U}$ to maximize the discounted sum of expected instantaneous rewards, which is stored as the value function

$$\begin{aligned} V^{\pi}(x_k) &= \textstyle\sum_{j=0}^{\infty}\gamma^j r^{\pi}_{k+j+1} \\ &= \textstyle\sum_{j=0}^{\infty}\gamma^j \rho\big(x_{k+j+1},\pi(x_{k+j})\big) \end{aligned} \tag{2.8}$$

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

The RL methods can be classified into three main categories [44] as follows:

- Actor-only: The methods that directly search for an optimal control law.
- Critic-only: The methods that first learn an optimal value function. The control law is then computed based on the value function.
- Actor-critic: The method that search for an optimal control law (actor) explicitly. In addition, a critic learns the value function and evaluates the performance of the controller.

In this thesis, the actor-critic RL method is used for learning controllers. The actor-critic RL method is convenient for problems where both the critic (value function) and the actor (control policy) are approximated via basis function parameterizations [45]. The critic used in this thesis is approximated as $\hat{V}(x,\theta) = \theta^T\Psi_{\mathrm{c}}(x)$ with a parameter vector $\theta\in\mathbb{R}^{n_{\mathrm{c}}}$ and a user-defined basis function vector $\Psi_{\mathrm{c}}(x)\in\mathbb{R}^{n_{\mathrm{c}}}$. Similarly, the actor is approximated as $\hat{\pi}(x,\mu) = \mu^T\Psi_{\mathrm{a}}(x)$, where $\mu\in\mathbb{R}^{n_{\mathrm{a}}}$ is a parameter vector and $\Psi_{\mathrm{a}}(x)\in\mathbb{R}^{n_{\mathrm{a}}}$ is a user-defined basis function vector. The temporal difference (TD) [116] is defined as

$$\delta_{k+1} = r_{k+1} + \gamma\hat{V}(x_{k+1},\theta_k) - \hat{V}(x_k,\theta_k). \tag{2.9}$$

The critic parameters are updated using the gradient ascent rule

$$\theta_{k+1} = \theta_k + \alpha_{\mathrm{c}}\delta_{k+1}\nabla_\theta\hat{V}(x_k,\theta_k) \tag{2.10}$$

where $\alpha_{\mathrm{c}} > 0$ is the critic learning rate. In addition, the eligibility trace $e_k(x)$, which includes information on the visited states, can be used to speed up learning.

Consequently, the critic parameters are updated as

$$e_{k+1} = \gamma\lambda e_k(x) + \nabla_{\theta_k}\hat{V}(x_k, \theta_k) \tag{2.11}$$

$$\theta_{k+1} = \theta_k + \alpha_c\delta_{k+1}e_{k+1}(x) \tag{2.12}$$

where $\lambda \in [0, 1)$ is a trace decay 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 input as

$$u_k = \text{sat}\left(\hat{\pi}(x_k, \mu_k) + \Delta u_k\right) \tag{2.13}$$

where Δu_k is a zero-mean white Gaussian noise as an exploration term. Finally, the actor parameters are updated by

$$\mu_{k+1} = \mu_k + \alpha_a\delta_{k+1}\Delta u_k\nabla_{\mu_k}\hat{\pi}(x_k, \mu_k) \tag{2.14}$$

with $\alpha_a > 0$ the actor learning rate. Algorithm 1 summarizes the learning process in the actor-critic RL method designed for dynamical system (2.1), where n_t and n_s denote the number of trials and samples, respectively.

Algorithm 1 Actor-critic reinforcement learning

Require: λ, γ, α_a, α_c, n_t, n_s

$e_0 = 0$
Initialize θ_0, μ_0
for $w = 1$ **to** n_t **do**
  Initialize x_0
  for $k = 0$ **to** $n_s - 1$ **do**
    Execute: apply the control input (2.13) to system (2.1), observe the next state x_{k+1} and compute the reward $r_{k+1} = \rho(x_{k+1}, u_k)$
    Temporal Difference:
      $\delta_{k+1} = r_{k+1} + \gamma\theta^T\Psi_c(x_{k+1}) - \theta^T\Psi_c(x_k)$
    Critic Update:
      for $i = 1$ **to** n_c **do**
        $e_{i,k+1} = \gamma\lambda e_{i,k} + \nabla_{\theta_{i,k}}\theta^T\Psi_c(x_k)$
        $\theta_{i,k+1} = \theta_{i,k} + \alpha_c\delta_{k+1}e_{i,k+1}$
      end for
    Actor update:
      for $i = 1$ **to** n_a **do**
        $\mu_{i,k+1} = \mu_{i,k} + \alpha_a\delta_{k+1}\Delta u_k\nabla_{\mu_{i,k}}\hat{\pi}(x_k, \mu_k)$
      end for
  end for
end for

2.4 Passivity-Based Learning Control

The use of learning in the context of passivity-based control (PBC) techniques describes passivity-based learning controllers as discussed in [112]. Using the equations of motion along with the system's total energy, defined in these control techniques, one can estimate the DoAs of learning controllers. This section reviews the main concepts of passivity-based learning controllers.

Passivity-based controllers have been extensively used for regulation problems in port-Hamiltonian (PH) systems, see for example [128]. The standard input-state-output form of a time-invariant PH system is given by

$$\begin{aligned} \dot{x} &= \big(J(x) - R(x)\big)\nabla_x H(x) + g(x)u \\ y &= g^T(x)\nabla_x H(x) \end{aligned} \tag{2.15}$$

where $x \in \mathbb{R}^n$ is the state vector, $J(x) = -J^T(x)$ is a skew-symmetric interconnection matrix, $R(x) = R^T(x)$ is a symmetric dissipation matrix, and y is a collocated output with the input matrix $g(x)$. Moreover, $H(x)$ is the system Hamiltonian, which determines the sum of energy stored in all the individual elements of the system. For instance, in a mechanical system, the Hamiltonian is obtained by summing up the kinetic and potential energies.

In PBC, the control objective is obtained by making the closed-loop system passive with respect to a storage function, which has a minimum at the desired equilibrium [94]. The PBC techniques are broadly classified into three main categories. The first is stabilization by damping injection (DI), which is the simplest approach, but it has a limited application. The second is energy balancing and damping injection (EB-DI), which is the most frequently used method for set point regulation [81]. The third is interconnection and damping assignment passivity-based control (IDA-PBC), which can be utilized to solve various control problems for a wide range of physical systems such as mechanical and electromechanical systems [94].

To design a passivity-based controller for a PH system, one has to solve partial differential equations, which are computationally costly and sometimes inefficient. If one parameterizes the control input and apply the actor-critic RL method for learning the unknown parameter vectors, the complexity of control synthesis considerably decreases, because the problem of solving partial differential equations is eliminated. Two methods energy balancing actor-critic (EB-AC) [112] and algebraic interconnection and damping assignment actor-critic (A-IDA-AC) [82] are discussed, which have been implemented for various physical systems.

2.4.1 Energy-Balancing Actor-Critic

In PH systems, regulation problems are usually attained by the EB-DI algorithm. The EB-DI goal is to find a feedback control law such that the desired closed-loop Hamiltonian $H_{\mathrm{d}}(x)$ has a local minimum at the equilibrium x^*, that is

$$x^* = \arg\min H_{\mathrm{d}}(x). \tag{2.16}$$

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

$$\begin{aligned} u(x) &= u_{\mathrm{es}} + u_{\mathrm{di}} \\ &= \big(g^T(x)g(x)\big)^{-1} g^T(x)\big(J(x) - R(x)\big)\nabla_x H_{\mathrm{a}}(x) \\ &\quad - K(x)g^T(x)\nabla_x H_{\mathrm{d}}(x) \end{aligned} \tag{2.17}$$

where $K(x) = K^T(x)$ is a symmetric positive semi-definite damping injection matrix and $H_{\mathrm{a}}(x)$ is an added energy term that satisfies the energy balancing equation

$$H_{\mathrm{a}}(x) = H_{\mathrm{d}}(x) - H(x). \tag{2.18}$$

The supplied energy function $H_{\mathrm{a}}(x)$ is found by solving a set of partial differential equations, called matching condition, given by

$$\begin{bmatrix} g^{\perp}(x)\big(J(x) - R(x)\big) \\ g^T(x) \end{bmatrix} \nabla_x H_{\mathrm{a}}(x) = 0 \tag{2.19}$$

with $g^{\perp}(x) \in \mathbb{R}^{(n-m)\times n}$ the left annihilator matrix of the input matrix $g(x)$ (i.e., $g^{\perp}(x)g(x) = 0$). Consequently, a solution of (2.19) that can satisfy the equilibrium condition (2.16) is selected as $H_{\mathrm{a}}(x)$. For more details refer to [112].

To design an EB-AC controller, first the energy functions need to be parameterized. The approximated parameterized desired Hamiltonian of a physical system in the EB-AC method is given by

$$\hat{H}_{\mathrm{d}}(x, \xi) = H_{\mathrm{di}} + H_{\mathrm{es}} = H_{\mathrm{di}} + \xi^T \Psi_{\mathrm{es}}(x) \tag{2.20}$$

with H_{di} and H_{es} the damping injection and energy shaping terms of $\hat{H}_{\mathrm{d}}(x, \xi)$, where $\xi \in \mathbb{R}^{n_{\mathrm{es}}}$ is an unknown parameter vector and $\Psi_{\mathrm{es}}(x) \in \mathbb{R}^{n_{\mathrm{es}}}$ is a user-defined basis function vector. The "hat" symbol represents the approximated terms (i.e., $\hat{H}_{\mathrm{d}}$ is the approximated desired Hamiltonian). Substituting the energy functions (2.18) and (2.20) in (2.17), the control policy is computed with respect to

the parameter vector ξ and the basis function $\Psi_{\mathrm{es}}(x)$ as

$$\begin{aligned}\hat{\pi}(x,\xi) &= g^{\dagger}(x)\big(J(x)-R(x)\big)\big(\nabla_x \hat{H}_{\mathrm{d}}(x,\xi)-\nabla_x H(x)\big)\\ &\quad -K(x)g^T(x)\nabla_x \hat{H}_{\mathrm{d}}(x)\\ &= g^{\dagger}(x)F(x)\Big(\xi^T\nabla_x\Psi_{\mathrm{es}}(x)-\nabla_x H(x)\Big)\\ &\quad -K(x)g^T(x)\nabla_x \hat{H}_{\mathrm{d}}(x)\end{aligned} \tag{2.21}$$

where $g^{\dagger}(x)=\big(g^T(x)g(x)\big)^{-1}g^T(x)$ is the pseudo inverse of matrix $g(x)$ and $F(x)=J(x)-R(x)$ is the system matrix. The damping injection matrix $K(x)$ is also parameterized using an unknown parameter vector $\psi\in\mathbb{R}^{n_{\mathrm{di}}}$ and a user-defined basis function vector $\Psi_{\mathrm{di}}(x)\in\mathbb{R}^{n_{\mathrm{di}}}$ as

$$[\hat{K}(x,\psi)]_{ij}=\sum_{l=1}^{n_{\mathrm{di}}}[\psi]_{ijl}[\Psi_{\mathrm{di}}(x)]_l \tag{2.22}$$

such that $[\psi]_{ij}\in\mathbb{R}^{n_{\mathrm{di}}}$ satisfies the condition

$$[\psi]_{ij}=[\psi]_{ji}. \tag{2.23}$$

If this equality holds, the symmetry condition of $K(x)$ will be also satisfied. Substituting the approximated damping injection matrix $\hat{K}(x)$ into (2.21) yields the control policy

$$\begin{aligned}\hat{\pi}(x,\xi,\psi) &= g^{\dagger}(x)F(x)\Big(\xi^T\nabla_x\Psi_{\mathrm{es}}(x)-\nabla_x H(x)\Big)\\ &\quad -\psi^T\Psi_{\mathrm{di}}(x)g^T(x)\nabla_x\hat{H}_{\mathrm{d}}(x)\end{aligned} \tag{2.24}$$

where the unknown parameter vectors ξ and ψ are updated using the actor-critic method. Consequently, the saturated control input of the EB-AC method is computed at each time step by

$$u_k=\mathrm{sat}\left(\hat{\pi}(x_k,\xi_k,\psi_k)+\Delta u_k\right) \tag{2.25}$$

where Δu_k is a zero-mean Gaussian noise, as an exploration term. Algorithm 2 summarizes the synthesis of an EB-AC controller.

2.4.2 Algebraic Interconnection and Damping Assignment Actor-Critic

The IDA-PBC algorithm is a nonlinear state-feedback controller that can be used for stabilizing and tracking control problems [93]. In this method, first the system interconnection is changed to ensure the local stability of the desired state and

Algorithm 2 Energy-balancing actor-critic algorithm

Require: System (2.15), $\gamma, \alpha_\mathrm{a}, \alpha_\mathrm{c}, n_\mathrm{t}, n_\mathrm{s}$
Initialize θ_0, ξ_0, ψ_0
for $w = 1$ **to** n_t **do**
  Initialize x_0
  for $k = 0$ **to** $n_\mathrm{s} - 1$ **do**
    Execute: apply the control input (2.25) to system (2.15), observe the next state x_{k+1} and compute the reward $r_{k+1} = \rho(x_{k+1}, u_k)$
    Temporal Difference:
      $\delta_{k+1} = r_{k+1} + \gamma\theta^T\Psi_\mathrm{c}(x_{k+1}) - \theta^T\Psi_\mathrm{c}(x_k)$
    Critic Update:
      $\theta_{k+1} = \theta_k + \alpha_\mathrm{c}\delta_{k+1}\nabla_{\theta_k}\theta^T\Psi_\mathrm{c}(x_k)$
    Actor update:
      $\xi_{k+1} = \xi_k + \alpha_\mathrm{a}\delta_{k+1}\Delta u_k\nabla_{\xi_k}\hat{\pi}(x_k, \xi_k, \psi_k)$
      $\psi_{k+1} = \psi_k + \alpha_\mathrm{a}\delta_{k+1}\Delta u_k\nabla_{\psi_k}\hat{\pi}(x_k, \xi_k, \psi_k)$
  end for
end for

then by assigning an extra damping the global stability is obtained. Consider the input-affine form of system (2.1) described by

$$\dot{x} = f(x) + g(x)u. \tag{2.26}$$

The control law u is chosen such that the closed-loop system is of the form

$$\dot{x} = \big(J_\mathrm{d}(x) - R_\mathrm{d}(x)\big)\nabla_x H_\mathrm{d}(x) \tag{2.27}$$

where $J_\mathrm{d}(x) = -J_\mathrm{d}^T(x) \in \mathbb{R}^{n\times n}$ is the desired skew-symmetric interconnection matrix and $R_\mathrm{d}(x) = R_\mathrm{d}^T(x) \in \mathbb{R}^{n\times n}$ is the desired symmetric dissipation matrix, hence the desired system matrix $F_\mathrm{d}(x) \in \mathbb{R}^{n\times n}$ is given by $F_\mathrm{d}(x) = J_\mathrm{d}(x) - R_\mathrm{d}(x)$.

To obtain the closed-loop system in the form of (2.27), using the pseudo inverse of the input matrix $g(x)$ results in the control law

$$u(x) = g^\dagger(x)\left(F_\mathrm{d}(x)\nabla_x H_\mathrm{d}(x) - f(x)\right) \tag{2.28}$$

such that the unknown elements of $F_\mathrm{d}(x)$ and $H_\mathrm{d}(x)$ can be found by solving the matching condition

$$g^\perp(x)\big(F_\mathrm{d}(x)\nabla_x H_\mathrm{d}(x) - f(x)\big) = 0. \tag{2.29}$$

To solve this condition one needs to first fix $F_\mathrm{d}(x)$ or $H_\mathrm{d}(x)$ or both [82]. Depending on which element is fixed first, the control algorithm varies.

Algebraic IDA-PBC is a method in which the desired Hamiltonian $H_\mathrm{d}(x)$ in the matching condition is fixed [39]. This makes (2.29) an algebraic equation that is

applied to compute the unknown elements of $F_{\mathrm{d}}(x)$. Consider a generic lossless fully actuated mechanical system

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial q}(x) \\ \frac{\partial H}{\partial p}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u \tag{2.30}$$

where the state vector $x = [q^T \; p^T]^T$ consists of the generalized position $q \in \mathbb{R}^{\bar{n}}$ and generalized momentum $p \in \mathbb{R}^{\bar{n}}$ such that $2\bar{n} = n$ and n is the system dimension [82]. In the algebraic IDA-PBC method, one of the simplest choice for the desired Hamiltonian is the quadratic function. The local minimum condition (2.16) at the desired state $x_{\mathrm{d}} = [q_{\mathrm{d}}^T \; 0]^T$ can be satisfied by choosing

$$H_{\mathrm{d}}(x) = \frac{1}{2} p^T M^{-1}(q) p + \frac{1}{2}(q - q_{\mathrm{d}})^T \Lambda (q - q_{\mathrm{d}}) \tag{2.31}$$

where $M(q) \in \mathbb{R}^{\bar{n} \times \bar{n}}$ is a positive-definite mass-inertia matrix and $\Lambda \in \mathbb{R}^{\bar{n} \times \bar{n}}$ is a positive-definite scaling matrix. For a generic system matrix

$$F_{\mathrm{d}}(x) = \begin{bmatrix} F_{11}(x) & F_{12}(x) \\ F_{21}(x) & F_{22}(x) \end{bmatrix} \tag{2.32}$$

the control law is described by

$$u = F_{21}(x)\Lambda(q - q_{\mathrm{d}}) + F_{22}(x) M^{-1}(q) p + \frac{\partial H}{\partial q} \tag{2.33}$$

where the unknown elements F_{21} and F_{22} need to be chosen appropriately. Since the control law is described in terms of unknown elements, one can use a learning method to obtain these elements. Applying a linear-in-parameters function approximator, the unknown elements F_{21} and F_{22} are parameterized such that (2.33) results in the control policy

$$\hat{\pi}(x, \vartheta) = \vartheta_1^T \Psi_{\mathrm{al}}(x) \Lambda (q - q_{\mathrm{d}}) + \vartheta_2^T \Psi_{\mathrm{al}}(x) M^{-1}(q) p + \frac{\partial H}{\partial q} \tag{2.34}$$

where $\vartheta = [\vartheta_1^T \; \vartheta_2^T]^T$ is an unknown parameter vector and $\Psi_{\mathrm{al}}(x)$ is a user-defined matrix of Fourier basis functions. Since the unknown parameter vector ϑ is learned using the actor-critic RL, this control method is called A-IDA-AC.

The control policy in the A-IDA-AC algorithm is described by

$$\hat{\pi}(x, \vartheta) = g^{\dagger}(x)\big(\vartheta^T \Psi_{\mathrm{al}}(x) \nabla_x H_{\mathrm{d}}(x) - f(x)\big) \tag{2.35}$$

where the unknown parameter vector ϑ is updated using the actor-critic method. Consequently, the saturated control input of the A-IDA-AC method is computed

at each time step by

$$u_k = \text{sat}\left(\hat{\pi}(x_k, \vartheta_k) + \Delta u_k\right) \tag{2.36}$$

where Δu_k is a zero-mean Gaussian noise, as an exploration term. Algorithm 3 summarizes the synthesis of a A-IDA-AC controller.

Algorithm 3 Algebraic interconnection and damping assignment actor-critic

Require: System (2.15), $\gamma, \alpha_\text{a}, \alpha_\text{c}, n_\text{t}, n_\text{s}$
Initialize θ_0, ϑ_0
for $w = 1$ **to** n_t **do**
  Initialize x_0
  for $k = 0$ **to** $n_\text{s} - 1$ **do**
    Execute: apply the control input (2.36) to system (2.15), observe the next state x_{k+1} and compute the reward $r_{k+1} = \rho(x_{k+1}, u_k)$
    Temporal Difference:
      $\delta_{k+1} = r_{k+1} + \gamma\theta^T\Psi_\text{c}(x_{k+1}) - \theta^T\Psi_\text{c}(x_k)$
    Critic Update:
      $\theta_{k+1} = \theta_k + \alpha_\text{c}\delta_{k+1}\nabla_{\theta_k}\theta^T\Psi_\text{c}(x_k)$
    Actor update:
      $\vartheta_{k+1} = \vartheta_k + \alpha_\text{a}\delta_{k+1}\Delta u_k\nabla_{\vartheta_k}\hat{\pi}(x_k, \vartheta_k)$
  end for
end for

2.5 Conclusions

This chapter discussed sequential composition as an effective supervisory control approach. With regards to the other supervisory techniques, the induced control automaton in sequential composition is usually simple with sophisticated controllers. Although sequential composition generates a well-structured supervisory structure, it is designed for structured environments for which every condition has been already taken into account.

Then, a brief review is provided on the actor-critic RL method as well as two passivity-based learning control algorithms EB-AC and A-IDA-AC. The actor-critic RL method is beneficial for the problems with continuous state and action spaces, which is mostly the case for model-based controllers. The EB-AC algorithm is useful for regulation problems defined in a subclass of physical systems, such as fully actuated mechanical systems. The A-IDA-AC algorithm is more general and can be used for various control problems defined in a wide range of physical systems, such as regulation and tracking of multi-domain systems. Applying the control algorithms EB-AC and A-IDA-AC not only speeds up the learning process, but also provides the dynamic equations and total energy (Hamiltonian) of the system. These equations with system Hamiltonian are required tools for estimating the DoA of the controller.

3 CHAPTER

Estimating the Domain of Attraction

To design a supervisory controller in the context of sequential composition, the DoAs of the low-level controllers and their goal sets have to be known. In this thesis, a fast sampling method is proposed for estimating the DoAs of nonlinear systems. This procedure is computationally effective, compared with the existing optimization-based techniques, and is useful for real-time applications. The sampling approach proposed has been used to estimate the DoAs of stable equilibria in several nonlinear systems. Moreover, it has been applied to a passivity-based learning controller designed for a magnetic levitation system.

3.1 Introduction

The DoA of a stable equilibrium in a nonlinear system is a region of the state space from which each trajectory starts, eventually converges to the equilibrium itself. In the literature, the DoA is also known as the region of attraction or basin of attraction [126, 4]. The DoA of an equilibrium and its computation is of main importance in control applications. However, in most cases, the DoA has a irregular shape and its computation is quite costly. This chapter aims to approximate the DoAs of nonlinear systems in real-time by introducing a sampling approach.

Several techniques have been proposed in the literature to compute an inner approximation for the DoA [23], which can broadly be classified into Lyapunov-based and non-Lyapunov methods [40]. Lyapunov-based approaches include sum of squares (SOS) programming [19], methods that apply both simulation and SOS programming [125], procedures that use theory of moments [47], etc. In this approach first a candidate Lyapunov function is chosen to show asymptotic stability of the system in a small neighborhood of the equilibrium. Next, the largest

sublevel set of this Lyapunov function in which its time derivative is negative definite is computed as an estimate for the DoA [91]. Non-Lyapunov methods include trajectory reversing [40, 90], determining reachable sets of the system [7], and occupation measures [49, 72]. Figure 3.1 illustrates a broad classification of the existing techniques for estimating the DoA.

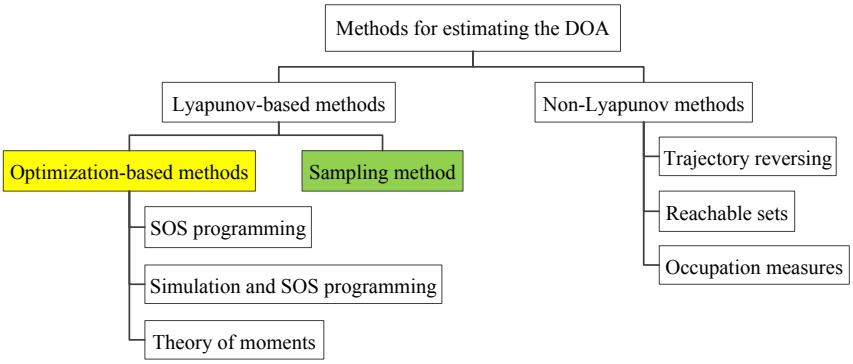

Figure 3.1: A broad classification of the existing techniques for estimating the DoA. This chapter proposes a sampling approach and makes a comparison with optimization-based methods.

Although Lyapunov-based techniques have been successfully implemented for estimating the DoAs of various nonlinear systems [23], there are still two main issues with using these approaches. The first is that most of the existing methods are limited to polynomial systems [46, 124]. In the case of non-polynomial systems, first the equations of motion are approximated by using the Taylor's expansion and then the DoA is computed based on the approximated polynomial equations. The second is that the available methods are usually computationally costly and time-consuming which makes them unsuitable for real-time applications [22].

This chapter proposes a fast sampling method for Lyapunov-based methods to estimate the DoAs of various nonlinear systems. This method is computationally effective and is beneficial for real-time applications. In this method, once a candidate Lyapunov function is chosen, a sampling algorithm searches for the largest sublevel set of the Lyapunov function such that its time derivative is negative definite throughout the obtained sublevel set. The proposed sampling method is applied to approximate the DoAs of several nonlinear systems, which have been already investigated in the literature, to validate its capability in comparison with the existing methods. This comparison goes beyond these examples and the sampling method is implemented to compute the DoAs of the passivity-based learning controllers [112] designed for a magnetic levitation system.

This chapter is organized as follows. Section 3.2 reviews the process of estimating the DoAs of nonlinear systems using Lyapunov-based techniques. Section 3.3 de-

scribes the sampling approach and provides a comparison between the estimated DoAs computed by the sampling method and by the existing optimization-based methods. Section 3.4 describes using DoA estimation for passivity-based learning controllers. Section 3.5 presents simulation results of the sampling method evaluated on a magnetic levitation system. Finally, Section 3.6 concludes the chapter after a short discussion on the capability of the proposed approach.

3.2 Lyapunov-Based Methods

Consider the closed-loop dynamical system (2.2). An analytical method to approximate the DoA is defined via Lyapunov stability theory as follows [57, 21].

Theorem 3.1 *A closed set $\mathcal{M} \subset \mathbb{R}^n$, including the origin as an equilibrium, can approximate the DoA for the origin of system (2.2) if:*

1. *$\mathcal{M}$ is an invariant set for system (2.2);*
2. *A positive definite function $L(x)$ can be found such that $\dot{L}(x)$ is negative definite within $\mathcal{M}$.*

For more details see [4]. If the equilibrium is non-zero, without loss of generality, the variable x can be replaced by $\bar{z} = x - \bar{x}^*$, where $\bar{x}^*$ is the non-zero equilibrium. As such, one can study the stability of the associated zero equilibrium [4]. The conditions of Theorem 3.1 ensure that the approximated set $\mathcal{M}$ is certainly contained in the DoA.

The choice of a candidate Lyapunov function is not a trivial task and the DoA approximation relies on the shape of the Lyapunov function's level sets. A procedure to find an appropriate Lyapunov function has been proposed in [18], where gradient search algorithms are implemented to compute a candidate Lyapunov function. Moreover, using composite polynomial Lyapunov functions [118] and rational Lyapunov functions instead of quadratic ones might lead to better approximations, since these have a richer representation power (see e.g., [129, 24]). Quadratic Lyapunov functions restrict the estimates to ellipsoids which are quite conservative [123]. A rational Lyapunov function is written in the form

$$L(x) = \frac{N(x)}{D(x)} = \frac{\sum_{i=2}^{\infty} R_i(x)}{1 + \sum_{i=1}^{n-2} Q_i(x)} \tag{3.1}$$

where $R_i(x)$ and $Q_i(x)$ are homogeneous polynomials of degree i, which are constructed by solving an optimization problem [129]. The sublevel set $\mathcal{L}(c)$ of the Lyapunov function $L(x)$ is defined by

$$\mathcal{L}(c) = \{x \in \mathcal{X} : L(x) \leq c\}. \tag{3.2}$$

According to Theorem 3.1, any sublevel set of a candidate Lyapunov function that satisfies the locally asymptotic stability of the equilibrium can be an estimate for the DoA if the time derivative of the Lyapunov function is negative everywhere within the sublevel set. Since the largest sublevel set provides a more accurate estimate, the problem of approximating the DoA is converted to the problem of finding the largest sublevel set of a given Lyapunov function [52]. To attain the largest estimate for the DoA, one needs to find the maximum value $c \in \mathbb{R}$ for $\mathcal{L}(c)$ such that the computed set satisfies the conditions of Theorem 3.1.

Theorem 3.2 *[23] The invariant set $\mathcal{L}(c_*)$, which is a sublevel set of the Lyapunov function $L(x)$, is the largest estimate of the DoA for the origin of system (2.2) if*

$$\begin{cases} c_* = \max\ c \\ \quad s.t.\ \mathcal{L}(c) \subseteq \mathcal{H}(x) \\ \quad\quad \mathcal{H}(x) = \{0\} \cup \{x \in \mathbb{R}^n : \dot{L}(x) < 0\}. \end{cases} \tag{3.3}$$

This can be approached as an optimization problem that has been solved by using SOS programming, methods that apply both simulation and SOS programming, and methods that use theory of moments. Although estimating the DoAs of nonlinear systems using SOS programming has been widely studied in literature (see e.g., [19, 49]), it is restricted to systems and Lyapunov functions described by polynomial equations. In the case of non-polynomial systems, the equations are approximated by polynomial terms using Taylor's expansion and so the DoA is estimated based on the polynomial equations [22]. This chapter presents an alternative approach using the sampling approach.

3.3 Sampling Method

The sampling approach presented in this chapter has the same goal as the Lyapunov based optimization approaches have: find the largest sublevel set of a candidate Lyapunov function to approximate the DoA. The conditions stated in Theorem 3.1 are explicitly evaluated for a given Lyapunov function with respect to a randomly chosen state x_i. The level sets associated with the sample x_i with positive derivative of the Lyapunov function are discarded. Two various sampling methods are proposed: memoryless and with a memory, designed to achieve tighter estimates.

3.3.1 Memoryless Sampling

This method searches for the upper bound of the parameter c_* in (3.3). First, a state x_i is randomly chosen within $\mathcal{X}$ or its subset and the conditions of Theorem 3.1

are checked for $L(x_i)$ and $\dot{L}(x_i)$. If these conditions are not satisfied, the upper bound of c_* is decreased to the value of $\hat{c}_* = L(x_i)$ and the sublevel set $\mathcal{L}(\hat{c}_*)$ is computed as an overestimation for the DoA. At the beginning of the algorithm, $\hat{c}_*$ is initialized at $\hat{c}_* = \infty$. As the sampling proceeds for a large number of samples (n_s) throughout the state space, the value of $\hat{c}_*$ converges to c_* from above and the obtained largest sublevel set $\mathcal{L}(\hat{c}_*)$ will be very close to $\mathcal{L}(c_*)$. Since this procedure just focuses on the upper bound of c_*, the achieved estimates are not tight enough and the condition of $\dot{L}(x) < 0$ may not be satisfied for some regions of the attained sublevel set as the computed value $\hat{c}_*$ is actually larger than the real value c_*. Nevertheless, this technique is very fast and its result is very close to the reported estimates in the literature for various classes of systems. Moreover, it does not require computer memory to save the computed results since once a new value is computed for $\hat{c}_*$, its current value is replaced by the new value. Algorithm 4 summarizes this method for estimating the DoA of a given stable equilibrium.

Algorithm 4 Memoryless sampling method for estimating the DoA

Require: $L(x)$, $\dot{L}(x)$, n_s
1: Initialize $\hat{c}_* = \infty$
2: **for** $i = 1$ **to** n_s **do**
3: Pick a random state x_i within the state space
4: **if** $\dot{L}(x_i) \geq 0$ **and** $L(x_i) < \hat{c}_*$ **then**
5: $\hat{c}_* = L(x_i)$
6: **end if**
7: **end for**
8: **return** $\hat{c}_*$

As an example, consider a pendulum described by the following nonlinear dynamic equations

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\sin(x_1) - 0.5x_2 \end{cases} \tag{3.4}$$

where x_1 is the angle of the pendulum measured from the vertical axis and x_2 is the angular velocity. The state vector is defined by $x = [x_1\ x_2]^T$. The sampling method is used with a uniform distribution to approximate the DoA of the stable equilibrium $x = (0, 0)$. To compute a candidate Lyapunov function, first the dynamic equations (3.4) are linearized around the equilibrium and then the candidate Lyapunov function is computed in the form $L(x) = x^T P x$, where P is the solution of the Lyapunov equation $A^T P + PA + Q = 0$ with the identity matrix Q. In this example, the candidate Lyapunov function is obtained as

$$L(x) = 2.25x_1^2 + x_1x_2 + 2x_2^2. \tag{3.5}$$

Figure 3.2 illustrates the evolution of $\hat{c}_*$ of the sampling approach with $n_\mathrm{s} = 500$ samples. The real value c_* for the candidate Lyapunov function (3.5), calculated

by solving the optimization problem (3.3), is $c_* = 9.287$ and the value computed by the proposed method is $\hat{c}_* = 9.702$.

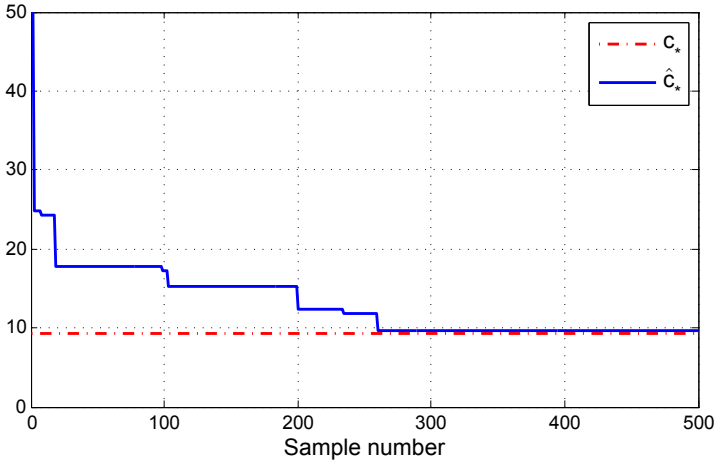

Figure 3.2: The evolution of $\hat{c}_*$ using the memoryless sampling method for the pendulum example.

3.3.2 Sampling with Memory

This method updates both the lower and the upper bounds of c_* denoted $\underline{c}_*$ and $\bar{c}_*$, respectively. Together, these bounds yield a more accurate estimate for the DoA. At the beginning of the algorithm, the lower bound of c_* is set to $\underline{c}_* = 0$ and its upper bound to $\bar{c}_* = \infty$. If for a randomly chosen state x_i we have $\dot{L}(x_i) < 0$ and $\underline{c}_* < L(x_i) < \bar{c}_*$, then the value of $\underline{c}_*$ is replaced by the value of its associated Lyapunov function, that is $\underline{c}_* = L(x_i)$. Otherwise, if $\dot{L}(x_i) \geq 0$ and $L(x_i) < \bar{c}_*$, then the value of $\bar{c}_*$ is replaced by $L(x_i)$. As the sampling proceeds, after a large number of samples, the value of $\underline{c}_*$ increases, but not necessarily monotonically. Eventually it converges to c_* and the largest sublevel set $\mathcal{L}(\underline{c}_*)$ is obtained. Moreover, the value of $\bar{c}_*$ monotonically decreases and converges to c_* from above.

When the conditions of Theorem 3.1 are satisfied for state x_i, the value of $L(x_i)$ is stored in an array as a possible estimate for c_*. This is required to guarantee that the approximated DoAs computed by the lower bound of c_* always verify the conditions of Theorem 3.1. This leads to tighter estimates. The array, denoted $\bar{\mathcal{E}}$, contains 0 initially. The length of this array, without counting its initial element, is in the worst case $n_\mathrm{s} - 1$. When $\dot{L}(x_i) < 0$ and $L(x_i) < \bar{c}_*$, the value of $L(x_i)$ is stored in an array $\bar{\mathcal{E}}$ as $\mathcal{L}\left(L(x_i)\right)$ is a potential estimate for the DoA. In the case $\dot{L}(x_i) \geq 0$ and $L(x_i) < \bar{c}_*$, if $\underline{c}_* \geq \bar{c}_*$ then the algorithm looks for a new lower bound $\underline{c}_*$ among the values stored in the array $\bar{\mathcal{E}}$. The maximum value of $\underline{c}_*$ is chosen from $\bar{\mathcal{E}}$ such that $\underline{c}_* < \bar{c}_*$. Selecting a previously stored lower bound

satisfies the condition $\dot{L} < 0$ for the obtained sublevel set $\mathcal{L}(\underline{c}_*)$. In the worst case scenario $\underline{c}_* = 0$. Algorithm 5 describes the sampling method with memory for estimating the DoA.

Algorithm 5 Sampling method with memory for estimating the DoA

Require: $L(x)$, $\dot{L}(x)$, n_s
Initialize $\underline{c}_* = 0$, $\bar{c}_* = \infty, \bar{\mathcal{E}} = \{0\}$
for $i = 1$ **to** n_s **do**
  Pick a random state x_i within the state space
  if $\dot{L}(x_i) < 0$ **and** $L(x_i) < \bar{c}_*$ **then**
    store $L(x_i)$ in $\bar{\mathcal{E}}$
    if $L(x_i) > \underline{c}_*$ **then**
      $\underline{c}_* = L(x_i)$
    end if
  else if $\dot{L}(x_i) \geq 0$ **and** $L(x_i) < \bar{c}_*$ **then**
    $\bar{c}_* = L(x_i)$
    if $\underline{c}_* \geq \bar{c}_*$ **then**
      $\underline{c}_* = \arg\max\{c \in \bar{\mathcal{E}} : c < \bar{c}_*\}$
    end if
  end if
end for
return $\underline{c}_*$

This approach is applied with a uniform distribution sampling to approximate the DoA for the equilibrium of the pendulum example. Figure 3.3 illustrates the values of the lower and upper bounds of c_* throughout the sampling process with 500 samples where $\underline{c}_* = 9.174$. Figure 3.4 depicts the approximated DoA of the equilibrium. The black ellipsoid represents the DoA estimate with $\underline{c}_* = 9.271$, the dashed blue line, which determines the boundary of the light blue area, represents the region in which $\dot{L}(x) < 0$, and the arrows represent the system trajectories. If the trajectories start inside the DoA estimate, they certainly converge to the origin. The randomly chosen sampling states, which are 500 samples in this example, are represented by red points throughout the state space.

3.3.3 Repeatability

To check the repeatability of the proposed sampling approach, the process of estimating the DoA for the equilibrium of the pendulum example is run various instances. Figure 3.5 illustrates the mean value of $\underline{c}_*$ and $\bar{c}_*$ (i.e., $(\underline{c}_* + \bar{c}_*)/2$) and its standard deviation by a black line and green bars, minimum of $\underline{c}_*$ and maximum of $\bar{c}_*$ by blue dashed lines at each sample in a simulation where the sampling method runs 1000 iterations each with 500 samples. The real value of $c_* = 9.287$ is represented by a dotted red line. While sampling proceeds, the mean, minimum

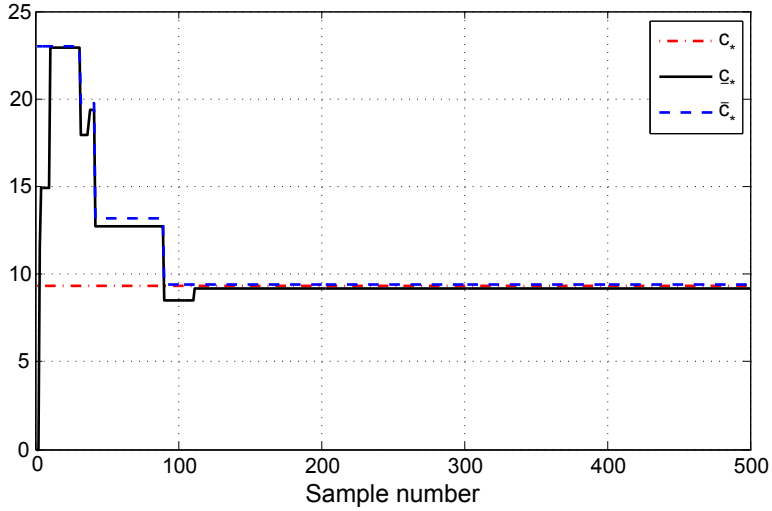

Figure 3.3: The evolution of $\underline{c}_*$ and $\bar{c}_*$ using the sampling method with memory for the pendulum example.

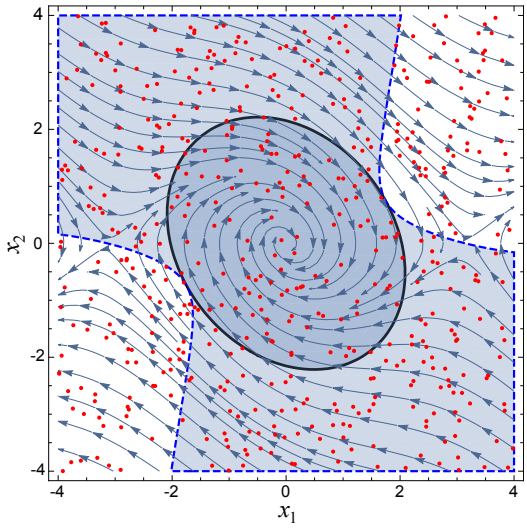

Figure 3.4: Approximated DoA for the pendulum example using a uniform distribution for sampling. The black ellipsoid represents the DoA estimate, the dashed blue line (boundary of the light blue area) represents the region in which $\dot{L}(x) < 0$, the arrows represent the system trajectories, and the red points represent the randomly chosen sampling states.

and maximum values converge to the real value of c_* and the value of the standard deviation decreases. These results validates the repeatability of the proposed sampling techniques for this particular model.

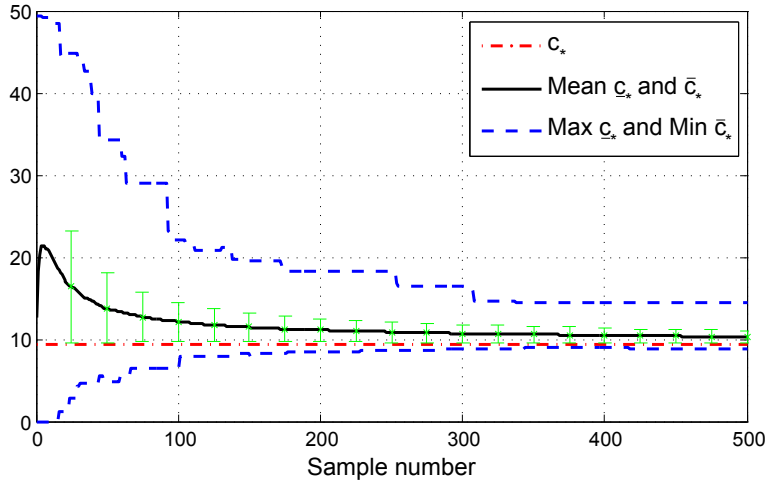

Figure 3.5: The evolution of mean value of $\underline{c}_*$ and $\bar{c}_*$ and its standard deviation, minimum value of $\underline{c}_*$, and maximum value of $\bar{c}_*$ for the sampling method in the pendulum example. The real value of c_* is represented by the dotted red line.

3.3.4 Directed Sampling

In the pendulum example, a uniform distribution is used for sampling the state space or its subset. However, if the structure of the level sets of the Lyapunov function are known, other distributions can be used to avoid sampling in areas of the state space which are already known not belong to the DoA. It is desirable to sample inside the largest level set found so far, specially in its boundary.

In general sampling with an arbitrary distribution is a challenging problem. Two main approaches exist in the literature: rejection sampling and inverse transform sampling [11], which focus on sampling the relevant locations of the state space at the cost of computational complexity. While evaluating a particular sample is costly (due to a complicated Lyapunov function or system dynamics), the extra cost incurred by sampling from a complex distribution may be negligible.

To test the trade-off between the speed of convergence and the computational cost, three different sampling approaches are applied to the pendulum example (3.4). The uniform sampling on a fixed box (a subset of the state space) is compared with uniform sampling mapped through polar coordinates to lie inside the largest

found valid level set and with exponential sampling mapped through polar coordinates to lie around the boundary of the largest found valid level set. Figure 3.6 illustrates the sampling points selected by the three types of distributions. The obtained data corroborate the hypothesis that different sampling leads to different convergence rates. Figure 3.7 illustrates the convergence statistics for 1000 iterations with 500 samples each. The exponential polar sampling converges the fastest and has the lowest variation between $\underline{c}_*$ and $\bar{c}_*$ while converging. This can be explained by observing in Figure 3.6(c) that most of the samples are focused around the boundary of the level set. For this particular example, the cost of evaluating the Lyapunov function and its time derivative is low, but the computation time increases with the complexity of the sampling algorithm. Table 3.1 shows the average computation time of each sampling method with 500 samples, implemented in the Mathematica software on an Intel core i7 2.7 GHz microprocessor.

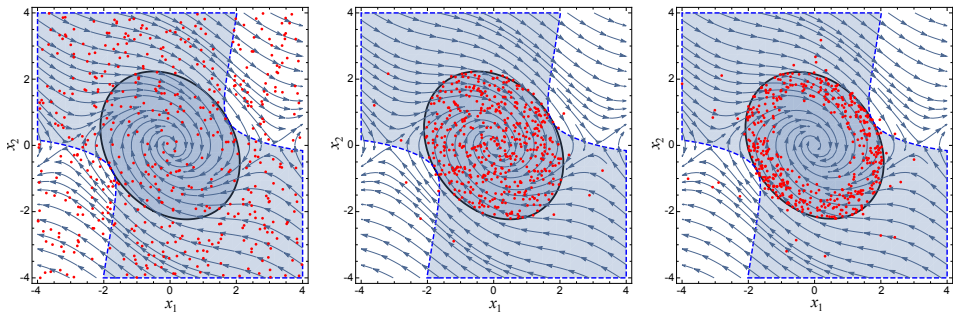

Figure 3.6: Approximated DoAs for the pendulum example using a (a) uniform, (b) polar uniform, and (c) polar exponential distribution for sampling. In the plots, the black ellipsoid represents the DoA estimate, the dashed blue line represents the region in which $\dot{L}(x) < 0$, the arrows represent the system trajectories, and the red points represent the randomly chosen sampling states.

Table 3.1: Computation time statistics of the sampling methods with various distributions for estimating the DoA of the pendulum example

Sampling method	Time [ms]
Uniform in a box	7.4
Uniform in polar coordinates	17.1
Exponential in polar coordinates	27.4

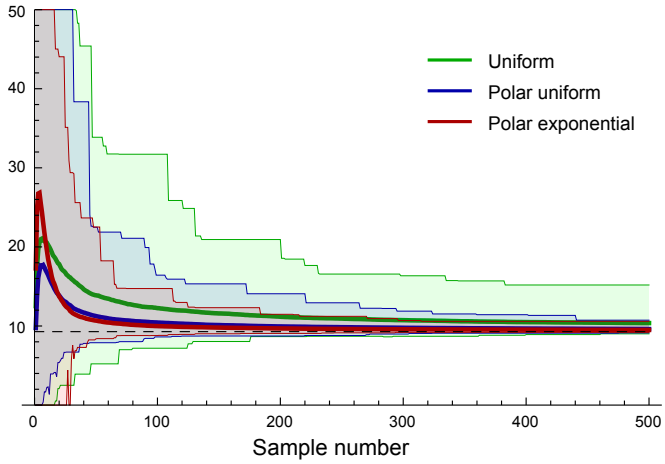

Figure 3.7: The evolution of the mean value of $\underline{c}_*$ and $\bar{c}_*$, minimum value of $\underline{c}_*$, and maximum value of $\bar{c}_*$ for the sampling technique implemented for the pendulum example with a uniform, polar uniform, and polar exponential distribution. The sampling method runs 1000 iterations each with 500 samples. The real value of c_* is represented by a dashed black line.

3.3.5 Sampling vs. Optimization-Based Methods

Both the sampling and optimization-based methods require a candidate Lyapunov function for estimating the DoA. Table 3.2 represents six dynamical systems with quadratic Lyapunov functions selected from the literature. The dynamic equations of the first three examples are polynomial and the equations of the last three are non-polynomial. Examples E3 and E6 are third-order systems and the others are second-order systems. For each system, the maximum possible value of c_* computed by the sampling approach with 1000 samples is compared with the result of optimization-based methods, reported in the literature. The estimates attained by the sampling technique are very close to the estimates derived by optimization-based methods. In some cases, such as example E2, the result of the sampling procedure is even more accurate. The last column of Table 3.3 presents the simulation time for approximating the DoA of each system using the sampling approach, implemented in the Matlab R2014a software on an Intel core i7 2.7 GHz microprocessor.

Similarly, Table 3.4 illustrates three dynamical systems with rational Lyapunov functions selected from the literature. Example E7 is a second-order polynomial system, E8 is a second-order non-polynomial system and E9 is a third-order polynomial system. Table 3.5 presents their corresponding rational Lyapunov functions based on (3.1). The maximum possible value of c_* obtained by the sampling approach with 1000 samples is compared with the result of optimization-based

Table 3.2: Dynamical systems with quadratic Lyapunov functions

Example	Systems dynamics	Lyapunov function
E1 [123, 46]	$\dot{x}_1 = -2x_1 + x_1x_2$ $\dot{x}_2 = -x_2 + x_1x_2$	$x_1^2 + x_2^2$
E2 [129, 46]	$\dot{x}_1 = -x_2$ $\dot{x}_2 = x_1 - x_2 + x_1^2x_2$	$1.5x_1^2 - x_1x_2 + x_2^2$
E3[124, 47]	$\dot{x}_1 = -x_1 + x_2x_3^2$ $\dot{x}_2 = -x_2 + x_1x_2$ $\dot{x}_3 = -x_3$	$x_1^2 + x_2^2 + x_3^2$
E4 [20, 108]	$\dot{x}_1 = -\frac{1}{4}x_1 + \ln(1 + x_2)$ $\dot{x}_2 = -\frac{3}{8}x_1 - \frac{1}{5}x_1x_2 + (\frac{1}{8}x_1 - x_2)\cos x_1$	$x_1^2 + x_2^2$
E5 [22]	$\dot{x}_1 = x_2$ $\dot{x}_2 = -0.2x_2 + 0.81\sin x_1\cos x_1 - \sin x_1$	$x_1^2 + x_1x_2 + 4x_2^2$
E6 [20]	$\dot{x}_1 = 1 + x_3 + \frac{1}{8}x_3^2 - \exp(x_1)$ $\dot{x}_2 = -x_2 - x_3$ $\dot{x}_3 = -x_2 - 2x_3 - \frac{1}{2}x_1^2$	$x_1^2 + x_2^2 + x_3^2$

methods, reported in the literature. The result of this comparison validates the proposed sampling technique particularly for non-polynomial systems. The simulation time for approximating the DoA of each system using the sampling procedure is given in the last column of Table 3.4, which are considerable smaller than the rarely reported simulation time for the optimization-based methods in the literature. Figure 3.8 depicts the approximated DoAs obtained by the sampling method for the origins of examples E1–E9. According to the results obtained, the proposed sampling approach is suitable for estimating the DoAs of both polynomial and non-polynomial systems. It is computationally effective and computes the DoA estimate considerably fast. Although the sampling method may offer less accurate estimates for the DoA at times, it is very useful for real-time applications.

Table 3.3: Simulation results of the sampling method for the systems of Table 3.2

Example	Optimization (c_*)	Sampling ($\underline{c}_*$)	Time [ms]
E1	4.0804	4.112	6.6
E2	2.09	2.318	6.7
E3	4.9188	4.971	8.4
E4	0.2737	0.278	8.3
E5	0.6990	0.708	7.2
E6	2.655	2.887	8.6

Table 3.4: Dynamical systems with rational Lyapunov functions

Example	Systems dynamics	Opt. (c_*)	Sampling ($\underline{c}_*$)	Time [ms]
E7 [97, 74]	$\dot{x}_1 = -x_2$ $\dot{x}_2 = x_1 - x_2 + x_1^2 x_2$	5.3133	5.131	14.0
E8 [22, 74]	$\dot{x}_1 = -x_1 + x_2$ $+0.5(\exp(x_1) - 1)$ $\dot{x}_2 = -x_1 - x_2 + x_1 x_2$ $+x_1 \cos x_1$	1.2251	1.218	14.6
E9 [47, 74]	$\dot{x}_1 = -x_1 + x_2 x_3^2$ $\dot{x}_2 = -x_2 + x_1 x_2$ $\dot{x}_3 = -x_3$	1.320	1.318	16.6

Table 3.5: Rational Lyapunov functions for the systems of Table 3.4

Example	Numerator and denominator terms of the Lyapunov function (3.1)
E7	$R_2(x) = 1.5x_1^2 - x_1x_2 + x_2^2$ $R_3(x) = 0$ $R_4(x) = -0.3186x_1^4 + 0.7124x_1^3x_2 - 0.1459x_1^2x_2^2 + 0.1409x_1x_2^3$ $-0.03769x_2^4$ $Q_1(x) = 0$ $Q_2(x) = -0.2362x_1^2 + 0.31747x_1x_2 - 0.1091x_2^2$
E8	$R_2(x) = x_1^2 + 1.3333x_1x_2 + 1.1667x_2^2$ $R_3(x) = -0.2272x_1^3 - 0.1396x_1^2x_2 + 0.3785x_1x_2^2 - 0.1798x_2^3$ $R_4(x) = 0.0136x_1^4 - 0.2864x_1^3x_2 + 0.1918x_1^2x_2^2 - 0.0530x_1x_2^3$ $+0.0172x_2^4$ $Q_1(x) = -0.5605x_1 - 0.7255x_2$ $Q_2(x) = 0.3254x_1^2 + 0.0910x_1x_2 + 0.1015x_2^2$
E9	$R_2(x) = 0.5x_1^2 + 0.5x_2^2 + 0.5x_3^2$ $R_3(x) = -0.0739x_1^3 + 0.2594x_1x_2^2 - 0.0739x_1x_3^2$ $R_4(x) = -0.0301x_1^4 + 0.0573x_1^2x_2^2 - 0.0301x_1^2x_3^2 + 0.2501x_1x_2x_3^2$ $-0.03x_2^4 - 0.03x_2^2x_3^2$ $Q_1(x) = -0.1478x_1$ $Q_2(x) = -0.0602x_1^2 - 0.06x_2^2$

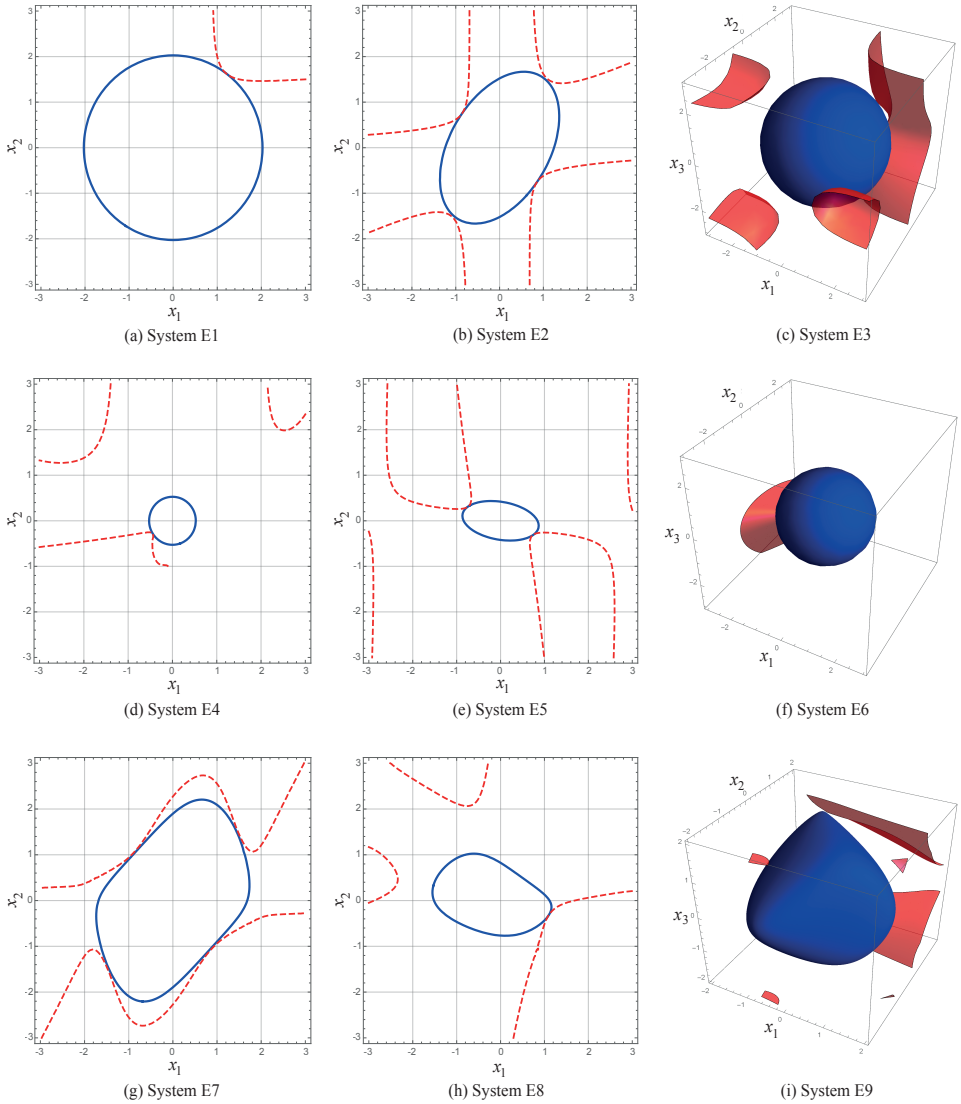

Figure 3.8: Approximated DoAs for the origins of examples E1–E9 described in Table 3.2 and Table 3.4 using the sampling method.

3.4 Passivity-Based Learning Control with Domain of Attraction Estimation

One of the main advantages of DoA approximation is to speed up the process of control design, specifically for passivity-based learning controllers. Consider the control input of the A-IDA-AC algorithm, described by (2.36), as a passivity-based learning controller. The DoA of the learned controller is approximated after each learning trial via the sampling method. Once the approximated DoA covers the desired regions of the state space, learning can be terminated. As such, monitoring the DoAs of the learned controllers provides a stopping criterion for the learning process. Algorithm 6 summarizes the procedure of the control design with DoA estimation. In this algorithm, the loop counter w counts the number of learning trials after which the DoA of the controller is sufficiently large to cover the initial state, 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. Figure 3.9 illustrates a block diagram representation of the A-IDA-AC algorithm together with DoA estimation.

Algorithm 6 Algebraic interconnection and damping assignment actor-critic algorithm with DoA estimation

Require: system (2.26), $x_0, \lambda, \gamma, \alpha_{\mathrm{a}}, \alpha_{\mathrm{c}}, n_{\mathrm{a}}, n_{\mathrm{c}}, n_{\mathrm{s}}, n_{\mathrm{t}}$

$w \leftarrow 0$
$e_0 = 0$
Initialize θ_0, ϑ_0
repeat
  $w \leftarrow w + 1$
  Initialize x_0
  for $k = 1$ **to** n_{s} **do**
    Algebraic interconnection and damping assignment actor-critic:
      $u_k = \mathrm{sat}\left(\hat{\pi}(x_k, \vartheta_k) + \Delta u_k\right)$
      Apply actor-critic RL to update ϑ_k based on Algorithm 3
  end for
  A-IDA-AC controller Φ
  Estimate the DoA $\mathcal{D}(\Phi)$ based on Algorithm 5
until $x_0 \in \mathcal{D}(\Phi)$ **or** $w = n_{\mathrm{t}}$

3.5 Simulation Results: Magnetic Levitation System

The performance of the proposed sampling method is evaluated for estimating the DoA of the A-IDA-AC controller designed for a magnetic levitation system to stabilize an steel ball at a desired position. The magnetic-levitation system [48],

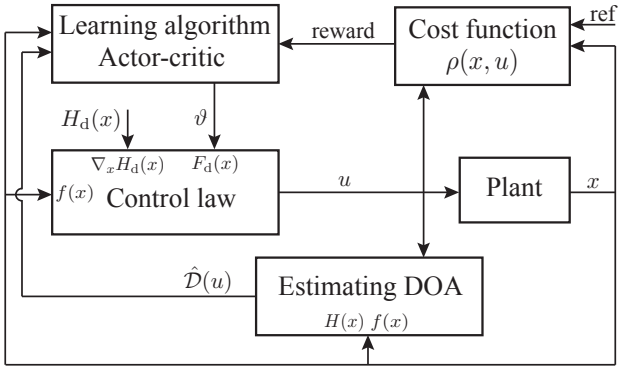

Figure 3.9: Block diagram representation of the A-IDA-AC algorithm with DoA estimation.

represented in Figure 3.10, is modeled by the dynamic equations

$$M\ddot{q} = Mg - \frac{e^2 C_1}{2\big(C_1 + I_0(C_2+q)\big)^2} \tag{3.6}$$

$$\dot{e} = -R\frac{e(C_2+q)}{C_1 + I_0(C_2+q)} + u$$

where q is the steel ball vertical position and $e = I(q)i$ is the magnetic flux with i the current through the coil and $I(q)$ the varying-inductance given by

$$I(q) = \frac{C_1}{C_2+q} + I_0. \tag{3.7}$$

The saturated control input u is the voltage across the coil. The state vector is defined by $x = [q\ p\ e]^T$, where $p = M\dot{q}$ is the momentum. Table 3.6 illustrates the values of the model parameters for the magnetic levitation system.

Table 3.6: Model parameters of the magnetic levitation system

Model parameter	Symbol	Value	Unit
Mass of steel ball	M	0.8	kg
Electrical resistance	R	11.68	Ω
Coil parameter 1	C_1	1.6×10^{-3}	Hm
Coil parameter 2	C_2	7×10^{-3}	m
Nominal inductance	I_0	0.8052	H
Gravity	g	9.81	$\mathrm{m{\cdot}s^{-2}}$

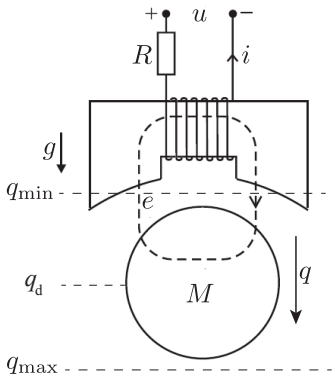

Figure 3.10: Schematic representation of the magnetic levitation system, adopted from [128].

The closed-loop system with respect to the feed-back control input is described by

$$\begin{bmatrix} \dot{q} \\ \dot{p} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -F_{22}(x) & F_{23}(x) \\ 0 & -F_{23}(x) & -R \end{bmatrix} \begin{bmatrix} \nabla_q H_{\mathrm{d}}(x) \\ \nabla_p H_{\mathrm{d}}(x) \\ \nabla_e H_{\mathrm{d}}(x) \end{bmatrix}. \tag{3.8}$$

The desired Hamiltonian $H_{\mathrm{d}}(x)$ that satisfies the equilibrium condition (2.16) at the desired state $x_{\mathrm{d}} = (q_{\mathrm{d}}, p, e_{\mathrm{d}}) = (0.065, 0, 1.2)$ is chosen in the quadratic form

$$H_{\mathrm{d}}(x) = \frac{1}{2}\gamma_{\mathrm{q}}(q - q_{\mathrm{d}})^2 + \frac{p^2}{2M} + \frac{1}{2I_0}(e - e_{\mathrm{d}})^2 \tag{3.9}$$

where γ_{q} is a unit conversion factor with the value of one. The desired magnetic flux e_{d} is described by

$$e_{\mathrm{d}} = \sqrt{2Mg/C_1}\big(C_1 + I_0(C_2 + q_{\mathrm{d}})\big). \tag{3.10}$$

Substituting (3.6)–(3.10) in (2.28) and applying Fourier basis functions to approximate F_{23} as $F_{23}(x, \vartheta) = \vartheta^T \phi(x)$, the parameterized control policy is given by

$$\hat{\pi}(x, \vartheta) = -\vartheta^T \phi(x)\frac{p}{M} - R\frac{(e - e_{\mathrm{d}})}{I_0} + R\frac{e(C_2 + q)}{\big(C_1 + I_0(C_2 + q)\big)} \tag{3.11}$$

which is then substituted in (2.36) to compute the saturated control input at each time step. The parameter vector ϑ is learned using the actor-critic RL method as described in Algorithm 6. The learning parameters and state limitations (due to the physical constraints) is given in Table 3.7. Moreover, Figure 3.11 shows the sum of rewards that a learning A-IDA-AC controller receives per trial through the learning simulation with 60 trials.

Table 3.7: Learning parameters and state limitations of the magnetic levitation system

Parameter	Symbol	Value	Unit
Sample time	T_s	0.004	s
Trial time	T_t	2	s
Number of trials	–	60	–
Decay rate	γ	0.95	–
Eligibility trace	λ	0.65	–
Exploration variance	σ^2	1	–
Learning rate of critic	α_c	0.01	–
Learning rate of $F_{21}(x)$	$\alpha_{\mathrm{a}\vartheta}$	1×10^{-7}	–
Max control input	u_max	60	V
Max position	q_max	13×10^{-3}	m
Max momentum	p_max	3×10^{-1}	kg·m·s^{-1}
Max magnetic flux	e_max	3	Wb

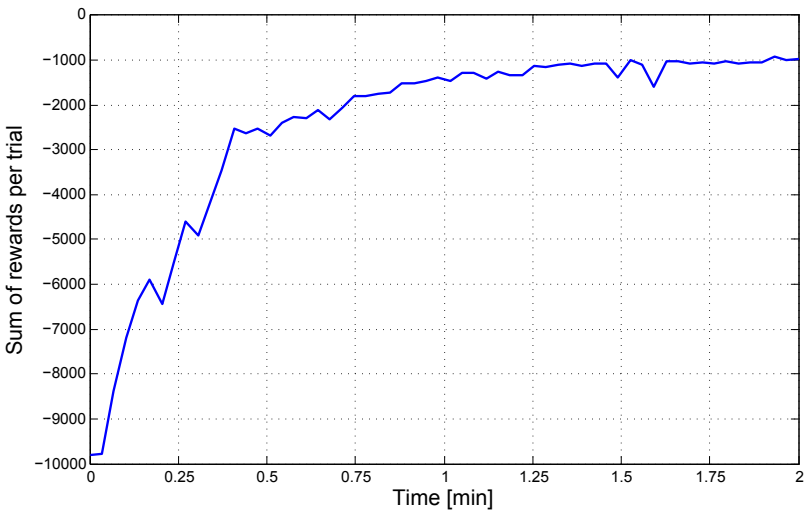

Figure 3.11: Sum of rewards that a learning controller receives per trial over a learning simulation with 60 trialsfor the magnetic levitation system.

The sampling method is implemented to approximate the DoA of the learned controller after each learning trial. The system Hamiltonian is deployed as a candidate Lyapunov function to estimate the DoA. Figure 3.12 presents the DoAs approximated for the learned controllers at four specific trials, where the trial numbers are also illustrated. While learning is in progress, the DoA typically enlarges centered at the desired state $x_{\mathrm{d}} = (0.065, 0, 1.2)$. In this example, after 24 trials the controller's DoA becomes sufficiently large to include the initial ball position. Figure 3.13 illustrates an evaluation of the learned controller in simulation. As shown in this figure, the steel ball stays at the initial position for 0.06 seconds while the control input is not zero. This time is necessary for magnetizing the coil.

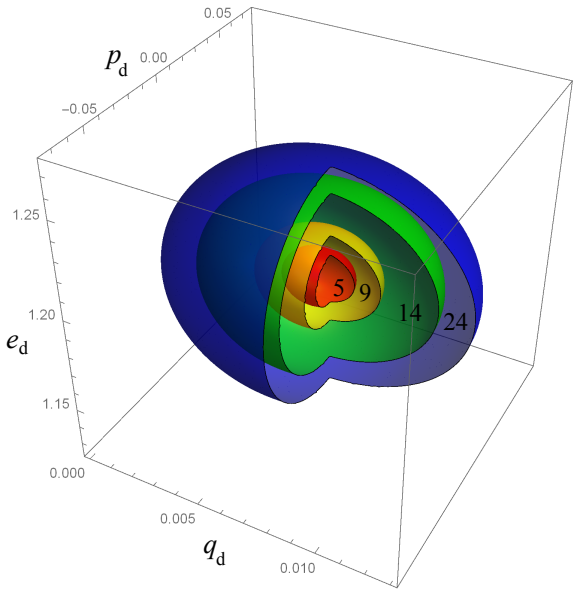

Figure 3.12: Approximated DoAs of the learned controllers at four specific trials for the magnetic levitation system, with trial numbers.

3.6 Conclusions

This chapter has proposed a fast sampling approach for estimating the DoAs of nonlinear systems in real-time. The approximated DoAs computed by this technique have been compared with the estimates derived by optimization based methods. It is concluded that the sampling approach is fast and computationally effective in comparison with optimization-based methods and it can be used for real-time applications. Although a formal guarantee for convergence does not exist yet, the empirical evidence arising from extensive simulations suggests that in practice this approach always converges to the exact level set for a sufficiently

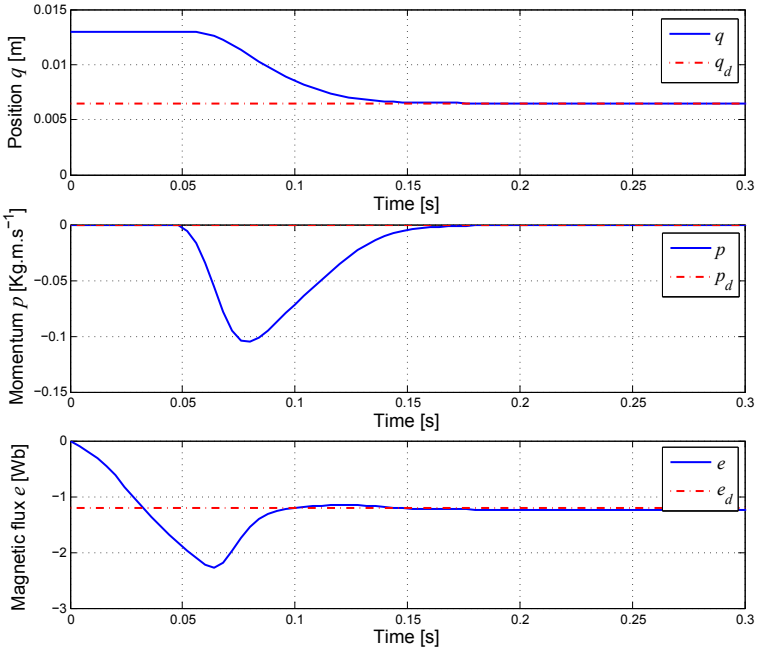

Figure 3.13: Simulation results of the learned controller for the magnetic levitation system.

large number of samples. Moreover, the rate of convergence depends on the distribution function selected for sampling. Using a more sophisticated distributed function can speed up convergence of the sampling procedure. As such, there is a trade-off between the speed of convergence and the computational cost imposed by the complexity of the sampling distribution function.

In addition, the sampling approach has been applied to approximate the DoAs of passivity-based learning controllers at every learning trial. This online approximation can be used as a stopping criterion for the learning process. This allows learning to be terminated as soon as the controller's DoA is sufficiently large to satisfy the control objective. Thus, the proposed sampling method enables learning in a short amount of time.

4 CHAPTER

Learning Sequential Composition Control

This chapter proposes a new approach to enable the automatic synthesis of supervisory controllers via a learning sequential composition control. It augments the given pre-designed control system by learning new controllers online and on demand, using the actor-critic RL method. The learning process is always safe since the exploration in the course of learning the new controller only takes place within the DoAs of the existing controllers. The proposed approach has been implemented on two nonlinear systems: nonlinear mass-damper system and underactuated inverted pendulum. This learning control technique has also been extended for situations where no controller exists initially and all controllers have to sequentially be synthesized so as to achieve the control objective. This algorithm is demonstrated on a simulated example of mobile robot navigation.

4.1 Introduction

As discussed in Chapter 2, the standard sequential composition cannot address the tasks for which no controller was designed a priori in the supervisory structure. This chapter studies automatic synthesis of supervisory control systems using the paradigm of sequential composition. A learning sequential composition control algorithm is developed to learn new controllers by means of RL on demand. Once learning is complete the supervisory control structure is augmented with the new learned controllers. As a consequence, the overall DoA of the supervisor can incrementally cover larger areas of the state space on a need basis.

Suppose a humanoid robot with a task of fetching the mail for which a dedicated controller is designed for walking, one for climbing stairs, and one other for grasping and dropping objects. As such, the fetching mail task can be interpreted as a sequential composition of controllers in the form: walk to the mail room, grasp the mail, walk up the stairs, and finally drop the mail at the office. If during its task of delivering the mail it encounters a floor covered with unknown debris, due to building maintenance, and if a "walk on debris controller" is not a part of the database of controllers of the robot, then either the robot must stop or it must "try" to cross the debris. Sequential composition alone would result in a stop. This section aims at enabling the try option by augmenting the supervisory controller with a learning module.

This chapter proposes a learning sequential composition control approach to handle unmodeled situations by means of online learning. The aim of this chapter is to address three main questions as follows:

1. How to learn a new controller online that can be added to the existing supervisory control architecture?
2. How to guarantee that the learning process is safe?
3. 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.

For learning new controllers, the RL methods are implemented in which the controller is computed by interaction with the system, without the need of a model [116]. 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 SOS programming or by applying the sampling method. 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 chapter is organized as follows. Section 4.2 proposes the use of learning in sequential composition. Section 4.3 discusses rapid learning by exploiting the DoAs of the learning controllers and using passivity theory. In Section 4.4, simulation and experimental results are presented for the application of the proposed method on two nonlinear systems. Section 4.5 develops a feedback motion planning technique based on the learning sequential composition control. Finally, Section 4.6 provides a brief discussion and then concludes the chapter with some research lines for future work.

4.2 Learning Sequential Composition

Consider a sequential composition controller designed for an input-saturated inverted pendulum, see Figure 4.1(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 $\Phi_{\rm up}$ and $\Phi_{\rm down}$. Controller $\Phi_{\rm up}$ stabilizes the pendulum at the "up" equilibrium ($q = 0$) and controller $\Phi_{\rm down}$ at the "down" equilibrium ($q = \pi$). Figure 4.1(b) illustrates the state space of the pendulum with the approximated DoAs and goal sets. Since the control input is saturated, controller $\Phi_{\rm up}$ cannot swing the pendulum up from any initial state. Hence, $\mathcal{D}(\Phi_{\rm up})$ is represented by a conservative ellipsoid centered at point $(0, 0)$. Controller $\Phi_{\rm down}$ is globally stabilizing and its DoA is the entire state space, hence $\mathcal{D}(\Phi_{\rm down})$ is illustrated by a rectangle covering the whole state space. The goal sets of the up and down controllers are the points $\mathcal{G}(\Phi_{\rm up}) = \{(0, 0)\}$ and $\mathcal{G}(\Phi_{\rm down}) = \{(\pi, 0)\}$, respectively. Figure 4.1(c) depicts the control automaton, in which every mode s_i is associated with controller Φ_i. There is a prepare relation between controller $\Phi_{\rm up}$ and $\Phi_{\rm down}$ since $\mathcal{G}(\Phi_{\rm up}) \subset \mathcal{D}(\Phi_{\rm down})$, i.e. event $down$ connects mode $s_{\rm up}$ to $s_{\rm down}$. The supervisor is automatically synthesized based on the prepare relation described between the two controllers.

If the system starts in mode $s_{\rm down}$, the feasible string of events for the control automaton are $S_{\rm down} = \{down^*\}$, where the operator "$*$" denotes the Kleene closure [106], i.e. $S_{\rm down} = \{\epsilon, down, down\ down, \cdots\}$ with ϵ an "empty" mode. If the system starts at mode $s_{\rm up}$, the available strings are $S_{\rm up} = \{up^* down^*\}$. Thus, if the reference event signal is given as $S_{\rm ref} = down\ up$, the supervisory controller will block, because there is no arc connecting mode $s_{\rm down}$ to $s_{\rm up}$. In such a case, the supervisor needs a new controller to construct the required connections in the control automaton.

The standard formulation of a hybrid automaton [2] is described by the tuple $H = (Q, X, \mathcal{F}, {\rm Init}, {\rm Inv}, E, G, R)$, where

- $Q = \{q_1, q_2, ..., q_n\}$ is a finite set of discrete states.
- $X \subseteq \mathbb{R}^n$ is a set of continuous states.

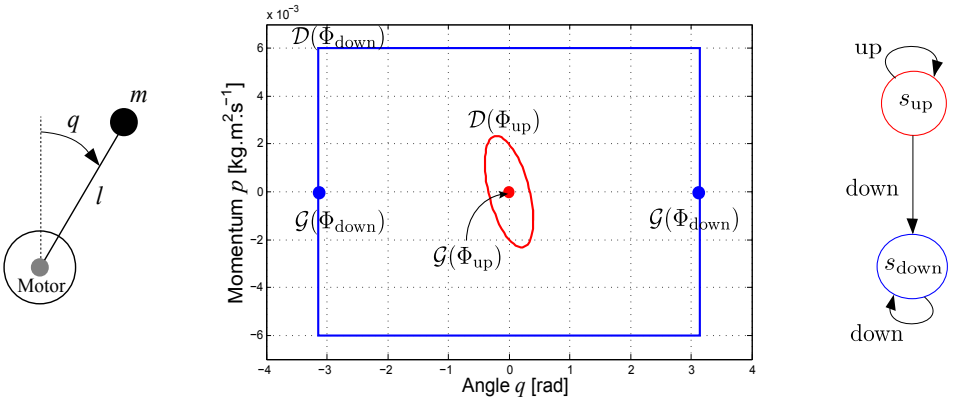

Figure 4.1: Sequential composition controller designed for an inverted pendulum. (a) Schematic representation of the inverted pendulum. (b) Approximated DoAs and goal sets. (c) Induced control automaton.

- $\mathcal{F} : Q \times X \to X$ is a vector field.
- $\text{Init} \subseteq Q \times X$ is a set of initial states.
- $\text{Inv} : Q \times \mathcal{P}(X)$ describes the invariants.
- $E \subseteq Q \times X$ is a set of edges.
- $G : E \to \mathcal{P}(X)$ is a guard condition.
- $R : E \to \mathcal{P}(X \times X)$ is a reset map.

In this definition, $\mathcal{P}(X)$ is the power set of X (i.e., the collection of all subsets of X) and the guard condition G returns a subset of X for each transition. The hybrid state is given by $(q, x) \in Q \times X$ and $\dot{x} = f(q, x)$ describes the evolution of the continuous state x in mode q.

The use of RL in sequential composition for online learning new controllers is proposed in [89], which is called learning sequential composition control. In this method, if the supervisor cannot find a sequence among the pre-designed 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, the standard formulation of a hybrid automaton [17] is adapted 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 4.1 *The tuple* $A_\ell = (\mathcal{X}, \mathcal{S}, E, (s_0, x_0), \Phi, F, D, G, g, \Gamma)$ *describes the learning control automaton, where the following holds.*

- $\mathcal{X} \subseteq \mathbb{R}^n$ *is the state space of a continuous-time system.*
- $\mathcal{S} = \{\epsilon, s_\ell, s_0, s_1, ..., s_q\}$ *is a finite set of discrete states or modes. Moreover, an empty mode* ϵ *and a learning mode* s_ℓ *is included. The hybrid state of the system is represented by the pair* $(s_i, x) \in \mathcal{S} \times \mathcal{X}$*.*
- $E = \{e_\ell, e_1, ..., e_p\}$ *is a finite set of events, where the learning event* e_ℓ *triggers the learning mode* s_ℓ*.*
- (s_0, x_0) *is the initial mode.*
- $\Phi = \{\Phi_\ell, \Phi_0, \Phi_1, ..., \Phi_q\}$ *is a set of controllers, where* Φ_ℓ *is an overall learning controller.*
- $F : \mathcal{S} \times \mathcal{X} \times \Phi \to \mathbb{R}^n$ *is a vector field that constrains the evolution of the continuous-time system to the differential equation* $\dot{x} = f\big(x, \Phi_i(x)\big)$*, with mode* $s_i \in \mathcal{S}/\{\epsilon\}$*.*
- $D : \mathcal{S} \to 2^{\mathcal{X}}$ *assigns to each mode* s_i *the DoA of its associated controller, hence* $D(s_i) = \mathcal{D}(\Phi_i)$*,* $D(\epsilon) = \emptyset$*, and* $D(s_\ell) = \mathcal{D}(\Phi)$*.*
- $G : \mathcal{S} \to 2^{\mathcal{X}}$ *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.*
- $g : \mathcal{S} \times E \to \mathcal{S}$ *is a discrete-event transition.*
- $\Gamma : \mathcal{S} \to 2^E$ *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, a binary function $P : \mathcal{X} \times \mathcal{X} \to \{\text{true}, \text{false}\}$ is executed. When P is false the learning mode is triggered. The binary function P is described as follows:

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

4.2.1 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 $S_{\text{ref}} = e_1 e_2 \dots e_r$ is available, the supervisor executes the associated controllers from the set $\{\Phi_1, \Phi_2, \cdots, \Phi_r\}$, sequentially. Each controller Φ_i needs an inherent time to evolve state $x_0^i \in \mathcal{D}(\Phi_i)$ to $x_{\text{d}}^i \in \mathcal{G}(\Phi_i)$ 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 $x_{\text{d}} \in \mathcal{X}$, 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 towards the desired state relies on the conditions outlined below [89]:

1. If $P(x_0, x_{\text{d}})$ is true, the standard sequential composition can drive the system from the initial state x_0 to the desired state x_{d}.

2. If $\neg P(x_0, x_{\text{d}})$, $\exists i : x_0 \in D(s_i)$, and $\exists j : x_{\text{d}} \in G(s_j)$, 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. See Figure 4.2.

3. If $\neg P(x_0, x_{\text{d}})$, $\exists i : x_0 \in D(s_i)$, $\nexists j : x_{\text{d}} \in G(s_j)$, but $\exists j : x_{\text{d}} \in D(s_j)$, a new controller is needed such that its goal set be the desired state. See Figure 4.3.

4. If $\neg P(x_0, x_{\text{d}})$ and $\nexists j : x_{\text{d}} \in D(s_j)$, 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. See Figure 4.4.

5. If $\neg P(x_0, x_{\text{d}})$ 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 which is not consider in this thesis. See Figure 4.5.

Table 4.1 summarizes the conditions that can happen for the learning sequential composition controller.

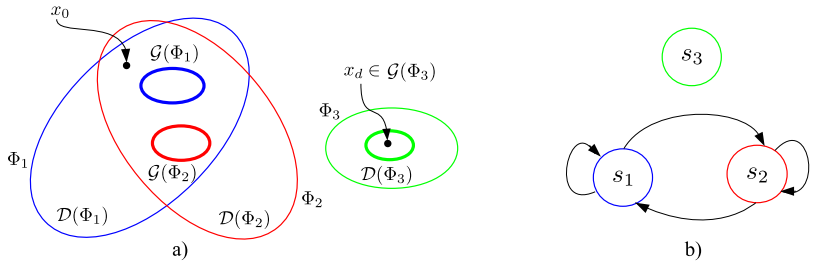

Figure 4.2: a) Pictorial sketch of the DoAs and the goal sets of a control system in which the initial state x_0 is in the DoA of a controller and the desired state x_d lies in the goal set of another controller, but there are no event transitions connecting these two controllers; b) the induced control automaton.

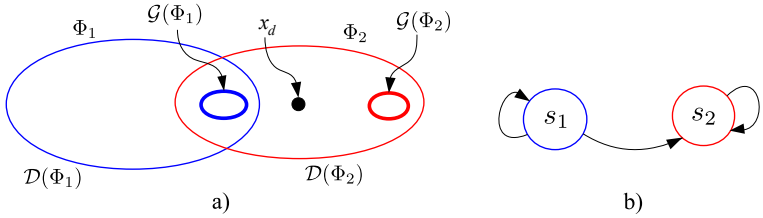

Figure 4.3: a) Pictorial sketch of the DoAs and the goal sets of a control system in which the desired state x_d is not in the goal set of any controller, but lies in the DoA of a controller; b) the induced control automaton.

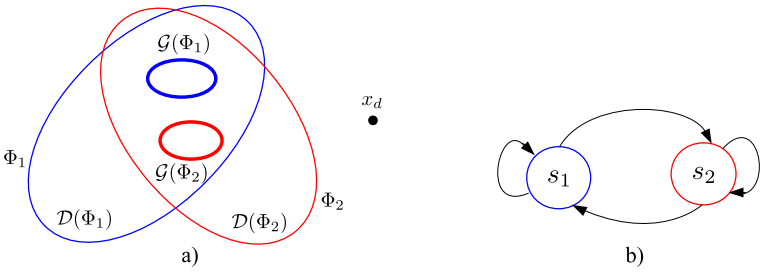

Figure 4.4: a) Pictorial sketch of the DoAs and the goal sets of a control system in which the desired state x_d does not lie inside the goal set of any controller; b) the induced control automaton.

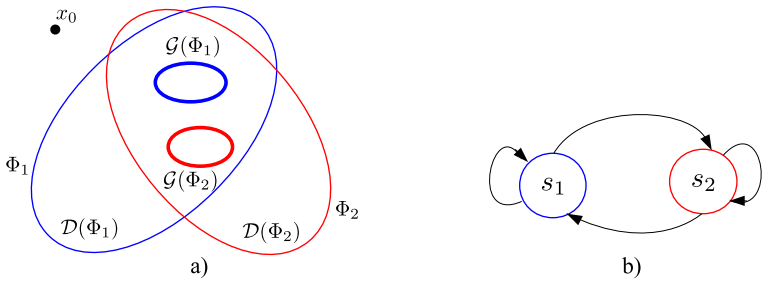

Figure 4.5: a) Pictorial sketch of the DoAs and the goal sets of a control system in which the initial state x_0 does not lie inside the DoA of any controller; b) the induced control automaton.

Table 4.1: Various conditions that might happen for the designed sequential composition controller

$P(x_0, x_\mathrm{d})$	$\exists j : x_d \in G(s_j)$	$\exists j : x_d \in D(s_j)$	$\exists i : x_0 \in D(s_i)$	Condition
true	true	true	true	–
false	true	true	true	2
false	true	true	false	5
false	false	true	true	3
false	false	true	false	3, 5
false	false	false	true	4
false	false	false	false	4, 5

The traditional sequential composition is not designed to cope with the situation $\neg P(x_0, x_\mathrm{d})$ (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 chapter, the actor-critic RL method is implemented.

The stability of the learning sequential composition controller can be addressed by three stability sub-problems. The first is the stability of the pre-designed controllers. Based on the properties of sequential composition, it is assumed that every pre-described 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, handled by the prepare relation.

4.2.2 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 union set of the existing controllers' DoAs. 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. This restricted learning process is called bounded learning. Consider the condition $\neg P(x_0, x_\mathrm{d})$, with $\exists i : x_0 \in D(s_i)$ and $\exists j : x_\mathrm{d} \in D(s_j)$. 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. Figure 4.6 illustrates a sequential composition controller with two controllers Φ_1 and Φ_2, where $x_0 \in \mathcal{D}(\Phi_1)$ and $x_\mathrm{d} \in \mathcal{D}(\Phi_2)$, but $x_\mathrm{d} \notin \mathcal{G}(\Phi_2)$. 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.

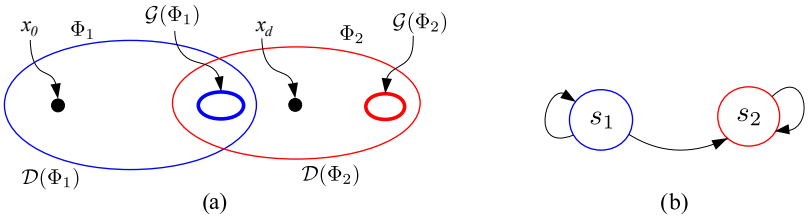

Figure 4.6: (a) Pictorial sketch of the DoAs and goal sets of a sequential composition controller, where $x_0 \in \mathcal{D}(\Phi_1)$, $x_\mathrm{d} \in \mathcal{D}(\Phi_2)$, and controller Φ_1 prepares controller Φ_2. (b) Induced control automaton, in which the learning mode can make the required connection s_2 to s_1 using bounded learning.

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 Figure 4.7. 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. This is called unbounded learning in the sense that the learner is not restricted to just explore within the overall DoA. This chapter only considers bounded learning.

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

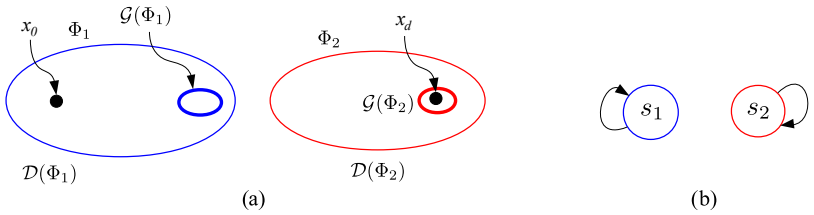

Figure 4.7: (a) Pictorial sketch of the DoAs and goal sets of a sequential composition controller, where $x_0 \in \mathcal{D}(\Phi_1)$ and $x_\mathrm{d} \in \mathcal{G}(\Phi_2)$, but controller Φ_1 does not prepare controller Φ_2. (b) the induced control automaton, in which the learning mode can only make the required connections using unbounded learning.

4.3 Rapid Learning

Learning processes can be time-consuming and computationally costly. A major challenge in RL is its non-reliance on models and prior knowledge. In practice, this results in RL processes usually exploring the entire state space to find an approximately optimal control law. In this work, partial prior knowledge of the system is used to speed up the learning process. By combining the learning methods with PBC, the learning speed can be increased considerably with respect to the standard learning methods [81]. As a consequence, the complexity of control synthesis in PBC is decreased. In addition to this approach, the DoA of the new learned controller is monitored after each learning trial. As such, the supervisor can terminate learning as soon as the new controller's DoA is sufficiently large to satisfy the learning objective [91].

While learning is in progress, the DoA of the new controller typically enlarges around its goal set, but not necessarily monotonically. As such, if the DoA of the learned controller is always monitored, the supervisor can terminate learning as soon as the DoA is 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 compared with the regard to the conventional learning methods, thanks to the PBC structure and DoA estimation [91].

Algorithm 7 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.

Algorithm 7 Learning sequential composition control using the EB-AC algorithm

Require: system (2.15), A_ℓ, x_0, x_d, λ, γ, α_a, α_c, n_s, n_t

if $P(x_0, x_\mathrm{d})$ is true **then**
  Execute: sequential composition controller
else
  Execute: learning mode s_ℓ
  $w \leftarrow 0$
  Initialize ξ_0, ψ_0
  repeat
    $w \leftarrow w + 1$
    Initialize x_0
    for $k = 1$ **to** n_s **do**
      Energy-balancing actor-critic:
        $u_k = \mathrm{sat}\left(\hat{\pi}(x_k, \xi_k, \psi_k) + \Delta u_k\right)$
        Apply actor-critic RL to update ξ_k and ψ_k based on Algorithm 2
    end for
    EB-AC controller Φ_ℓ
    Estimate the DoA $\mathcal{D}(\Phi_\ell)$ based on Algorithm 5
  until $\mathcal{G}(\Phi_i) \subset \mathcal{D}(\Phi_\ell)$
  Add controller Φ_ℓ to the learning control automaton
end if

4.4 Simulation and Experimental Results

The proposed learning sequential composition is implemented on two nonlinear dynamical systems. The first addresses a positioning problem in a simulated second-order system consisting of a mass with a nonlinear damper, where the control input is saturated. The second studies the stabilization of a physical inverted pendulum with saturated control input.

4.4.1 System 1: Nonlinear Mass-Damper

Consider a mass with a nonlinear damper, as illustrated in Figure 4.8. The dynamics are given by

$$m\ddot{q} = -B(q)\dot{q} + u \tag{4.1}$$

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 $\pm 3\,\mathrm{N}$. The state vector of the system is described by $x = [q\ p]^T$, where $p = m\dot{q}$ is the momentum with $m = 1\,\mathrm{kg}$.

The sequential composition controller consists of two LQR controllers Φ_1 and Φ_2. Controller Φ_1 steers the mass to point $(q, p) = (-0.7, 0)$ and controller Φ_2 to point $(q, p) = (1, 0)$. To design these controllers, the equation of motion (4.1) is linearized

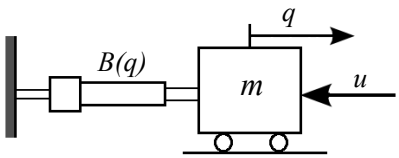

Figure 4.8: Schematic representation of a nonlinear mass-damper system with damping coefficient $B(q)$.

around the operating points. Then, the gain matrices for the linearized system are computed as $K_1 = [0.447\ 0.654]$ and $K_2 = [2.000\ \ 0.663]$. Figure 4.9(a) presents the approximated DoAs and goal sets of controllers Φ_1 and Φ_2 and Figure 4.9(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}.

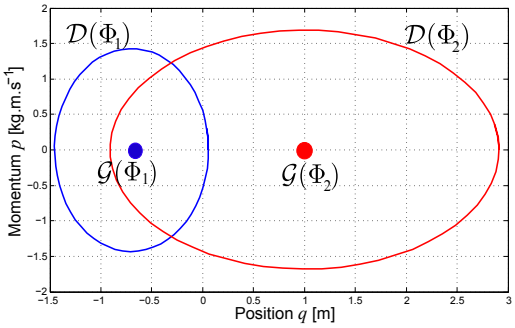

Figure 4.9: Sequential composition controller designed for the nonlinear mass-damper system. (a) Approximated DoAs and goal sets. (b) Induced control automaton.

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. This example applies the A-IDA-AC controller for the learning mode, which is defined by (2.36). The reward function is described as

$$\rho(x_{k+1}, u_k) = -\alpha_1 q_{k+1}^2 - \alpha_2 \dot{q}_{k+1}^2 - \alpha_3 u_k^2 \tag{4.2}$$

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. Figure 4.10 presents the sum of

rewards that a learning controller receives per trial over a simulated experiment with 60 trials.

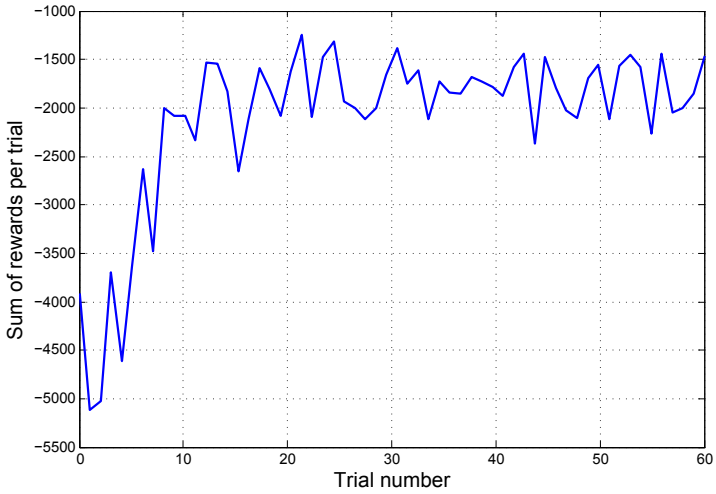

Figure 4.10: Sum of rewards that a learning controller receives per trial over a simulation with 60 trials for the nonlinear mass-damper system.

The desired Hamiltonian of the system is chosen to be quadratic as $H_{\mathrm{d}}(x) = x^T \Lambda x$ with $\Lambda = [1\ 0.5; 0.5\ 1]$ a symmetric positive definite matrix. The desired Hamiltonian is used 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, SOS programming is used to approximate the DoAs by following the same procedure described in [23]. Figure 4.11 shows the DoA of the new learned controller Φ_ℓ after seven specific trials (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. Additionally, Figure 4.12 presents the average learning curve received for 50 simulations, each including 60 trials. Once the learner receives the required sum of rewards per trial, the DoA of the learned controller is sufficiently large to cover state $(0.05, 0)$.

Figure 4.13 illustrates the learning control automaton during and after learning. In Figure 4.13(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 the learned controller, the learning process can be terminated. Figure 4.14 depicts the approximated DoA of the learned controller Φ_ℓ together with

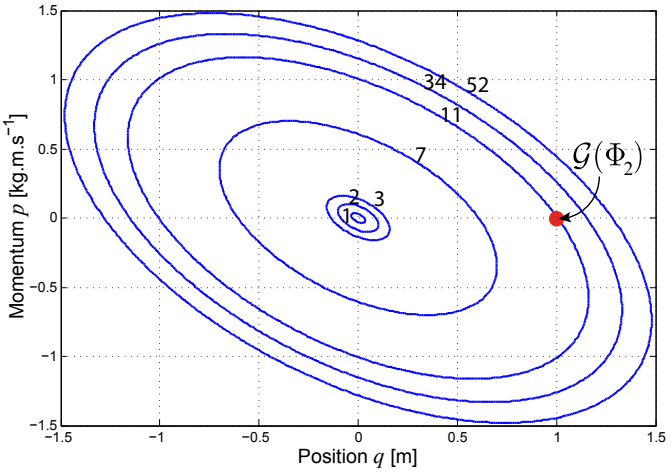

Figure 4.11: Approximated DoAs of the learned controllers after seven specific trials for the nonlinear mass-damper system. The trial numbers are also indicated.

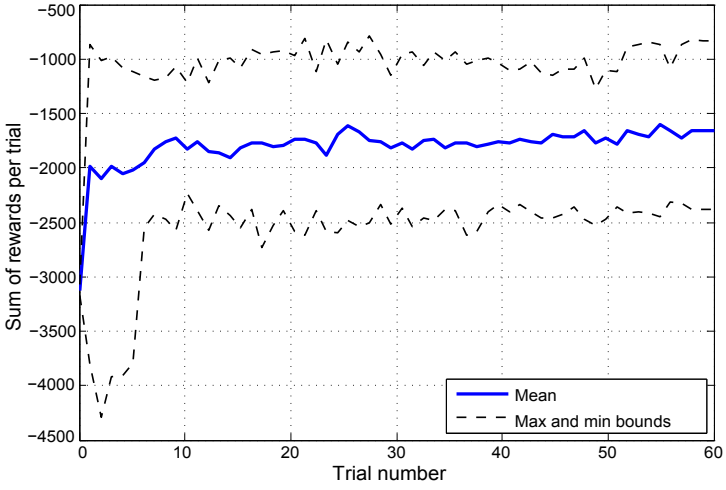

Figure 4.12: Average learning curve received for 50 learning simulations, each lasting one minute (60 trials).

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 Figure 4.13. Consequently, the resulting learning sequential composition controller is able to switch from controller Φ_2 to Φ_1 via the new learned controller.

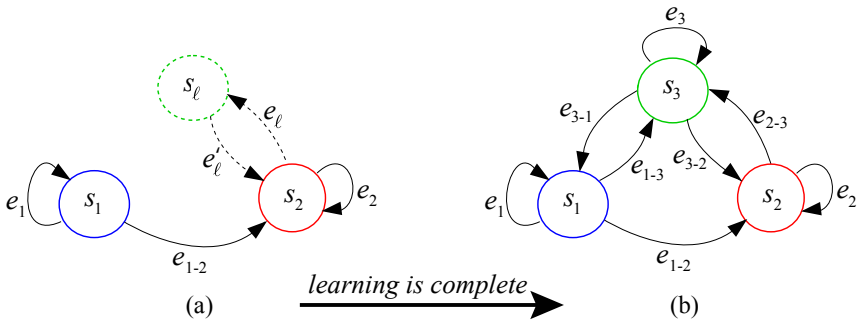

Figure 4.13: Learning control automaton for the nonlinear mass-damper system. (a) During learning. (b) After learning.

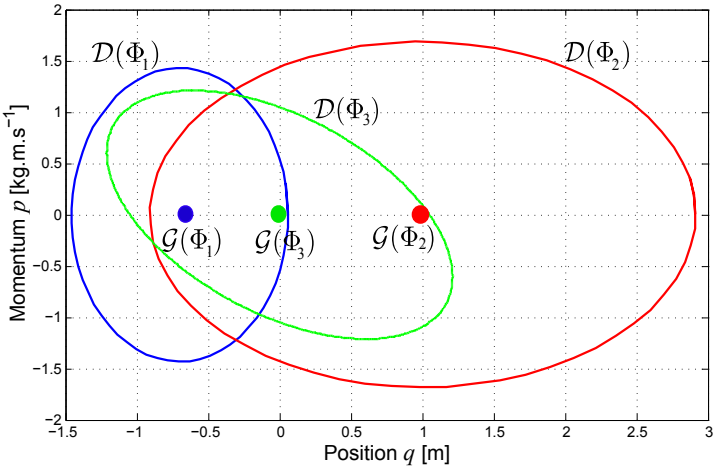

Figure 4.14: Approximated DoAs and goal sets of the controllers for the nonlinear mass-damper system after learning.

4.4.2 System 2: Inverted Pendulum

The inverted pendulum, as shown in Figure 4.15, is modeled by the nonlinear equation of motion

$$J\ddot{q} = mgl\sin(q) - \left(b + \frac{K^2}{R}\right)\dot{q} + \frac{K}{R}u \tag{4.3}$$

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 4.2 presents the physical parameters of the pendulum. The values are found partly by measuring and partly estimated using nonlinear system identification.

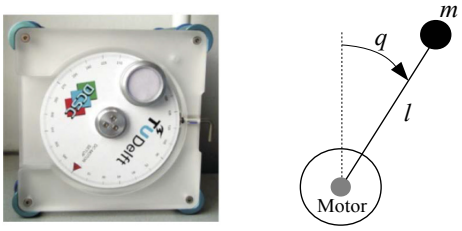

Figure 4.15: Inverted pendulum and its schematic.

Table 4.2: Physical parameters of the inverted pendulum

Physical parameter	Symbol	Value	Unit
Pendulum inertia	J	1.91×10^{-4}	kg·m^2
Pendulum mass	m	6.8×10^{-2}	kg
Gravity	g	9.81	m·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}	Nm·A^{-1}
Rotor resistance	R	9.5	Ω

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 (4.3) 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 $K_{\text{up}} = [8.486\ 3.439 \times 10^3]$, which generate the approximated DoAs represented in Figure 4.1(b).

The control task is defined as tracking a reference string of events $S_{\text{ref}} = (up\ down)^*$ where event up triggers mode s_{up} and event $down$ triggers mode s_{down} in the con-

trol 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 $S_{\text{ref}}(0) = up$ can be executed since mode s_{up} has a self triggered event up [3]. If $x \in \mathcal{G}(\Phi_{\text{up}})$, the supervisor fires the next event $S_{\text{ref}}(1) = down$. 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 $S_{\text{ref}}(2) = up$, 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. The reward function for the learning method is defined by

$$\rho(x_{k+1}, u_k) = -\alpha_1\big(1 - \cos(q_{k+1})\big) - \alpha_2 \dot{q}_{k+1}^2 - \alpha_3 u_k^2 \tag{4.4}$$

which gives higher rewards to the transitions where the learning controller can stabilize the pendulum at the up equilibrium. In this example, two different methods are implemented to learn the unknown parameters of the control input: the EB-AC algorithm, and the A-IDA-AC algorithm.

Learning via Energy-Balancing Actor-Critic

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 if the learner cannot reach $\mathcal{D}(\Phi_{\text{up}})$ after a number of samples. Figure 4.16 represents the sum of rewards that a learning controller receives per trial over a simulated experiment with 60 trials, each lasting 1 s.

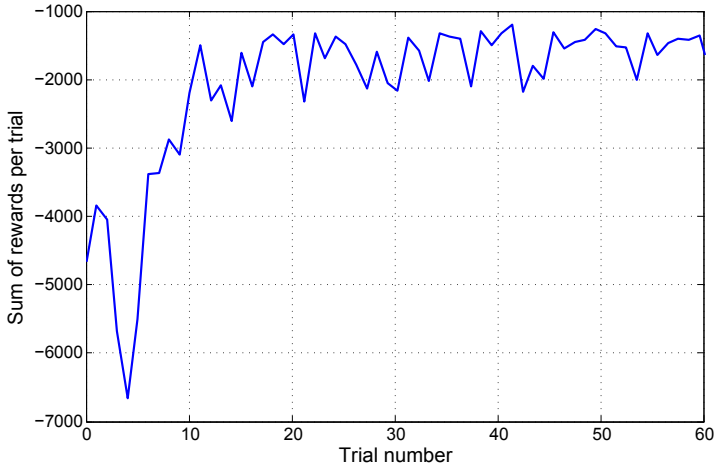

Figure 4.16: Sum of rewards that a learning EB-AC controller receives per trial over a simulation with 60 trials for the inverted pendulum.

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, the sampling method is used to approximate the DoAs by following the same procedure described in [71]. Figure 4.17 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.

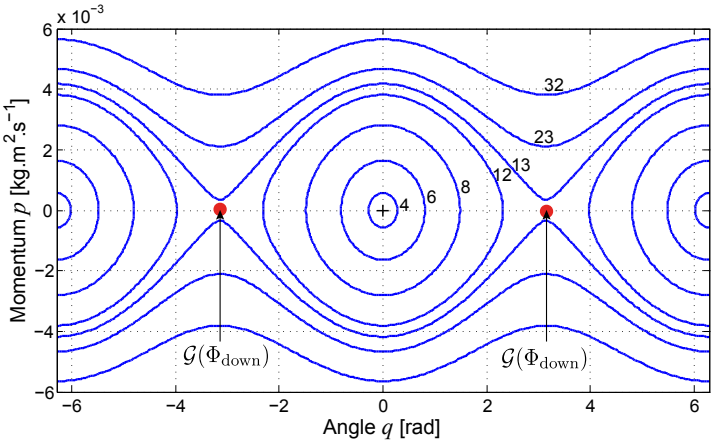

Figure 4.17: Approximated DoAs of the learned EB-AC controllers after seven specific trials for the inverted pendulum. The trial numbers are also indicated.

Figure 4.18 shows the learning control automaton during and after learning. In Figure 4.18(a), event e_ℓ connects mode s_{down} 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. Figure 4.19 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 Figure 4.18(b).

Figure 4.20 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

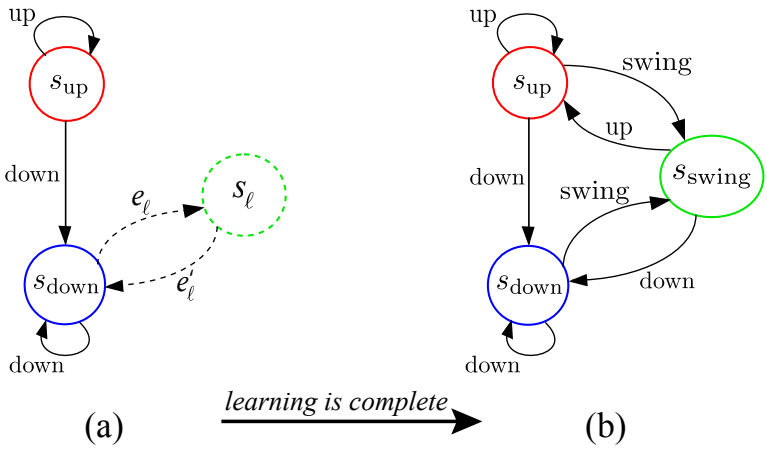

Figure 4.18: Learning control automaton for the inverted pendulum. (a) During learning. (b) After learning.

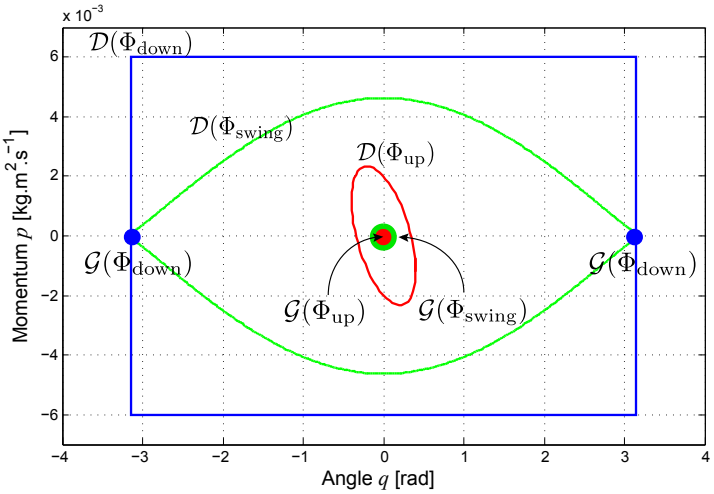

Figure 4.19: The approximated DoAs and goal sets of the controllers for the inverted pendulum after learning.

$\mathcal{G}(\Phi_{\text{down}}) \subset \mathcal{D}(\Phi_{\text{swing}})$. Although learning can be terminated at this stage, it runs for 60 trials for illustration purposes. Once learning is completed, the new mode s_{swing} is added to the learning control automaton. The learning sequential composition controller can track the reference event up via events $swing$ and up.

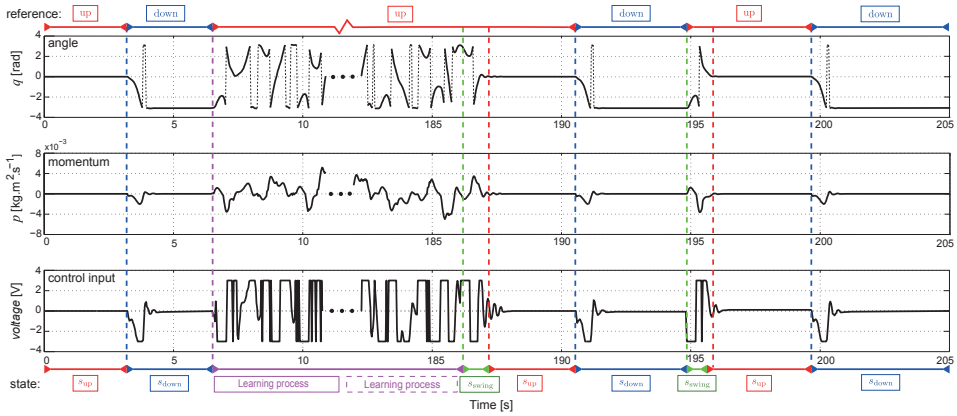

Figure 4.20: Experimental results of the learning sequential composition controller for the inverted pendulum. The reference events are displayed at the top and the controllers at the bottom. When the pendulum is in the down mode and the reference event is up, the control system learns how to swing the 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 resulting 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`.

Learning via Algebraic Interconnection and Damping Assignment Actor-Critic

The objective of the algebraic IDA-PBC method is to find a feed-back control law $u(x)$ such that the closed-loop system is described as

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -F_{21}(x) & -b \end{bmatrix} \begin{bmatrix} \nabla_q H_{\text{d}}(x) \\ \nabla_p H_{\text{d}}(x) \end{bmatrix}. \tag{4.5}$$

Moreover, the desired Hamiltonian is chosen in a quadratic form

$$H_{\text{d}}(x) = \frac{1}{2}\gamma_{\text{q}}(q - q_{\text{d}})^2 + \frac{p^2}{2J} \tag{4.6}$$

where γ_{q} is a unit conversion factor. This desired Hamiltonian satisfies the equilibrium condition (2.16) at the desired state $x_{\mathrm{d}} = (q_{\mathrm{d}}, p) = (0, 0)$. Substituting (4.3), (4.6), and (4.6) in (2.28), then using Fourier basis functions to approximate F_{21} by $F_{21}(x, \vartheta) = \vartheta^T \phi(x)$, the control law is obtained as

$$u = -\vartheta^T \phi(x)\gamma_{\mathrm{q}}(q - q_{\mathrm{d}}) - mgl\sin(q). \tag{4.7}$$

The unknown parameter vector ϑ is learned using the actor-critic method [82], as defined in Algorithm 3, with given learning parameters in Table 4.3. Suppose the case that there is no up and down controllers a priori and the supervisor has to learn an A-IDA-AC controller to be able to achieve both swing-up and stabilization of the pendulum at the up equilibrium.

Suppose the case where none of the up and down controllers were designed a priori and the supervisor has to learn an A-IDA-AC controller to be able to achieve both swing-up and stabilization of the pendulum at the up equilibrium. Two controllers are learned: one for simulation and one for the physical inverted pendulum. Figure 4.21 illustrates the sum of rewards that a learning control law receives per trial over a 80 trials learning process in simulation and experiment.

Table 4.3: Learning parameters of the inverted pendulum

Parameter	Symbol	Value	Unit
Sample time	T_{s}	0.03	s
Trial time	T_{t}	3	s
Number of trials	–	60	–
Decay rate	γ	0.97	–
Eligibility trace decay	λ	0.65	–
Exploration variance	σ^2	1	–
Learning rate of critic	α_{c}	0.01	–
Learning rate of $F_{21}(x)$	$\alpha_{\mathrm{a}\vartheta}$	1×10^{-8}	–
Max control input	$u_{\max}$	3	V

In the A-IDA-AC method, the Hamiltonian of the system is computed at every learning trial which is exploited as a candidate Lyapunov function to approximate the DoAs of the learned controllers. Figure 4.22 illustrates the DoAs computed by the sampling method at seven trials in simulation. The numbers of sample trials are also given. While learning is in progress, typically the DoAs of the learned controllers enlarge, but not necessarily monotonically. Here, the DoA of the controller is large enough almost after 40 trials to cover the initial state. As such, the learning process can be terminated instead of running for the entire scheduled trials. Using the sampling method not only approximates the DoA very fast, but also speeds up the process of control design because of its described learning stopping criterion for learning.

Figure 4.23 depicts the angle and angular momentum of the pendulum with re-

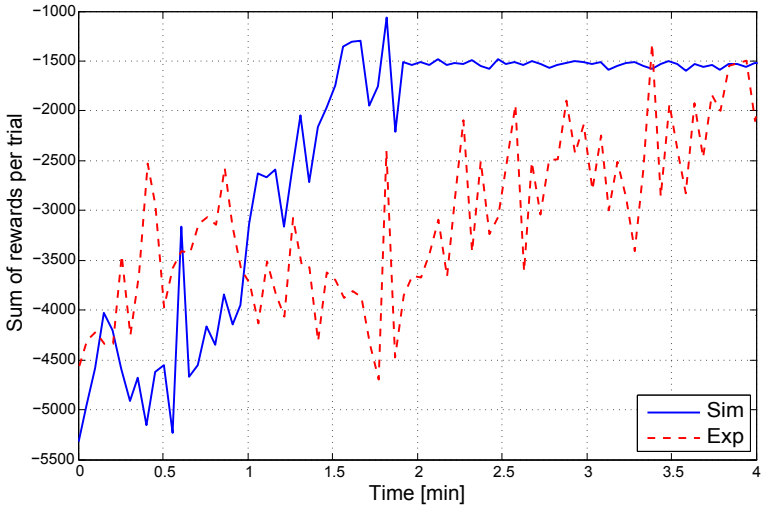

Figure 4.21: Sum of rewards that a learning A-IDA-PBC controller receives per trial over a learning process with 80 trials (each lasting 0.03 s) for the inverted pendulum in simulation and experiment.

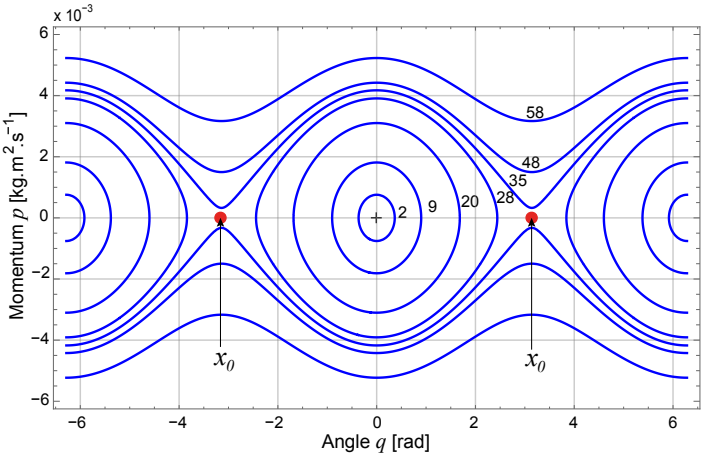

Figure 4.22: Approximated DoAs of the learned A-IDA-AC controllers after seven specific trials for the inverted pendulum. The trial numbers are also indicated.

spect to the learned A-IDA-AC controller in simulation and experiment. Since the control input is saturated to 3 volts, the pendulum first needs to achieve the required momentum by swinging back and forth. Once it reserves the sufficient energy, the controller can first swing the pendulum up and then stabilize it at the up equilibrium.

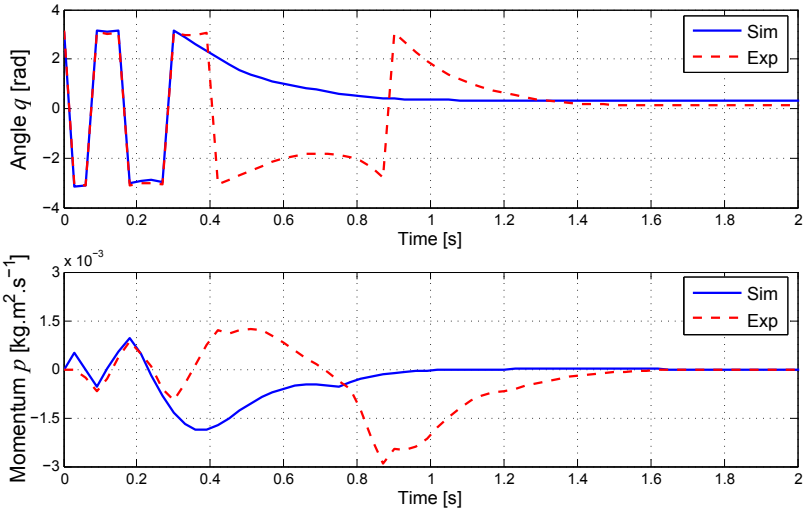

Figure 4.23: Simulation and experimental results of the learned A-IDA-AC controller for the inverted pendulum.

4.5 Probabilistic Learning Trees

In the previous sections, learning sequential composition control is introduced and the case of incomplete supervisory controller is discussed where extra controllers are learned, on a need basis, to enable the supervisor to cope with unforeseen situations. In this section, the situation is investigated in which there is no controller designed a priori and all controllers have to be learned from scratch and on a random manner. In fact, the control law A-IDA-AC computed by (2.36) is combined with DoA estimation to automatically build a collection of stabilizing controllers. Since the equilibrium is chosen randomly at each step and the supervisor constructs trees using the DoAs of the learned controllers, this control approach is called probabilistic learning trees (PLTs).

Algorithm 8 describes the main steps of the proposed learning trees, where a set of controllers are learned such that the union of their DoAs covers the desired range of the state space to be controlled, denoted $\bar{\mathcal{X}}$. The control automaton stores the s_i tuples, each consisting of the controller Φ_i, its DoA $\mathcal{D}(\Phi_i)$ and goal set $\mathcal{G}(\Phi_i)$. In

steps 1 and 2 of the algorithm, the control automaton is initialized together with the overall desired equilibrium x_0^* for this method. In step 6, an A-IDA-AC learning experiment is executed, based on Algorithm 3, for the translated dynamics, denoted $\bar{f}$, from the desired equilibrium x^* to the origin for n_s samples and n_t trials. The learning experiment can be stopped and instantly switched to an existing stabilizing controller if the state is about to leave the current union of all DoAs. This is a form of safe learning. Clearly this is not possible for the first learning experiment since no stabilizing controller exists yet.

Algorithm 8 Probabilistic learning trees algorithm

Require: $\beta, c^*, n_\mathrm{s}, n_\mathrm{t}$

1: Initialize list of controllers $\mathcal{C},\ k = 0$, with $\#\mathcal{C} = k$
2: Initialize desired equilibrium x_k^*
3: **repeat**
4: Initialize θ_0, ϑ_0
5: **repeat**
6: $(\theta, \vartheta) \leftarrow$ *A-IDA-AC*$(\bar{f}, x_k^*, \theta, \vartheta)$ runs a learning experiment for the desired equilibrium x_k^* with n_s samples and n_t trials based on Algorithm 3
7: $\bar{x}_k^* \leftarrow \mathrm{ComputeEquilibrium}(\bar{f}, \hat{\pi}(x, \vartheta), x_k^*)$ computes the actual equilibrium $\bar{x}_k^*$ for the learned controller
8: $c_k \leftarrow$ *DoA*$(\bar{f}, \hat{\pi}(x, \vartheta), \bar{x}_k^*)$ approximates the DoA of the learned controller based on Algorithm 5
9: **until** $||x_k^* - \bar{x}_k^*|| < \beta \ \wedge \ c_k > c^* \ \wedge \ \exists i \neq k : \bar{x}_k^* \in \mathcal{D}(\Phi_i)$
10: $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\Phi_k, \mathcal{D}(\Phi_k), \mathcal{G}(\Phi_k)\})$ saves new controller, its DoA and goal set.
11: $x_{k+1}^* \leftarrow \mathrm{FindNewDesiredEquilibrium}\,(\mathcal{D}(\Phi))$
12: $k \leftarrow k + 1$
13: **until** the union of the DoAs covers the desired range $\bar{\mathcal{X}}$

Once the learning process finishes for a pre-defined number of samples and trials two steps take place. First, the function "ComputeEquilibrium", defined in step 7, calculates the actual equilibrium $\bar{x}^*$ of the closed-loop system is computed. In general there is no guarantee that for a finite number of learning trials the closed-loop system will converge to the desired equilibrium. Consider the resulting learning controller after one single sample, it will most likely not stabilize at x^*. As such it is important to track how this "learned equilibrium" evolves, such that it can be used to correctly compute the DoA. The minimum distance between the actual learning stabilizing equilibrium and the desired equilibrium is denoted by β. Second, once the actual equilibrium $\bar{x}^*$ is obtained the DoA of its associated stabilizing controller is computed. This is achieved by finding the largest sublevel set $\mathcal{L}(c_k)$ at each trial such that $\forall x : H_\mathrm{d}(x, \bar{x}^*) < c_k$ then $\dot{H}_\mathrm{d}(x, \bar{x}^*) < 0$. Here, $H_\mathrm{d}(x, \bar{x}^*)$ is the translated Hamiltonian from the desired equilibrium to the origin. Note that H_d is used as a candidate Lyapunov function. The learning process for the desired equilibrium x_k^* is repeated if any of the following criteria are not fulfilled:

- $||x_k^* - \bar{x}_k^*|| < \beta$. In practice this means that the actual learned stabilizing equilibrium $\bar{x}_k^*$ should be sufficiently close to the desired equilibrium x_k^*. In the early states of learning, it can happen that the system quickly converges to a nearby attractor of the vector field f. In this situation, the closed-loop actual equilibrium $\bar{x}_k^*$ matches with the intrinsic equilibrium of the attractor of the uncontrolled f. If an approximation of the DoA of this attractor is performed, it might be very large, not due to the controller but due to the intrinsic properties of f. Finding such a large DoA without looking at the location of $\bar{x}_k^*$ distracts from the task at hand. It is important then to make sure that $\bar{x}_k^*$ is close to x_k^* such that the learning process can be terminated at the right time.
- $c_k > c^*$. The resulting DoA should be at least larger than a pre-defined sublevel set $\mathcal{L}(c^*)$, where c^* is the minimum DoA level.
- $\exists i \neq k : \mathcal{G}(\Phi_k) = \bar{x}_k^* \in \mathcal{D}(\Phi_i)$. The goal set $\bar{x}_k^*$ of the learned controller must lie at least within the DoA of an existing controller Φ_i. This is required in order to make the overall desired equilibrium x_0^* reachable from all states, via the sequential composition. This condition does not apply for the first controller.

Once all of these criteria are realized, a new controller is stored in the control automaton. Next, a new desired equilibrium must be found to explore new regions of the state space. There are many approaches to find such a new equilibrium. Here, the level sets of the DoA are used to check if a point lies in a δ-boundary of $\mathcal{D}(\Phi)$, with $0 < \delta < 1$. A test is constructed based on the logic expression

$$z \in \bar{\mathcal{X}} \cap \left(\bigcup_k \{x : H_{\mathrm{d}}(x, \bar{x}_k^*) < c_k\} \right) \cap \left(\bigcap_k \{x : H_{\mathrm{d}}(x, \bar{x}_k^*) > \delta c_k\} \right). \tag{4.8}$$

If a state z can be found to fulfill (4.8), then such a point lies in the desired boundary. In Algorithm 8, the function "FindNewDesiredEquilibrium" searches for the new desired equilibrium within the area defined by (4.8).

Algorithm 8 runs until expression (4.8) evaluates to false for all states. The result is a list of controllers with their DoAs and goal sets stored in $\mathcal{C}$. The induced control automaton is found by checking the prepare relation in $\mathcal{C}$ and following

1. Create a node s_i for each controller Φ_i.
2. $\forall i, j$ create an arc from $s_i \to s_j$ if $\mathcal{G}(\Phi_i) \subset \mathcal{D}(\Phi_j)$.

As an example, consider a sequential composition controller designed for a land robot that automatically explores through an unstructured terrain, as illustrated in Figure 4.24. Due to the complexity of navigating the robot through a large environment, a controller is dedicated to locomotion in grass in a particular region

of the workspace, a second in traversing sand, a third on swimming, a fourth on rocks, etc. To achieve the described plan, the supervisor executes a set of controllers sequentially. For instance, while traversing a river, the supervisory controller would generate the sequence of states: grass, sand, water, sand, and rocks. The proposed PLTs algorithm is applied to design the controllers for a simplified version of this navigation problem.

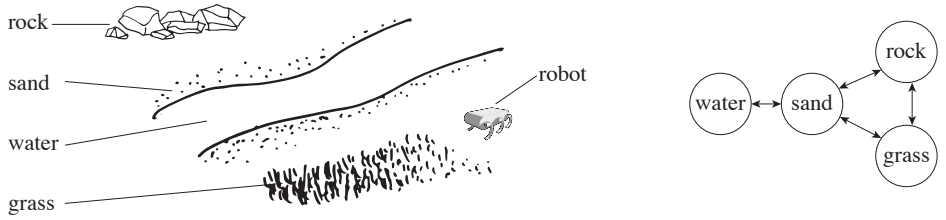

Figure 4.24: Sequential composition controller for a land legged robot to traverse various terrains.

The design of each control law can be simplified as the velocity control of a mass on a viscous landscape. This problem is described by a two-dimensional first-order system of the form

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2) + u_1 \\ \dot{x}_2 &= f_2(x_1, x_2) + u_2 \end{aligned} \tag{4.9}$$

where $x = [x_1\ x_2]^T$ is the state vector with x_1 and x_2 the mass position along the x and y axes, respectively. Moreover, function $f = [f_1\ f_2]^T$ is obtained by taking the gradient of a potential function $h(x_1, x_2)$ as

$$f_i(x_1, x_2) = \nabla_{x_i} h(x_1, x_2) = \frac{\partial h(x_1, x_2)}{\partial x_i}. \tag{4.10}$$

The potential function h defines the shape of the landscape described as the sum of Gaussians in the form

$$h(x_1, x_2) = \sum_{i=0}^{n} a_i \exp\left((x_1 - \mu_{i1})^2 + (x_2 - \mu_{i2})^2\right) \tag{4.11}$$

with parameters given in Table 4.4. The values of these parameters are fixed for the sake of reproducibility.

The choice of dynamics f is arbitrary and was chosen to keep the dimension and complexity of the system low to avoid the curse of dimensionality inherent to RL. The resulting vector field f and potential h are illustrated in Figure 4.25.

Table 4.4: Parameters of the potential function h described in (4.11)

i	a_i	μ_{i1}	μ_{i2}	σ_i
1	-0.1	-1.4	2.5	1.8
2	0.2	1.3	2.2	1.5
3	0.3	-3.4	-2.5	2

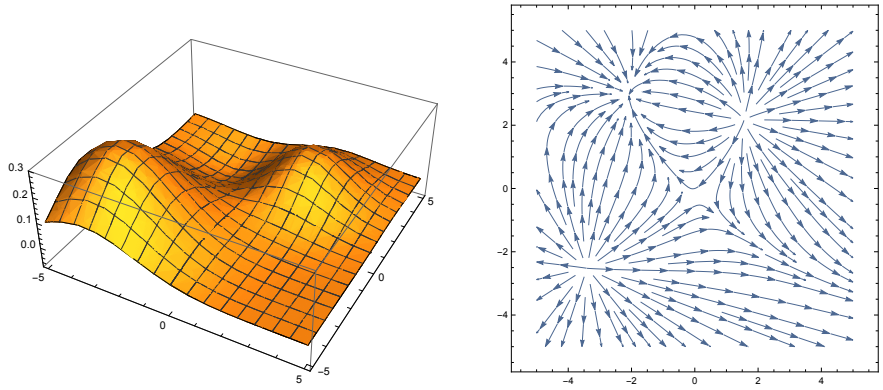

Figure 4.25: Dynamics used for the example of the land robot navigation. The potential function h is represented on the left side and its gradient that describes the vector field f is depicted on the right side for the range $x_1 \in [-5, 5],\ x_2 \in [-5, 5]$.

The A-IDA-AC control law (2.36) is calculated with the input matrix

$$g(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.12}$$

and the quadratic desired Lyapunov function

$$H_{\mathrm{d}}(x) = \frac{1}{2} x^T x. \tag{4.13}$$

The learning structure follows Algorithm 3 with the parameters of Table 4.5. The polynomial basis functions are applied to represent the approximated value function $\hat{V}(x, \theta)$ and approximated policy $\hat{\pi}(x, \vartheta)$ as

$$\begin{aligned} \hat{V}(x,\theta) &= \theta^T \Psi_{\mathrm{c}}(x) = \textstyle\sum_{i=0}^{\bar{n}_{\mathrm{c}}} \sum_{j=0}^{\bar{n}_{\mathrm{c}}} \theta_{(i+(n_{\mathrm{c}}+1)j)} x_1^i x_2^j \\ \hat{\pi}(x,\vartheta) &= \vartheta^T \Psi_{\mathrm{a}}(x) \nabla_x H_{\mathrm{d}}(x) \\ &= \textstyle\sum_{i=0}^{\bar{n}_{\mathrm{a}}} \sum_{j=0}^{\bar{n}_{\mathrm{a}}} x_1^i x_2^j \begin{bmatrix} x_1 & 0 \\ 0 & x_2 \end{bmatrix} \vartheta_{(i+(n_{\mathrm{a}}+1)j)} \end{aligned} \tag{4.14}$$

Table 4.5: Learning parameters for the example of the land robot navigation

Parameter	Symbol	Value	Unit
Sample time	T_s	0.01	s
Number of samples	n_{s}	200	–
Number of trials	n_{t}	50	–
Decay rate	γ	0.97	–
Learning rate of actor	α_{a}	0.0005	–
Learning rate of critic	α_{c}	0.02	–
Basis function parameters for each actor	$\bar{n}_{\mathrm{a}}, n_{\mathrm{a}}$	1, 4	–
Basis function parameters for critic	$\bar{n}_{\mathrm{c}}, n_{\mathrm{c}}$	1, 4	–
Boundary for find new desired equilibrium parameter	δ	0.5	–
Minimum distance between x^* and $\bar{x}^*$	β	0.2	–
Minimum DoA level	c^*	1	–

where each parameter ϑ_k is a two-dimensional vector. It is found the basis function parameters for the actor and critic $\bar{n}_{\mathrm{a}} = \bar{n}_{\mathrm{c}} = 1$ to be sufficient for this system resulting in

$$\begin{aligned} \hat{V}(x,\theta) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 \\ \hat{\pi}(x,\vartheta) &= \begin{bmatrix} \vartheta_{0,1} x_1 + \vartheta_{1,1} x_1^2 + \vartheta_{2,1} x_2 x_1 + \vartheta_{3,1} x_1^2 x_2 \\ \vartheta_{0,2} x_2 + \vartheta_{1,2} x_1 x_2 + \vartheta_{2,2} x_2^2 + \vartheta_{3,2} x_1 x_2^2 \end{bmatrix}. \end{aligned} \tag{4.15}$$

The reward function is defined in the form

$$\rho(x,u) = -10x^T x - 10u^T u \tag{4.16}$$

with $u = [u_1\ u_2]^T$. For the learning experiment the control input u is saturated via the function "sat" to verify $u_i \in [-0.5, 0.5]$ and the state is reshaped via "shape" from the range $[-5,5]^2$ to $[-1,1]^2$. Consequently, the resulting closed-loop system using the learned policy takes the form

$$\dot{x} = f\big(x, \mathrm{sat}\left(\hat{\pi}(\mathrm{shape}(x), \vartheta)\right)\big) = f\big(x, \bar{\pi}(x,\vartheta)\big). \tag{4.17}$$

Figure 4.26 illustrates the evolution of the PLTs algorithm applied to system (4.9). Each time a controller is learned and its DoA approximated, this information is stored. A new desired equilibrium is found by randomly generating samples within the desired range until expression (4.8) is verified.

Moreover, Figure 4.27 shows the search area for finding the new desired equilibria with respect to the logic expression (4.8). A state can be chosen from the state space if it lies in a δ-boundary of the union of all existing DoAs with $\delta = 0.9$.

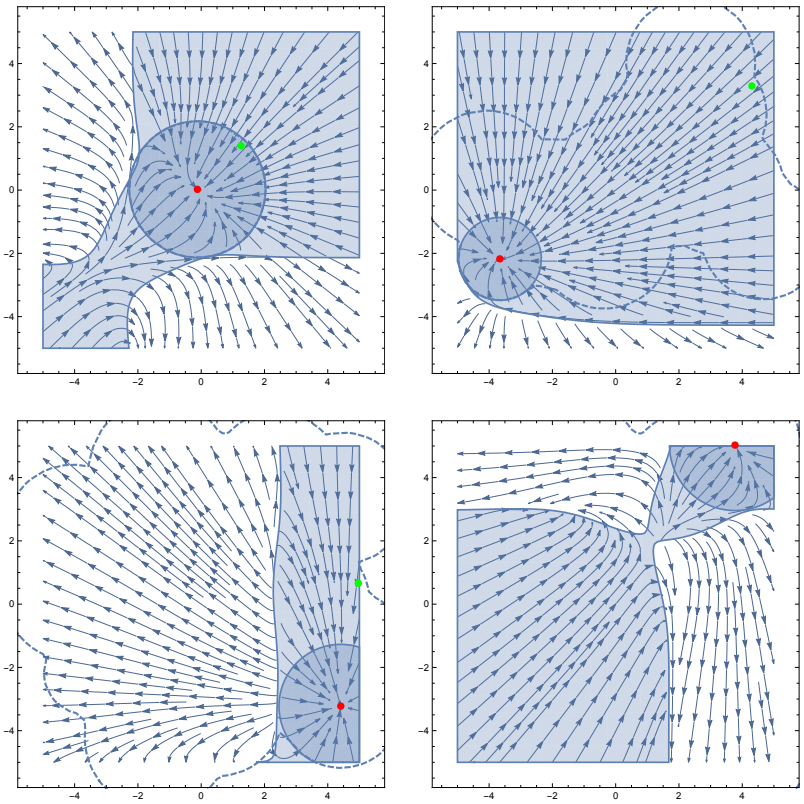

Figure 4.26: Snapshots of the evolution of the algorithm after iterations 1, 10, 20, and 32 (top left, top right, bottom left, bottom right, respectively). The arrows represent the trajectories of the resulting closed-loop system. The red dots are the learned actual stabilizing equilibria. The green dots are the locations of the upcoming desired equilibria for which new learning controllers are defined. The new equilibrium is computed to be close to the boundary of the union of all current DoAs, depicted by the dashed closed curve. The light blue shaded areas represents the sets defined by $\dot{L} \leq 0$ for each iteration and the dark blue shaded disks represent the approximated DoAs for each learned controller, i.e. the level set defined by $L(c_*)$.

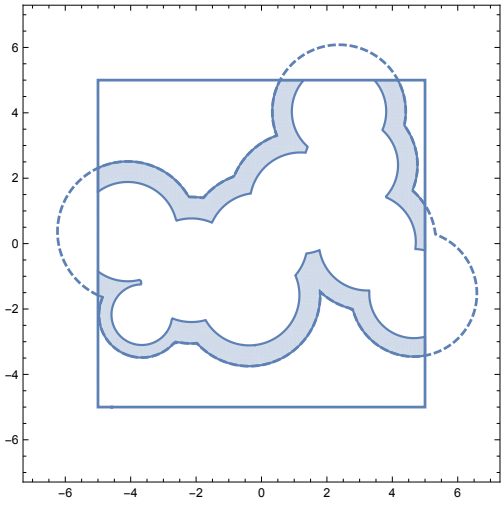

Figure 4.27: Search area (blue color area) for the next desired equilibrium in the boundary of the union of all existing DoAs (the dashed line) for iteration 10, illustrated at the top right plot of Figure 4.26. The function "FindNewDesiredEquilibrium" defined in Algorithm 8 uses this area for searching the new desired equilibrium.

Figure 4.28 illustrates the resulting DoAs for a complete execution of the PLTs algorithm. In our example, 32 controllers were learned. Figure 4.29 shows the induced control automaton. It can be shown that the first node, the stabilizing controller for the overall desired equilibrium, is reachable from all other nodes. This implies that this equilibrium is stabilizable from any state within the union of all DoAs.

Figure 4.30 presents the number of trials required to learn each controller. As expected these numbers depend on the local properties of the vector field f. In the areas where the magnitude of f is larger, more samples are required to learn a controller that fulfills the desired requirements. Figure 4.31 illustrates multiple continuous-time simulations of system (4.9) controlled by the PLTs algorithm. On the right side of the figure, the induced control automaton is overlapped to the trajectories in the state space to highlight the switching structure of the final controller. The supervisory controller precomputes the shortest path in the induced control automaton. It applies the sequence of stabilizing controllers switching instantly when entering the DoA of the next controller. Table 4.6 presents the simulation time for each step of the PLTs, implemented in the Mathematica software on an Intel core i7 2.7 GHz microprocessor.

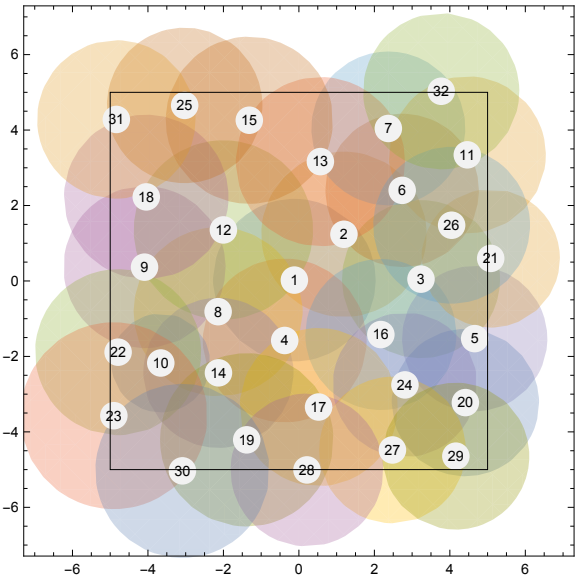

Figure 4.28: Resulting collection of the learned controllers using the PLTs algorithm. The stable equilibria are indicated by the numbers and each enclosing colored circle represents an estimate of the DoA of a learned controller. The union of all DoAs covers completely the desired operation range represented by a rectangle.

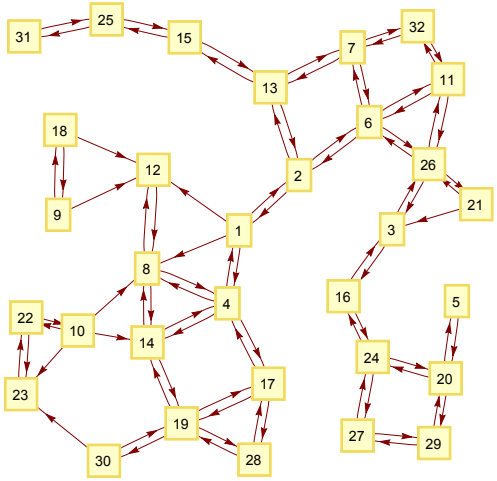

Figure 4.29: Induced control automaton generated using the PLTs algorithm. The node labeled "1" represents the stabilizing controller at the origin which is reachable from all other nodes.

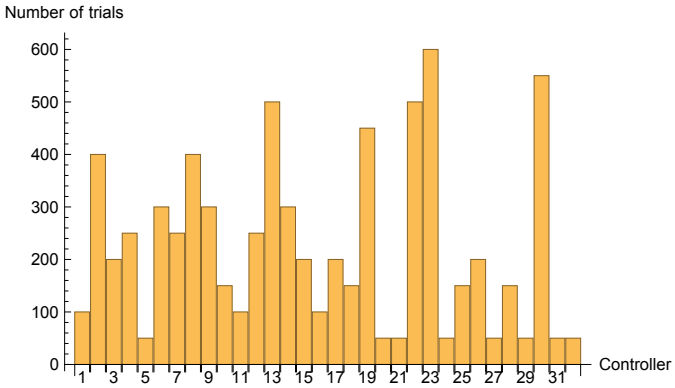

Figure 4.30: The number of trials required to learn each of the controllers. The controllers that require more learning trials, 13, 23, and 30, have equilibria in the neighborhood of the bumps on the potential function h illustrated in Figure 4.25. In these regions, the magnitude of the vector field f is larger.

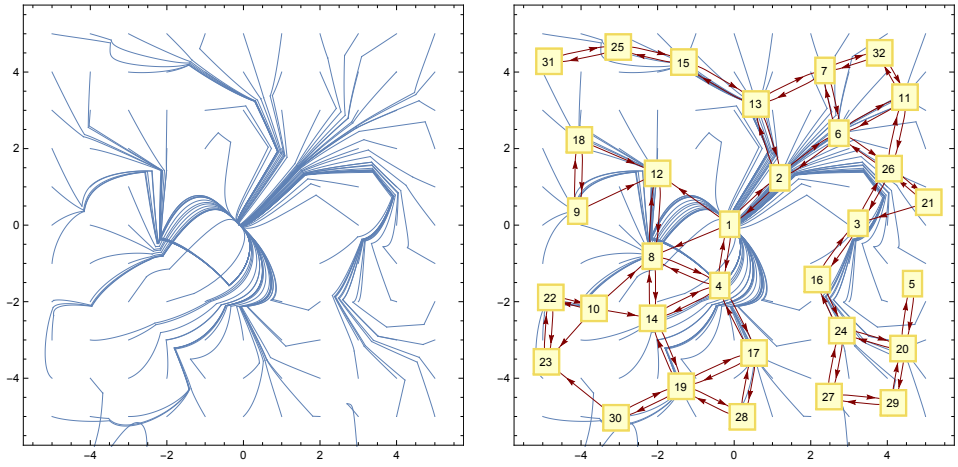

Figure 4.31: Multiple simulations of the resulting sequential composition controller with initial conditions in the range $[-5, 5] \times [-5, 5]$ with discrete intervals of 1. The plot on the right overlaps the trajectories from the left plot with the induced control automaton from Figure 4.29.

Table 4.6: Simulation time for each step of the PLTs algorithm

Function	Time
A-IDA-AC algorithm for 200 samples and 50 trials	~ 2 s
A-IDA-AC algorithm total time	~ 4.8 min
Estimation of the DoA using 5000 samples	~ 0.3 s
Estimation of the DoAs total time	~ 43 s
Finding a new equilibrium	~ 0.004 s
Finding new equilibria (total time)	~ 0.57 s
Total time	~ 5.5 min

4.6 Conclusions

A learning sequential composition control approach has been developed 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 pre-designed 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, with respect to the conventional learning methods, 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 chapter did not address the automatic choice of these parameters.

In addition, the proposed learning control approach has been extended to the situations where no controller is designed a priori, hence the supervisor needs to learn all controllers from scratch. The PLTs algorithm is developed that can learn a collection of controllers safely for a class of systems. The PLTs inherits the challenges present in RL, such as the curse of dimensionality, albeit with a better learning performance than the standard RL thanks to the use of prior knowledge in the form of a PH model. The complexity introduced by sequential composition and the approximation of the DoAs is relatively residual in comparison with the complexity of the learning process. However, by learning in small regions of the state space, as opposed to the entire state space at once, brings performance benefits. The PLTs introduces a trade off between optimality and time complexity.

5 CHAPTER

Cooperative Sequential Composition Control

Although sequential composition works properly for a single system, it is not designed for cooperative systems. This chapter extends the standard sequential composition by introducing a novel approach to compose multiple sequential composition controllers towards cooperative control. Given two or more systems, cooperation is achieved by composing each of the systems' control automaton, together with estimation of the DoAs of the resulting composed controllers. This typically results in new events for the original sequential composition controllers. Applying these events, the cooperative control system can fulfill the tasks which are not possible to satisfy with the original controllers individually. Each sequential composition controller works separately for its associated system until the control system requires the collaboration of multiple controllers. We present simulation results of an inverted pendulum system collaborating with two second-order DC motors for cooperative swing-up maneuvers.

5.1 Introduction

The standard sequential composition is typically designed for isolated systems, where there is no interaction between multiple systems (see e.g., [26, 91]). This chapter investigates how to exploit a collection of pre-designed sequential composition controllers so as to achieve a task cooperatively. The main goal is to accomplish collaborative behavior without having to redesign the low-level controllers. Given a particular interaction between two controlled systems, the effect of the

resulting interconnected system on the original controlled systems is computed for each combination of the available controllers.

In supervisory control systems, there are tasks that cannot be accomplished in isolation while they can be performed using multiple systems cooperatively [98]. For instance, assume the task of picking an object up from a table. This task may not be possible to attain using only one robotic arm due to the shape of its gripper, but it might be fulfilled if two robotic arms collaborate simultaneously. This is the main objective of cooperative control, i.e. enabling the coordination between multiple controlled systems [92].

Cooperative control has been investigated from various points of view, because of its numerous applications like multi-agent systems, cooperative manipulation, rescue mission, navigation and planning, formation control, etc. For example, various control techniques have been proposed for the cooperative control of multi-agent systems, which can broadly be classified into behavior-based approaches [8, 63], virtual structure algorithms [65, 9], leader-follower methods [120, 33], artificial potentials techniques [92], and graph theoretical methods [76, 37]. Moreover, some other work has been done to bridge the computer science symbolic approach with the continuous-time control's perspective, namely decentralized cooperative control approaches [61]. For instance, a cooperative multi-vehicle testbed is developed in [31] to facilitate the experimentation of cooperative control systems and mobile sensor networks. A decentralized cooperative control algorithm is proposed for controlling of heterogeneous vehicle groups in [119].

This chapter proposes a new approach in the field of cooperative control using the sequential composition paradigm that we call cooperative sequential composition control. It composes multiple sequential composition controllers to accomplish collaborative behavior on cooperative settings. The sequential composition controllers communicate with each other to share their corresponding system dynamics and low-level structures. Using this data along with the interaction dynamics enables computation of the DoAs of the resulting composed controllers. Based on the prepare relation, defined between the obtained DoAs and the goal sets, the original supervisors are augmented with new events through their low-level controllers. Applying these events, the cooperative control system can fulfill tasks which are not possible to accomplish with the original controllers individually. Each sequential composition controller works separately for its corresponding system until the control system needs the cooperation of multiple systems.

This chapter is organized as follows. Section 5.2 describes the cooperative sequential composition and its design. Section 5.3 investigates a case study of the collaboration between an inverted pendulum with two DC motors while simulation results are presented in Section 5.4. Finally, Section 5.5 concludes the chapter.

5.2 Cooperative Sequential Composition

The synthesis of cooperative supervisory control systems is studied using the paradigm of sequential composition. A cooperative sequential composition control algorithm is proposed to enable the coordination between a set of sequential composition controllers without any change in their low-level structures. It automatically constructs a cooperative sequential composition controller by operating on each control automaton and analyzing the interaction dynamics. This typically results in new events for the original sequential composition controllers. Using these events, the cooperative control system creates a new set of behaviors that are not achievable in isolation. The synthesis of the cooperative sequential composition controller follows three steps: composition, interaction, and cooperation.

5.2.1 Composition

The first phase towards enabling collaboration of the sequential composition controllers, designed for each controlled system individually, relies on composing the graph structure of their control automata. Assume there are multiple systems each endowed with its own independent control automaton. The dynamic equation of a system is given by

$$\dot{x}_i = f_i(x_i, u_i) \tag{5.1}$$

where $x_i \in \mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ is the state vector and $u_i \in \mathcal{U}_i \subseteq \mathbb{R}^{m_i}$ is the control input. Suppose $\Phi_i^p : \mathcal{X}_i \to \mathcal{U}_i$ represents the pth controller of system i, where the set of controllers is given by

$$\Phi_i = \left\{\Phi_i^1, \Phi_i^2, \ldots, \Phi_i^n\right\}. \tag{5.2}$$

Each mode $s_i^p \in \mathcal{S}_i$ in the control automaton is associated with controller Φ_i^p and defined by the tuple

$$s_i^p = \left\{\Phi_i^p, \mathcal{D}\left(\Phi_i^p\right), \mathcal{G}\left(\Phi_i^p\right)\right\} \tag{5.3}$$

where $\mathcal{S}_i$ is the finite set of modes in the control automaton. The Lyapunov function of system i, denoted by $L_i(x_i)$, is computed by applying the linearized equation of (5.1) in the Lyapunov equation.

The general form of the composed controller in the context of cooperative sequential composition control is defined as

$$\bar{\Phi}_{i,j,\ldots,r}^{p,q,\ldots,\ell} = \left\{\Phi_i^p, \Phi_j^q, \ldots, \Phi_r^\ell\right\}. \tag{5.4}$$

This composed controller simultaneously executes the pth controller of system i, the qth controller of system j, and so forth.

In the composition phase, a parallel composition is taken from the original control automata of the systems [17]. Consider the equations of motion for two controlled

independent dynamical systems described as

$$\begin{cases} \dot{x}_i = f_i(x_i, u_i) \\ \dot{x}_j = f_j(x_j, u_j). \end{cases} \tag{5.5}$$

When these systems are in collaboration, the structure of their control automata need to be shared by the systems to allow for computation of the composed controllers, their DoAs and goal sets. A framework is proposed to formalize the elements required for the collaboration. The standard definition of a hybrid automaton is adapted with the supervisory structure and cooperative settings to describe a cooperative control automaton. The new elements of this supervisory finite-state machine, in comparison with the standard hybrid automaton, are the composed controllers as well as the composed events.

Definition 5.1 *The cooperative control automaton defined in the context of cooperative sequential composition control is given by*

$$\bar{A} = A \cup \{\bar{E}\} \tag{5.6}$$

where A is the initial control automaton and $\bar{E} = \{\bar{e}_1, \bar{e}_2, \cdots, \bar{e}_s\}$ is a finite set of the composed events.

In the cooperative control automaton, we define individual events and composed events. The individual events are the original events defined in the pre-designed sequential composition controllers. The composed events are the result of the collaboration between the sequential composition controllers. When a composed event is triggered, the associated controllers in each system are executed simultaneously to accomplish the cooperative task. The composed event $\bar{e}_k \in \bar{E}$ is defined as $\bar{e}_k : \bar{\mathcal{S}}_{i,j} \rightarrow \bar{\mathcal{S}}_{i,j}$, with $\bar{\mathcal{S}}_{i,j}$ the finite set of modes in the cooperative control automaton given by

$$\bar{\mathcal{S}}_{i,j} = \mathcal{S}_i \times \mathcal{S}_j. \tag{5.7}$$

$\mathcal{S}_i$ and $\mathcal{S}_j$ represent the finite set of modes in system i and system j, respectively. Each composed mode $(s_i^p, s_j^q) \in \bar{\mathcal{S}}_{i,j}$ in the cooperative control automaton is the composition of the pth mode from system i and the qth mode from system j. The composed mode (s_i^p, s_j^q) is associated with the composed controller $\bar{\Phi}_{i,j}^{p,q}$ and is defined by the tuple

$$(s_i^p, s_j^q) = \left\{ \bar{\Phi}_{i,j}^{p,q}, \mathcal{D}\left(\bar{\Phi}_{i,j}^{p,q}\right), \mathcal{G}\left(\bar{\Phi}_{i,j}^{p,q}\right) \right\}. \tag{5.8}$$

Figure 5.1 illustrates the composition phase of the collaboration between two control automata, each with a set of pre-designed controllers. In the control automata, s_i^n and s_j^m denote the last modes of systems i and j, respectively.

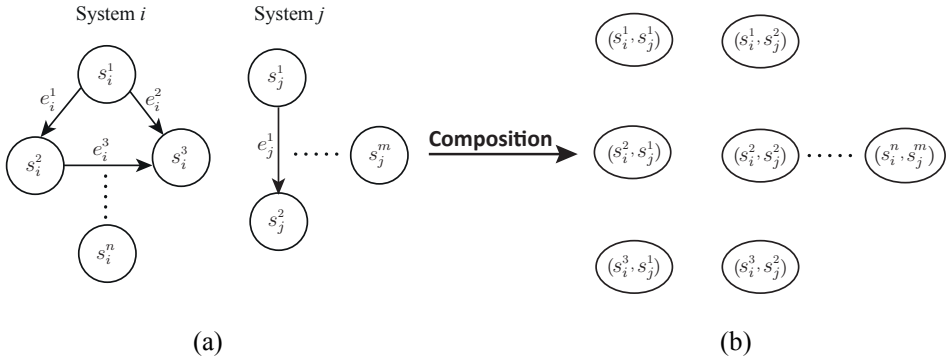

Figure 5.1: Composition phase of the collaboration between two dynamical systems: (a) before composition and (b) after composition. In this step the parallel composition of the control automata is computed.

5.2.2 Interaction

The second phase of the cooperative sequential composition is the interaction in which the dynamics of the interconnected systems are investigated. In this phase, the cooperative systems communicate with each other to share their dynamics as they physically interact. Modeling the interaction dynamics and computing the system dynamics is the most computationally intense step in the synthesis of the cooperative sequential composition control, because the DoAs of the composed controllers need to be computed. The information required to do these computations consists of the control automata of each system and the mathematical expressions of every system's model.

Suppose that the two dynamical systems described in (5.5) are interconnected via a physical interface such as a belt on two motors. The following set of equations represent a model of the interconnected system, here assuming that the interaction dynamics appear as the additional functions h_i and h_j to the model.

$$\begin{cases} \dot{x}_i = f_i(x_i, u_i) + h_i(\bar{x}) \\ \dot{x}_j = f_j(x_j, u_j) + h_j(\bar{x}) \end{cases} \tag{5.9}$$

where $\bar{x} \in \mathcal{X}_i \times \mathcal{X}_j \cdots \times \mathcal{X}_r$ is the composed state vector, with $\mathcal{X}_i$ the state space of system i, $\mathcal{X}_j$ the state space of system j, and so forth. Moreover, $h_i(\bar{x})$ and $h_j(\bar{x})$ illustrate the constraint equations of systems i and j, respectively. This is compatible with mechanical systems where physical interactions arise as the addition of constraint forces. Here, it is assumed that the constraint equations can be computed based on the prior knowledge on the systems' dynamics.

Once the interaction dynamic equations are obtained, one can approximate the

DoAs of the composed controllers, which are defined as

$$\bar{\Phi}_{i,j} \quad = \{\Phi_i^1\Phi_j^1, \Phi_i^1\Phi_j^2, \ldots, \Phi_i^n\Phi_j^m\}. \tag{5.10}$$

The Lyapunov function of the cooperative system, denoted by $\bar{L}(\bar{x})$, is calculated by using the linearized equations of (5.9), substituted by the composed controllers (5.10), in the Lyapunov equation. Given the Lyapunov function $\bar{L}(\bar{x})$, the DoAs of the composed controllers are computed with respect to their goal sets. The goal set of each composed controller is found by applying the controller on the cooperative system to make it stable. The resulting stable equilibrium is given as the goal set of the controller. Once all the DoAs are found, the cooperative control automaton is augmented with events that represent the prepare relations obtained by analyzing the composed dynamics.

Figure 5.2 represents the interaction phase of the coordination between two control automata. On one side, the new composed events are added to the cooperative control automaton by computing the DoAs and goal sets of the composed controllers. On the other side, some of the individual events defined in the original control automata might be discarded, due to the interaction dynamics and the composed controllers' DoAs.

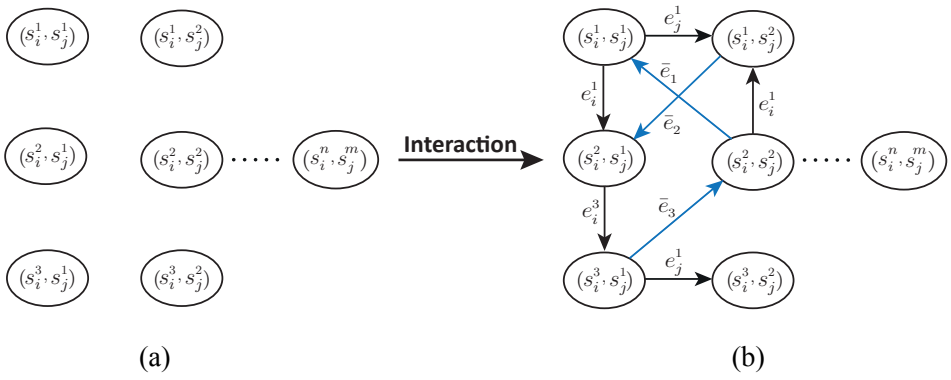

Figure 5.2: Interaction phase of the collaboration between two dynamical systems: (a) before interaction and (b) after interaction. In this step, the new composed events (blue arrows) are added to the cooperative control automaton by computing the DoAs and goal sets of the composed controllers while some individual events are discarded due to the interaction dynamics.

5.2.3 Cooperation

The third phase of the cooperative sequential composition control is the cooperation, in which each individual control automaton is augmented with the newly computed events resulting from the interaction dynamics. The composed events are projected on each control automata via the function "Proj". Given the cooperative control automaton $\bar{A}_{i,j}$ and the index of, for instance, system i, the projection function is defined as

$$\mathrm{Proj}(\bar{A}_{i,j}, i) = \bar{\mathcal{S}}_i \cup \bar{E}_i \tag{5.11}$$

where $\bar{\mathcal{S}}_i \subseteq \mathcal{S}_i$ is a subset of modes of system i. Each mode of $\bar{\mathcal{S}}_i$ is one of the pairs of a composed mode in the cooperative control automaton $\bar{A}_{i,j}$ for which a composed event is defined in $\bar{E}$. Moreover, $\bar{E}_i \subseteq \bar{E}$ is a set of composed events described between the composed modes corresponding to the modes of $\bar{\mathcal{S}}_i$. Thus, every composed event is projected on its corresponding individual control automaton. For instance, the composed events $\bar{E}_i$ are appended to the set of events in system i, i.e. $E_i' = E_i \cup \bar{E}_i$.

Figure 5.3 illustrates the cooperation phase of the collaboration between two control automata. The new composed events are appended to each individual control automaton. In the cooperation phase, each system has an augmented control automaton that consists of two types of events: individual events represented by black color and composed events represented by blue color in Figure 5.3(b). Algorithm 9 summarizes the synthesis of the cooperative sequential composition controller for two collaborating dynamical systems.

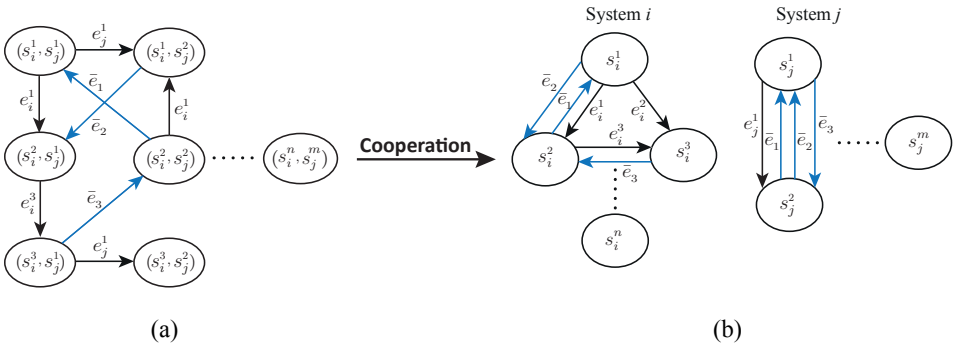

Figure 5.3: Cooperation phase of the collaboration between two dynamical systems: (a) before cooperation and (b) after cooperation. In this phase, the new composed events (blue arrows) are appended to each individual control automaton while the individual events (black arrows) exist a priori.

Algorithm 9 Synthesis of cooperative sequential composition control for two collaborating dynamical systems

Require: A_i, A_j, system (5.9)

Composition:
$(s_i^p, s_j^q) = s_i^p \times s_j^q$ for $\forall\, s_i^p \in \mathcal{S}_i$ and $\forall\, s_j^q \in \mathcal{S}_j$
Interaction:
For $\forall\, (s_i^p, s_j^q) \in \bar{\mathcal{S}}_{i,j}$ compute the interconnected dynamic equations (5.9) by replacing the composed controller $\bar{\Phi}_{i,j}^{p,q}$ and then estimate the DoA $\mathcal{D}(\bar{\Phi}_{i,j}^{p,q})$ based on Algorithm 5.
for $\forall\, (s_i^p, s_j^q) \in \bar{\mathcal{S}}_{i,j}$ **do**
  for $\forall\, \bar{\sigma} \in \bar{\Phi}_{i,j}$ / $\bar{\Phi}_{i,j}^{p,q}$ **do**
    if $\mathcal{G}(\bar{\Phi}_{i,j}^{p,q}) \subset \mathcal{D}(\bar{\sigma})$ **then** $\bar{\Phi}_{i,j}^{p,q} \succeq \bar{\sigma}$
    if $\mathcal{G}(\bar{\sigma}) \subset \mathcal{D}(\bar{\Phi}_{i,j}^{p,q})$ **then** $\bar{\sigma} \succeq \bar{\Phi}_{i,j}^{p,q}$
  end for
end for
Cooperation:
For $\forall\, \bar{e}_k \in \bar{E}$ project the composed event $\bar{e}_k$ on A_i and A_j using the projection rule (5.11).

5.3 Cooperation of the Inverted Pendulum with Two DC Motors

The proposed cooperative sequential composition approach is applied to the collaboration of an inverted pendulum with two DC motors. Consider the inverted pendulum with a DC motor, studied in Section 4.4.2, which is described by the nonlinear dynamic equation

$$J\ddot{q}_1 = mgl\sin(q_1) - \left(b + \frac{K^2}{R}\right)\dot{q}_1 + \frac{K}{R}u_1 \tag{5.12}$$

where the state vector is defined by $x_1 = [q_1\ \dot{q}_1]^T$, with q_1 the angle of the pendulum measured from the upright position and $\dot{q}_1$ the angular velocity. Here, the pendulum mass is $m = 0.03\,\text{kg}$. The control system consists of two controllers Φ_{UP} and Φ_{DOWN} to stabilize the pendulum at the up and down equilibria, respectively. These controllers are represented by modes "UP" and "DOWN" in the control automaton, as shown in Figure 5.4(b). Moreover, event D connects mode UP to DOWN and switches the control system from controller Φ_{UP} to Φ_{DOWN}. The control objective is to stabilize the pendulum at the up equilibrium starting from any initial state within the union of the existing DoAs.

Another DC motor, with the same configuration, is connected to the motor of the inverted pendulum by a belt, as schematically shown in Figure 5.4(a). The

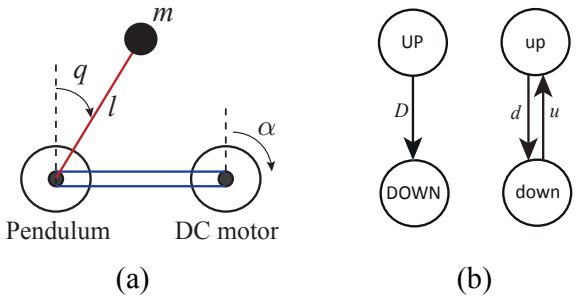

Figure 5.4: (a) Schematic representation of the inverted pendulum collaborating with two DC motors. (b) Individual control automata.

dynamic equation of the DC motor is described by

$$J\ddot{q}_2 = -\left(b + \frac{K^2}{R}\right)\dot{q}_2 + \frac{K}{R}u_2 \tag{5.13}$$

where the state vector is given by $x_2 = [q_2\ \dot{q}_2]^T$, with q_2 the motor angle measured from the upright position and $\dot{q}_2$ the angular velocity. The DC motor is controlled by a sequential composition controller similar to the control system of the inverted pendulum. It consists of two individual controllers Φ_{up} and Φ_{down} to drive the initial bottom point of the motor to the up and down positions, respectively. These controllers are denoted by "up" and "down" in the control automaton, as shown in Figure 5.4(b). Event u connects mode down to up and drives the motor to the up position. Inversely, event d connects mode up to down and steers the motor to the down position.

The first step of this cooperation is the composition that provides composed controllers, represented by the composed modes in the cooperative control automaton. Since in this example two systems are collaborating, the composed controllers are in pairs. If the first sequential composition controller has r_1 controllers and the second has r_2 controllers, there will be $r_1 \times r_2$ composed controllers in the composition phase. While a composed controller is executed, the first individual controller is activated in the first system and the second individual controller is triggered in the second system, simultaneously. Figure 5.5 represents the composition phase for the inverted pendulum collaborating with two DC motors.

The second step is the interaction, where the systems physically interact with each other to achieve the cooperative control objectives. The interconnected system is described by the dynamic equations

$$\begin{cases} J\ddot{q}_1 = mgl\sin(q_1) - \left(b + \frac{K^2}{R}\right)\dot{q}_1 + \frac{K}{R}u_1 + h_1 \\ J\ddot{q}_2 = -\left(b + \frac{K^2}{R}\right)\dot{q}_2 + \frac{K}{R}u_2 + h_2. \end{cases} \tag{5.14}$$

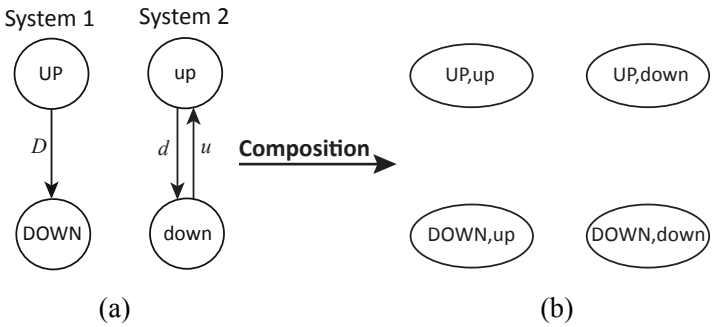

Figure 5.5: Composition phase of the collaboration of the inverted pendulum with two DC motors: (a) before composition and (b) after composition.

The composed state vector of the cooperative system is described by

$$\bar{x} = [q_1 \;\; \dot{q}_1 \;\; q_2 \;\; \dot{q}_2]^T \tag{5.15}$$

and the composed controllers are given by

$$\bar{\Phi} = \{\bar{\Phi}_{\mathrm{UP,up}}, \bar{\Phi}_{\mathrm{UP,down}}, \bar{\Phi}_{\mathrm{DOWN,up}}, \bar{\Phi}_{\mathrm{DOWN,down}}\}. \tag{5.16}$$

To compute the constraint equations h_1 and h_2, which are the external torques that each system applies on the another system, the connecting belt has to be modeled as it exerts the torques to the DC motors. From a human expert approach, creating the interaction model is simple, but it can be very difficult from the perspective of an autonomous robot. To simplify the equations, it is assumed that the connecting belt does not slip on the motors' shafts, hence $h_1 = -h_2$. The belt is modeled by a spring-damper system as schematically shown in Figure 5.6.

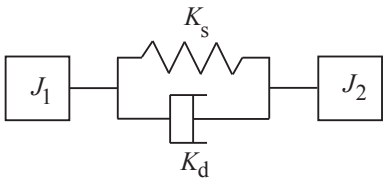

Figure 5.6: Modeling the connected belt by a spring-damper system.

The dynamic equations of this spring-damper model are defined by

$$\begin{cases} h_1(\bar{x}) = -K_\mathrm{s}(q_1 - q_2) - K_\mathrm{d}(\dot{q}_1 - \dot{q}_2) \\ h_2(\bar{x}) = -K_\mathrm{s}(q_2 - q_1) - K_\mathrm{d}(\dot{q}_2 - \dot{q}_1). \end{cases} \tag{5.17}$$

When controllers Φ_{UP} and Φ_{up} are active in the first and second systems respectively (i.e., the composed controller $\bar{\Phi}_{\mathrm{UP,up}}$ is triggered), the first-order dynamic

equations of the interconnected system are described by

$$\begin{cases} \dot{\bar{x}}_1 &= \bar{x}_2 \\ \dot{\bar{x}}_2 &= mgl\sin(\bar{x}_1) - \left(b + \frac{K^2}{R}\right)\bar{x}_2 + \frac{K}{R}\Phi_{\mathrm{UP}} - K_{\mathrm{s}}(\bar{x}_1 - \bar{x}_3) - K_{\mathrm{d}}(\bar{x}_2 - \bar{x}_4) \\ \dot{\bar{x}}_3 &= \bar{x}_4 \\ \dot{\bar{x}}_4 &= -\left(b + \frac{K^2}{R}\right)\bar{x}_4 + \frac{K}{R}\Phi_{\mathrm{up}} - K_{\mathrm{s}}(\bar{x}_3 - \bar{x}_1) - K_{\mathrm{d}}(\bar{x}_4 - \bar{x}_2). \end{cases} \tag{5.18}$$

The DoA of each composed controller and its computation are of main importance at this step. The sampling method is used to approximate the DoAs [84]. The Lyapunov function of the cooperative system is computed by using the linearized equations of (5.18) in the Lyapunov equation. Figure 5.7 shows the DoAs approximated for the composed controllers $\bar{\Phi}_{\mathrm{UP,up}}$, $\bar{\Phi}_{\mathrm{UP,down}}$, $\bar{\Phi}_{\mathrm{DOWN,up}}$, and $\bar{\Phi}_{\mathrm{DOWN,down}}$ projected on the plane $\bar{x}_3 = \bar{x}_4 = 0$. For this specific situation, the projected DoAs $\mathcal{D}(\bar{\Phi}_{\mathrm{UP,down}}) = \mathcal{D}(\bar{\Phi}_{\mathrm{DOWN,up}}) = \emptyset$. Moreover, $\mathcal{G}(\bar{\Phi}_{\mathrm{UP,up}}) \in \mathcal{D}(\bar{\Phi}_{\mathrm{DOWN,down}})$ and $\mathcal{G}(\bar{\Phi}_{\mathrm{DOWN,down}}) \in \mathcal{D}(\bar{\Phi}_{\mathrm{UP,up}})$, meaning that controller $\bar{\Phi}_{\mathrm{UP,up}}$ prepares controller $\bar{\Phi}_{\mathrm{DOWN,down}}$ and vice versa. Figure 5.8 illustrates the induced cooperative control automaton of the interaction phase, which consists of the composed modes and composed events.

The third step is the cooperation, in which every sequential composition controller designed for the individual systems are augmented by the new connections computed from the interaction dynamics. Figure 5.9 illustrates the cooperation phase, where the the composed events are projected on their associated control automata.

5.4 Simulation Results

The cooperative sequential composition control technique has been implemented on an inverted pendulum collaborating with two DC motors, as shown in Figure 5.10. The axis of the pendulum's DC motor is connected to the axis of the another DC motor via a flexible belt. This connecting belt is modeled by a spring-damper model as schematically illustrated in Figure 5.6.

The control objective is that the control system can switch from controller Φ_{UP} to Φ_{DOWN} and vice versa. Following the three steps composition, interaction, and cooperation consecutively, a cooperative sequential composition controller is synthesized that can switch between the pre-designed controllers. Simulation results of this cooperation are illustrated in Figure 5.11, representing the pendulum's angle and angular velocity.

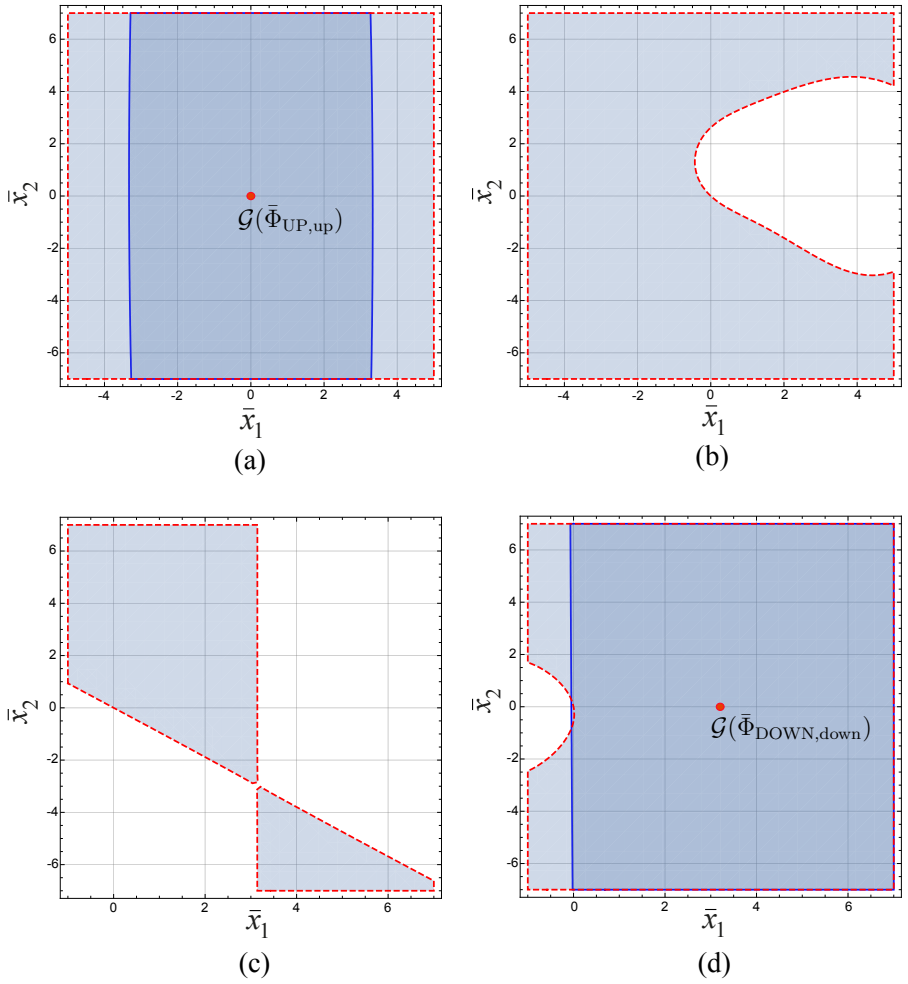

Figure 5.7: Projections of the DoAs approximated for the composed controllers (a) $\bar{\Phi}_{\mathrm{UP,up}}$, (b) $\bar{\Phi}_{\mathrm{UP,down}}$, (c) $\bar{\Phi}_{\mathrm{DOWN,up}}$, and (d) $\bar{\Phi}_{\mathrm{DOWN,down}}$ on the plane $\bar{x}_3 = \bar{x}_4 = 0$. The blue line (boundary of the dark blue area) represents the estimated DoAs and the dashed red line (boundary of the light blue area) illustrates the region in which $\dot{\bar{L}}(\bar{x}) < 0$. The projected DoAs $\mathcal{D}(\bar{\Phi}_{\mathrm{UP,down}}) = \mathcal{D}(\bar{\Phi}_{\mathrm{DOWN,up}}) = \emptyset$. The projected goal sets $\mathcal{G}(\bar{\Phi}_{\mathrm{UP,up}})$ and $\mathcal{G}(\bar{\Phi}_{\mathrm{DOWN,down}})$ are the points $\bar{x} = [0\ 0\ 0\ 0]^T$ and $\bar{x} = [\pi\ 0\ 0\ 0]^T$, respectively. Since $\mathcal{G}(\bar{\Phi}_{\mathrm{UP,up}}) \in \mathcal{D}(\bar{\Phi}_{\mathrm{DOWN,down}})$ and $\mathcal{G}(\bar{\Phi}_{\mathrm{DOWN,down}}) \in \mathcal{D}(\bar{\Phi}_{\mathrm{UP,up}})$, $\bar{\Phi}_{\mathrm{UP,up}} \succeq \bar{\Phi}_{\mathrm{DOWN,down}}$ and inversely $\bar{\Phi}_{\mathrm{DOWN,down}} \succeq \bar{\Phi}_{\mathrm{UP,up}}$.

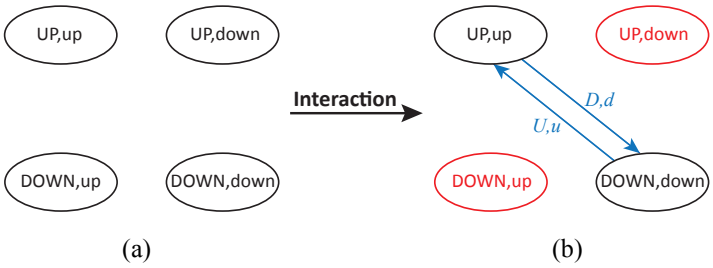

Figure 5.8: Interaction phase of the collaboration of the inverted pendulum with two DC motors: (a) before interaction and (b) after interaction, where the new composed events (blue arrows) are added to the cooperative control automaton.

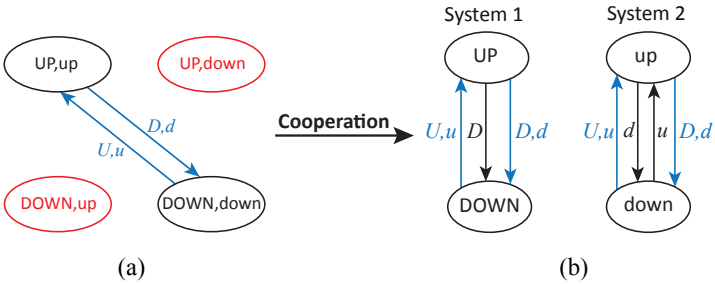

Figure 5.9: Cooperation phase of the collaboration of the inverted pendulum with two DC motors: (a) before cooperation and (b) after cooperation, where the new composed events (blue arrows) are appended to each individual control automaton.

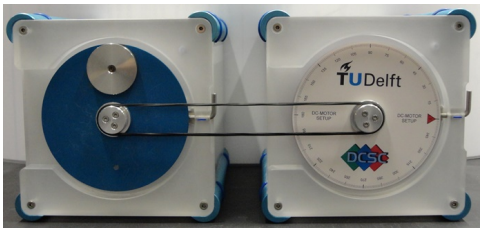

Figure 5.10: Inverted pendulum collaborating with two DC motors. The axis of the pendulum's DC motor is connected to the axis of the another DC motor via a flexible belt.

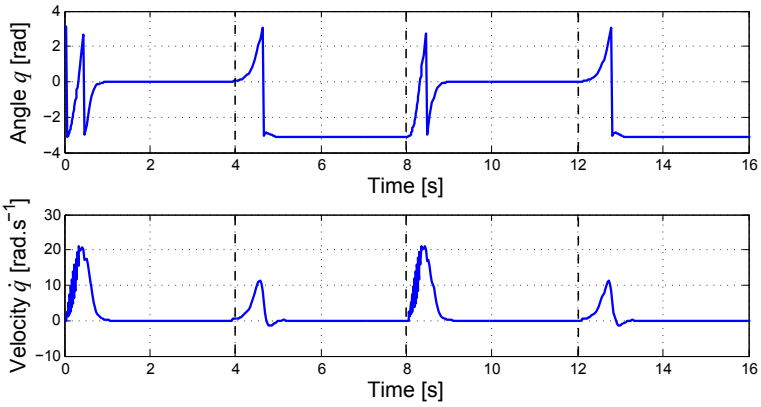

Figure 5.11: Simulation results of the inverted pendulum collaborating with two DC motors during two iterations of the cooperative swing-up maneuver. The top plot illustrates the pendulum angle at each iteration started from $\pm\pi$ rad (pendulum is at the down equilibrium) and ended at 0 rad (pendulum is stabilized at the up equilibrium). The jumps from π to $-\pi$ is due to the fact that both angles represent the down equilibrium. Rotation of the pendulum on a circle starts from π and ends at $-\pi$. The bottom plot represents the pendulum angular velocity at each iteration. The velocity is 0 while the pendulum is stable either at the up or at the down equilibria.

5.5 Conclusions

A cooperative sequential composition control approach is developed that systematically enables the composition of two independent supervisory controllers to achieve new tasks that were not possible by each controller individually. This approach currently relies on mathematical models for all the dynamics present on the system together with an expression for the constraints introduced by the interaction. If all of this information is available, then it is feasible to apply the composition rules to obtain a new cooperative control automaton for cooperative systems. Given two or more systems that require to interact, the computational process needs to be executed in at least one of the systems. They share their control automata along with the mathematical expressions of their models. The computation is implemented in one of the systems and the results, in the form of extra composed events in the cooperative control automaton, are shared with other systems. Finally, each system receives only the relevant composed events for its own control automaton. Simulation results presented in this chapter illustrate that for a class of known models cooperation can be automated. Future research includes using partial dynamical models where learning needs to happen in real-time towards achieving a new interaction.

6 CHAPTER

Robot Contact Language

This chapter studies the synthesis of supervisory control systems for robotic planning and manipulation. The problem of dividing a manipulation task is addressed to obtain an appropriate sequence of subtasks with regard to the contact-based task division. A robot contact language is defined for robotic manipulation based on making and breaking contact between the components involved, namely robots, objects, and surfaces. This planner is modular enough to deploy geometrical and physical information of the components and translate supervisory planning to low-level robot controllers. This robot language is validated in three case studies, each with a specific control objective.

6.1 Introduction

The supervisory architecture defined in the earlier chapters is applied to a robotic language, designed for the manipulation of multiple objects by multiple robots. This chapter specifically addresses mechanical systems and describes a supervisory control system in the form of a contact language. It proposes a new approach for robotic manipulation planning in terms of the contact between the involved components, particularly: robots, objects, and surfaces. The dynamics of a manipulation change when contact between these components are made or broken. Using this paradigm, the robotic manipulation planner is developed. A supervisory structure is generated using a set of derived rules and the available geometrical information.

The problem of intelligently dividing a manipulation task into a correct sequence of sub-tasks is addressed in this chapter with contact-based task division being the primary paradigm behind the research. The novelty of this proposed planner

is its ability to plan a manipulation on an abstract level as well as be modular enough to involve geometrical and physical information of the scene components and easily translate high-level planning to low-level robot control.

Various generalized planners exist for robotic motion planning, artificial intelligent (AI) planning and task planning. The aSyMov planner [43] combines symbolic planning with probabilistic road map and is based on a forward heuristic search. This planner relates closely to the one proposed in this chapter but is not based on the notion of contact. A symbolic and geometrical planner is proposed in [53], which uses aggressive hierarchy. Moreover, a manipulation planner is described in [32] that uses a learned mapping between geometric states and logical predicates. It builds symbolic representations from the scenes a robot observes and translates them into geometric states that could further be used to carry out manipulations.

There have been dedicated languages developed for generic AI planning, the most widely used being the planning domain definition language (PDDL) [41] inspired by an older planner and scheduler, the stanford research institute problem solver (STRIPS) [38]. This language consists of objects and predicates along with initial and goal states for a task. Actions or operations can be performed to change the state of the world. Several manipulation planning algorithms have been developed within the PDDL framework (see e.g., [34, 16]). Graphplan [12] is another planner that uses specialized graphs and STRIPS inspired planning. It uses the STRIPS based formulation for planning and formulates the problem into special types of graphs, called planning graphs. These graphs consist of a problem formulated via fact and action nodes along with special operator nodes. Standard graph traversal techniques can be applied on these graph for generic AI planning. The planner has been proven to be effective in reducing the planning search compared to other planners.

For planning manipulations among movable objects an algorithm is developed in [114] that uses reverse-time search to samples future robot actions and constraints the space of prior object placements. It checks for objects blocking the path of the primary object to be manipulated and recursively, all blocking objects are displaced to make way for the primary manipulation. The work has been further improved by extending the algorithm's ability to deal with artificial constraints during planning [113].

High-level planning and control is either followed or complemented by low-level planning and control. For the specific case of robotic manipulators, this involves end-effector path planning and executing control algorithms to track the desired paths or trajectories. Various path planning algorithms have been successfully used in real-time for path planning in joint space or workspace. The most commonly used path planning algorithm is the rapidly exploring random trees (RRTs) [62] and its variants [60, 130], which are efficient in exploring higher dimensional and constrained spaces.

Manipulating tasks require interaction of robot's end-effector with the environment. To track the desired paths and manipulate objects, various implicit or explicit force based controllers are used to regulate this interaction and make the robot compliant. Hybrid position-force control [30, 73] is a direct force control technique where position and force feedback are applied in orthogonal workspace, depending on the task. Another widely used indirect force control technique is impedance control [50, 51], wherein the robot end-effector behaves as a virtual mass-spring-damper system. Interaction with the environment can be varied by altering the spring parameters. Further extensions to impedance controllers can be found in [115, 95, 15]. Cooperative manipulation control is a modified extension to the previous two compliance control techniques while adhering to the closed-chain constraints [133]. The control techniques are based on hybrid-position and force control [127, 100] or impedance control [14, 96].

None of the manipulation planning algorithms and techniques mentioned above addresses manipulation planning using a purely contact-based approach. The high-level AI planners and languages focus on higher abstraction of a given task. They are generic and do not consider low-level control and assume the robot can perform the planned task. On the other hand, sequential composition has more focus on low-level control and dynamics of a system but is incapable of incorporating high-level intelligence for task or manipulation planning.

It is known that the dynamics of a manipulation task change when the contact between a robot and the object it is manipulating changes. When a contact is made or broken by an object with other robots, objects or surfaces, it is practical and intuitive to assign a new controller for the different state of contact or contact configuration between these entities. Given the initial and goal contact configuration of the objects to be manipulated, the proposed planner first generates a symbolic contact graph via node expansion and exploration from the initial to the goal node based on a set of pre-defined rules. With the available geometrical information, the graph is enhanced and weights are assigned to the graph edges. A high-level manipulation planning is defined as finding a path on this contact graph which can be carried out using standard graph-traversal algorithms. The shortest path lists the relevant nodes or contact configurations in the correct sequence and the robot needs to successfully carry out the manipulation. The low-level planning can then be done for each relevant contact configuration for task execution.

This chapter is organized as follows. Section 6.2 formulates the nomenclature, mathematical notations and rules for the proposed symbolic planner. Section 6.3 describes the process of planning a manipulation task using the derived rules followed by low-level planning and control. Then, Section 6.4 presents three simulation test cases for validation of the proposed planner. Finally, Section 6.5 presents a brief discussion and concludes the chapter with final comments.

6.2 Robot Contact Language

In this chapter, we consider that the components in a general manipulation scene can be divided into three broad categories, namely robots, objects and surfaces. These can be represented as follows

$$\begin{aligned} R &= \{R_1, R_2, \ldots, R_m\} \\ O &= \{O_1, O_2, \ldots, O_n\} \\ S &= \{S_1, S_2, \ldots, S_q\} \end{aligned} \tag{6.1}$$

where R, O, and S are the set of robots, objects and surfaces present in a manipulation scene with m, n and q being their indices, respectively.

Robots are components that can carry out manipulations. Here, we assume robots refer to robotic manipulators. Each dexterous robot is associated with a workspace. If a robot is in contact with an object, the object can be manipulated throughout the robot's workspace. A robot can also be associated with a set of different controllers or a specific controller with varying parameters (as in the case of this chapter), which can be set as per the task requirements. Objects are referred to all movable entities in a scene that can be manipulated by the available robots whereas surfaces are static entities on which objects can rest in a stable position. A complete knowledge of all these components: robots, objects, and surfaces is assumed to be known to the planner.

A contact between two or more components is represented by a contact pair as

$$O_i(O_j|S_k|R_l) \tag{6.2}$$

where $i \neq j$. Objects can be in contact with another object O_j, a robot R_l, or a surface S_k. The symbol "|" represents the "or" operation. For example, O_1R_1 represents contact of O_1 with R_1. It is wise to mention here that contact pairs of the types R_iR_j, R_iS_j and S_iS_j are invalid. All possible types of contact pairs are hence given as follows

- O_iR_j: object in contact with a robot
- O_iS_j: object in contact with a surface
- O_iO_j, $i \neq j$: object in contact with an object

Two or more contact pairs can be represented using the logical "and" operator, illustrated by the symbol "$\wedge$" as follows

$$N_q = O_i(O_j|S_k|R_l) \wedge O_w(O_r|S_s|R_v) \wedge \ldots \tag{6.3}$$

with $i \neq j$ and $w \neq r$. This represents an object in contact with multiple entities or two different objects in contact with one or more components. For instance, $O_1R_1 \wedge O_1S_1 \wedge O_2R_2$ represents that O_1 is in contact with R_1 and S_1 along with O_2 is in contact with R_2. A contact configuration can thus be defined as a set of one or more contact pairs, given by (6.2) and (6.3), that represent contact between the components in a scene. Let the set of all valid contact configurations be given by

$$\mathcal{N} = \{N_1, N_2 \dots N_p\} \tag{6.4}$$

where N_i represents the ith configuration with p being the total number of valid configurations. Each contact configuration is illustrated by a node on the contact graph.

6.2.1 Assumptions

For the purpose of simplification and abstraction, various assumptions have been made for definition of the manipulation planner. These assumptions are listed as follows.

- A robot can only manipulate or be in contact with one object at a time.
- There is a physical limit to the number of robots that can be in contact with a particular object.
- The number of components (robots, objects, and surfaces) in the scene along with their necessary geometrical information is known and accessible to the planner.
- No manipulation can occur if no robot is in contact with an object. Such a contact pair is called a static contact pair.
- An object is always statically stable when not being manipulated.
- An object cannot be in contact with two surfaces simultaneously.
- A robot making contact with surfaces or other robots are not of interest and such pairs are omitted, i.e. $R_iS_j \notin \mathcal{N}$ and $R_iR_j \notin \mathcal{N}$.
- Two surfaces in contact are not considered and such pairs are omitted, i.e. $S_iS_j \notin \mathcal{N}$.

6.2.2 Language Rules

The initial and goal positions of the objects manipulated lie in a certain contact configuration that can be represented by the pre-described nomenclature. Every

contact pair represents the initial and goal nodes in the contact graph that is generated to plan the manipulation task.

In order to build the contact graph, certain rules need to be defined for generating a set of nodes representing contact configurations. These rules are based on the robots, objects, and surfaces making and breaking contact between one another. Using these rules, a set of nodes is generated. Starting from the initial node and exploring the graph further, a path from the initial to goal node can be found. There are five basic rules postulated for this planner. The first four are based on change in contact and the last one is meant for parallelizing tasks.

Robot Making Contact

This rule is for a robot making contact with an object. The object could be either resting on a surface or an object, or could be held by another robot before the robot makes contact, described by

$$O_i(O_k|S_l|R_m) \wedge \mathcal{Q} \rightarrow R_j O_i \wedge O_i(O_k|S_l|R_m) \wedge \mathcal{Q} \tag{6.5}$$

where $\mathcal{Q}$ represents the set of other contact pairs present in the node, $R_j \notin \mathcal{Q}$ (i.e., robot R_j is not already in contact with other objects), $i \neq k$, and $j \neq m$. Here, $(O_k|S_l|R_m)$ represents another object, robot, or a surface in contact with the object that is being made contact with and $\mathcal{Q}$ represents the set of other contact pairs present in the node.

Robot Breaking Contact

A robot can drop an object on a surface, another object, or release an object that is being held by another robot, defined by

$$R_j O_i \wedge O_i(O_k|S_l|R_m) \wedge \mathcal{Q} \rightarrow O_i(O_k|S_l|R_m) \wedge \mathcal{Q} \tag{6.6}$$

with $R_j \notin \mathcal{Q}$, $i \neq k$, $j \neq m$. If the object is placed on top of another object or a collection of objects, a check is performed to make sure that these objects are statically laying on a surface, thus avoiding "floating" groups of objects. This check is formalized by the rule

$$R_j O_i \wedge O_i O_k \wedge \mathcal{Q} \rightarrow O_i O_k \wedge \mathcal{Q} \tag{6.7}$$

if there exists a sequence

$$O_i O_k \wedge \cdots \wedge O_l O_r \wedge O_r S_w \in \mathcal{Q} \tag{6.8}$$

that establishes contact with a static surface, with $i \neq k \neq l \neq r$.

Robots Placing Object

One or more robots can place an object on only one surface or object in one transition. A prerequisite for placing an object is that at least one robot needs to be in contact with that object to carry out that task (manipulating the object). This rule is illustrated by

$$\mathcal{P} \wedge \mathcal{Q} \leftrightarrow \mathcal{P} \wedge O_i(O_l|S_m) \wedge \mathcal{Q} \tag{6.9}$$

such that

$$\mathcal{P} = R_j O_i \wedge \cdots \wedge R_k O_i. \tag{6.10}$$

In this rule, $\mathcal{P}$ is the set contact pairs that represent the contact of all the robots with the object being placed with components other than the one it is lifted from or placed at. Here, $R_j, \ldots, R_k \notin \mathcal{Q}$ and $\mathcal{P} \nsubseteq \mathcal{Q}$ along with $i \neq l$ and $j \neq k$. Moreover, $O_i O_l \notin \mathcal{Q}$, i.e. O_i is not already in contact with O_l.

Robots Lifting Object

One or more robots that are in contact with an object or a surface can lift that object to remove contact. Similar to placing objects, a prerequisite for lifting an object is that at least one robot needs to be in contact with that object, represented as

$$\mathcal{P} \wedge O_i(O_l|S_m) \wedge \mathcal{Q} \leftrightarrow \mathcal{P} \wedge \mathcal{Q}. \tag{6.11}$$

The conditions of robots placing object also apply for lifting objects. Moreover, the robots in contact can lift the object only if it is not resting on more than one surface or object, that is, $O_i O_n \notin \mathcal{Q}$ with $n \neq i, l$. Similar to the rule of making and breaking contact, the relationship from place object to lift object, that is, from (6.9) to (6.11) is injective and non-surjective in nature.

Parallelization of tasks

When a manipulation task is being performed, it is possible that one or more subtasks can be carried out simultaneously. On the contact graph, these branches could be broken down into parallel manipulation branches. This is done by the following parallelization rule

$$X_i X_j \wedge X_k X_l \leftrightarrow (X_i X_j) \parallel (X_k X_l) \tag{6.12}$$

such that

$$\begin{aligned} &X \in \{R, O\} \rightarrow i \neq j \neq k \neq l \\ &X \in S \end{aligned} \tag{6.13}$$

and operator "$\|$" denotes parallelization. If X is either a robot or an object, all components in the contact pairs need to be different for parallelization to occur. However, it is possible for X to be a common surface. This is because multiple objects can be simultaneously manipulated by different robots on the same surface at different positions.

6.3 Manipulation Planning and Control

Planning a manipulation task starts with the given initial and goal nodes. Using these two nodes, new nodes are created using the rules, described in Section 6.2, from the initial node, leading to graph expansion towards finding a path that connects the initial configuration to the goal. From the available geometrical and other relevant information, the contact graph is pruned. Moreover, parallelization can be done for tasks that are not dependent on each other. These steps are explained as follows.

6.3.1 Contact Graph Generation

Using the rules described in Section 6.2.2 and assumptions in Section 6.2.1, nodes are further expanded with the aim to find an edge in the direction of the goal node. It is possible to generate all the conceivable nodes at once, but the total number of nodes grow exponentially with the number of components present in the scene. This may lead to memory problems and even the possibility of larger search times. To counter this problem, nodes are generated online using a suitable heuristic to reduce the required memory and quickly converge to the goal node. A search can result in one or multiple paths on the contact graph. These paths are further analyzed with the help of geometrical information.

Consider an example with a robot R_1, an object O_1, and two surface S_1 and S_2. Let the task be to pick O_1 from S_1 and place it on S_2 using the robot. This is shown in Figure 6.1. The initial node is O_1S_1 and the goal node is O_1S_2. One can first

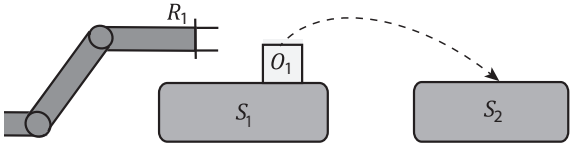

Figure 6.1: Schematic representation of a scene with a robot R_1, an object O_1, and two surface S_1 and S_2. The task is to pick O_1 from S_1 and place it on S_2 using R_1, i.e. the initial configuration is O_1S_1 and the goal configuration is O_1S_2.

apply the rule of robot making contact with an object followed by robot lifting object. This will be followed by the robot placing the object on the other surface and finally, releasing contact. These nodes are illustrated in Figure 6.2.

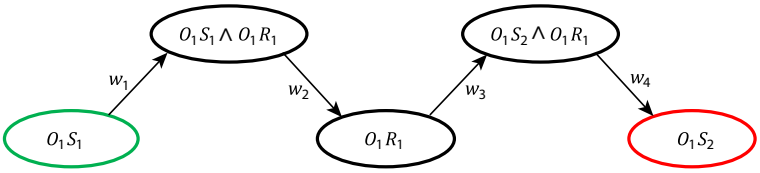

Figure 6.2: Contact graph describing node expansion from the initial to goal node using the set of contact language rules. The green node is the initial node and the red one is the goal node for the task of picking and placing O_1 from S_1 to S_2 using the robot R_1. The weights on the edges are given by w_i for $i = 1$ to 4.

This is the basic process of manipulation planning which is carried out by automatically applying the contact language rules. It is important to note that in each transition, only one component is making or breaking contact. This is an inherent property of the derived rules. The transition between two nodes represents edges on the contact graph. An edge between the ith and jth node is given by $E_{i,j}$. The set of all possible edges is called $\mathcal{E}$.

6.3.2 Geometrical and Physical Constraints

It is quite evident that not all possible paths found using the initial planning are always possible to achieve. There are certain geometrical and physical constraints that render certain edges and nodes of the contact graph irrelevant or unattainable. Using the available information, these constraints are identified and the respective nodes and edges are removed from the graph. A robot can not manipulate an object if it is not within its reachable workspace. Moreover, two or more robots cannot cooperatively manipulate an object if there exists no overlapping workspace for the involved robots.

Geometrical information of all components can be represented as symbolic inequalities using generalized equations of surfaces and volumes. Let $\mathcal{W}_i^X$ represent the associated workspace of X_i with $X \in \{R, O, S\}$. For robots, this could be the dexterous workspace whereas for objects and surfaces it could be the stable contact areas or other contact points. For instance, the workspace of a planar robot R_1 with revolute joints about axis x can be approximated with an equation of a circle. Hence, its workspace $\mathcal{W}_1^R$ is given by

$$\mathcal{W}_1^R = \{(x_1, x_2, x_3) \in \mathbb{R}^2 : (x_2 - y_0)^2 + (x_3 - z_0)^2 = r^2\} \tag{6.14}$$

where x_1, x_2, and x_3 represent the Cartesian positions of the object being manipulated by that robot, y_0 and z_0 are the position of robot base with respect to an inertial frame and r is the radius of the robotic arm when completely extended. Similar set of geometrical equations can be used to define the geometry of objects and surfaces. Consider an infinitely extending floor surface S_1 in the xy plane. Its workspace can be written as $z = 0$ with respect to the same frame. For a contact configuration of $O_1R_1 \wedge O_1S_1$, the object O_1 can be manipulated in the intersecting workspace of R_1 and S_1, i.e. $\mathcal{W}_1^R \cap \mathcal{W}_1^S$.

The workspace of a node $N_j \in \mathcal{N}$ is defined as $\mathcal{W}_j^N$, meaning that the manipulation workspace for all the objects involved in that node. The workspace of an edge between two nodes on the contact graph can be computed by the intersection of the workspaces of two nodes, i.e.

$$\mathcal{W}_{k,l}^E \Leftrightarrow \mathcal{W}_k^N \bigcap \mathcal{W}_l^N \tag{6.15}$$

where $\mathcal{W}_{k,l}^E$ is the workspace of the edge and $\mathcal{W}_k^N$ and $\mathcal{W}_l^N$ are the workspaces of the kth and lth nodes, respectively. This is an abstract form of representing the node and edge workspaces. For computational purposes, the manipulable workspace of each object needs to be considered separately. The workspace of a node represents the physical space in which the robots involved can manipulate the objects whereas the workspace of an edge represents the space in which the transition between two contact configurations can take place. Geometrical information can be extracted using various robot vision techniques from 3D point clouds followed by spatial reasoning to determine and extract various geometrical features of different objects [107]. On having this geometrical information, certain nodes and edges can be discarded from the planning graph if their corresponding workspaces are null, i.e. $\mathcal{W}_j^N = \emptyset$ and $\mathcal{W}_{k,l}^E = \emptyset$.

Additionally, geometrical, temporal and other metrics are used to define weights on the edges of the contact graph, as shown in Figure 6.2, using the weights w_i. By adding weights, the manipulation planning problem becomes a standard graph search/traversal problem. In a more involved case, many paths exist to reach from the initial to the goal node. A need arises to find the shortest path with the weights acting as metrics for the search problem, which can be found via search algorithms like Dijkstra's, A^*, etc [29].

6.3.3 Parallelization

It is possible that for a given overall task, there could be intermediate sub-tasks that are independent and can be carried out in parallel. In such a case, the parallelization rule given by (6.12) can be applied and those tasks could be carried out in parallel. Parallelization would consequently reduce the amount of time to perform the overall task. Take for example the case where an intermediate task

is that two individual robots R_1 and R_1 have to pick up two separate objects O_1 and O_2, respectively. If this task is done sequentially, the time taken will be the total time to perform both tasks. In contrast, when it is done simultaneously or in parallel, the overall time taken will be the maximum one of the two tasks. This has been illustrated in Figure 6.3.

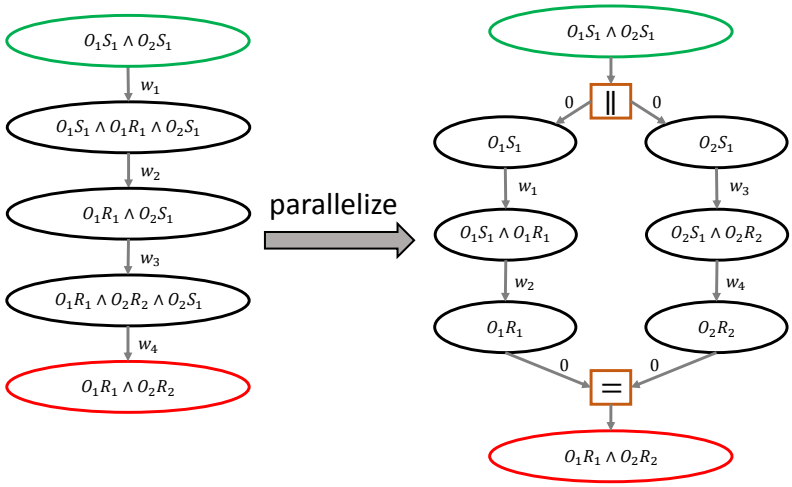

Figure 6.3: Two contact graphs comparing a task performed sequentially and in parallel, using the parallelization rule. When the task is performed sequentially, the total time will be $\sum_{i=1}^{4} w_i$ whereas if it is done in parallel, the total time will be $\max(w_1 + w_2, w_3 + w_4)$.

Considering the weights to be time, if the task is performed sequentially, it is seen that the total task time will be the sum of all weights, i.e. $\sum_{i=1}^{4} w_i$. On the other hand, if the task is carried out in parallel, the total task time will be $\max(w_1 + w_2, w_3 + w_4)$, which is less than the former one assuming finite time for each transition. As such, parallelization can lead to interesting temporal dynamics of the overall hybrid system, thereby, opening up the possibility to carry out interesting optimizations and time scheduling for larger systems using the proposed planner. However, when parallelization is used, the search algorithms such as Dijkstra and A^* cannot directly be used and a modification is needed to decide the best path on the contact graph.

6.3.4 Low-Level Planning and Control

Once the path for a given manipulation is generated on the contact graph, the next step towards accomplishing the task is workspace path planning followed by controller assignment. Each node on this path represents a contact configuration along with a sub-manipulation task that needs to be achieved in that config-

uration. Workspace path planning is done for each node followed by controller assignment that performs those sub-manipulations to achieve the overall object manipulation.

Workspace path planning refers to manipulation planning in workspace. Before the path of the robotic manipulators is planned, the path of the object being manipulated needs to be planned. From the shortest path on the contact graph, we have the relevant nodes and the initial and final position of the object. The objective of the object path-planner is to plan a path separately for each of these nodes for object manipulation and reaching the goal position. To achieve this objective, a piecewise path planner should be implemented which satisfies the condition. The goal position of the path in each node is served as the initial position of the object path planning for the next node. This way, the resulting overall path will be connected and smooth. Here, we suppose that the controllers' DoAs completely cover the workspace, hence we assume that there always exists a low-level controller.

In order to manipulate an object, the robots need specific manipulation controllers. These could be a single robot manipulating an object or multiple robots cooperatively grasping and manipulating an object. As manipulation tasks involve interaction of the robot with the object manipulated, some form of compliance is required in the arm and its grippers to control the interaction with the object and avoid chattering and damage to the robot and the object. The most commonly applied control method is impedance control [50, 95]. This model based control method makes the robotic arm to behave like a virtual mass-spring-damper system. The dynamics of a robot in the Cartesian space is given by

$$\Lambda(x)\ddot{x} + \mu(x,\dot{x})\dot{x} + F_{\mathrm{g}} = F_{\tau} + F_{\mathrm{ext}} \tag{6.16}$$

where x is the robot's end-effector six dimensional position and orientation vector. Moreover, $\Lambda(x)$, $\mu(x,\dot{x})$, and F_{g} are respectively the robot's inertia, Coriolis and gravitational matrix in the Cartesian space. F_{τ} and F_{ext} are the end-effector input and external forces in the Cartesian space, respectively [95]. The desired end-effector dynamics for its impedance control, without inertia shaping, seen as

$$\Lambda(x)\ddot{\tilde{x}} + D_{\mathrm{d}}\dot{\tilde{x}} + K_{\mathrm{d}}\tilde{x} = F_{\tau}. \tag{6.17}$$

Here, K_{d} and D_{d} are the stiffness and damping parameter of the virtual spring, respectively [36, 115]. Moreover, F_{τ} is the end-effector force imposed by the robot and $\tilde{x}$ is end-effector position error vector (i.e., $x - x_d$). The spring parameters can be varied to change the behavior of this virtual spring, and hence the robotic arm, thereby making it either compliant or stiff. The behavior can also be controlled by changing the desired end-effector position. The end-effector force is indirectly controlled by varying the spring parameters and the desired end-effector position. During cooperative manipulation, the grasp geometry also leads to certain kinematic constraints which have to be adhered to a successful cooperative robotic

manipulation (see e.g., [14, 133, 100, 127]). The joint input torque τ is transformed from the end-effector input force via

$$\tau = J(q)^T(F_\tau + F_{\mathrm{g}}) \tag{6.18}$$

where q is the joint position vector and $J(q)$ is the robot's Jacobian matrix describing the robot kinematics [77, 95]. In addition, F_{g} is used for compensating the gravitational forces on the robot.

Controllers are required in manipulation in order to make and break contact with the object manipulated. These are called transition controllers. To demarcate a change in contact involving robots, these controllers are used. These controller are in line with the making and breaking contact rules stated in Section 6.2.2. The idea of transition and manipulation controllers is similar to the general concept of transfer and transit controllers [114, 1]. A final step to achieve the manipulation task is putting all these controllers together and executing them in a synchronized and hybrid manner.

It is seen that the workspaces of the contact modes are already overlapping for a given manipulation task. The path planning algorithm also works in a way that the goal position of each sub-task or contact mode lies in the workspace of the next one. Hence, the workspaces are already sequentially composed. The DoA on the other hand, should include all the relevant dynamic information or states in this composition. On having a control automaton, performing the given task is just a matter of executing the controllers, starting from the initial condition, switching among relevant controllers and accomplishing the overall task.

6.4 Simulation Results

The first example of this chapter is symbolic manipulation planning. For this purpose, various planning scenarios with different initial and goal conditions were tested using the software Mathematica. To demonstrate the translation of the task planner to low-level control, the virtual robotics experimentation platform (VREP) software was used in conjunction with the Matlab for simulating a manipulation task using two robots and an object. Three different test cases are presented in this chapter to illustrate task-planning on a symbolic level. One of the test cases has been extended to low-lower control as well. Each test case consists of different number of robots, objects and surfaces along with a different initial and goal position. To cover a broad spectrum of tasks, each test case consists of a special aspect to the task being performed.

6.4.1 Pushing and Lifting an Object

This example consists of two robots, one object and one surface. The initial position of the object is on the surface and the goal position is in the air, held by the two robots. This example is also used further to illustrate low-level path planning, control as well as cooperative manipulation.

The contact graph is generated by applying the rules and assumptions of the robot contact language, as illustrated in Figure 6.4. This graph shows the set of all possible nodes along with the possible edges between the nodes.

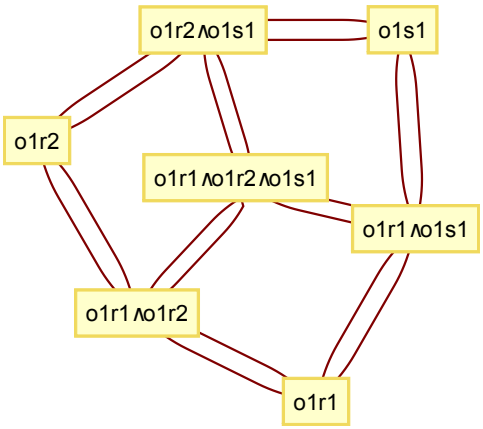

Figure 6.4: Contact graph for the example of pushing and cooperatively lifting an object with two robots, one object, and one surface.

From the given scenario, the initial node is O_1S_1 and the goal node is $O_1R_1 \wedge O_1R_2$. The manipulation planner returns one or more paths from the initial to goal node on this graph. Considering the same weight for all edges on the graph, there are multiple paths with the same weight between the two initial and goal nodes. Two of these paths are represented in Figure 6.5 using red lines. On interpreting the nodes in these paths, the manipulation plan would be to lift the object first by either one of the robotic arms followed by the other arm making contact with the object when it is in the air and is held by the first arm. It is natural to assume that all the generated plans cannot be carried out due to physical and geometrical limitations. On using this information further, certain paths will have higher costs or would be completely eliminated, leading to some infeasible plans or paths.

The simulation setup in VREP consists of two robots with partly overlapping workspaces. The initial position of the object lies in the workspace of R_1 and the goal position obviously lies in the common workspace of R_1 and R_2, as shown in Figure 6.6 which illustrates four selected snapshots for the task of pushing and cooperatively lifting an object. This geometrical information is used to modify the contact graph and leads to elimination of certain edges on the graph. A physical

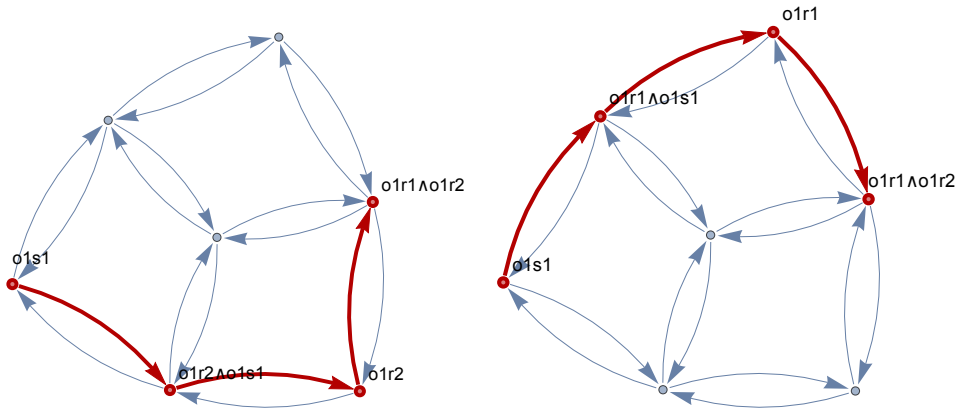

Figure 6.5: Contact graphs for the example of pushing and cooperatively lifting an object without considering the geometries and physical limitations of the components. The graphs represent two different paths from the initial to goal node as shown in red.

limitation with these robots is that they do not have a gripper or a grasper. This restricts them to individually manipulate the object in the air. Hence, the nodes O_1R_1 and O_1R_2 are invalid and can be removed from the contact graph. Due to these modifications, the path for the given task can not be as given in Figure 6.5.

The new path is given in Figure 6.7 that involves cooperative manipulation of the object using the two arms, as seen from the node $O_1R_1 \wedge O_1R_2$. Moreover, this path is consistent with scene geometry and physical limitations of robots. Hence, one can continue with object path planning and low-level control assignment for each of the sub-paths. Each node in the shortest path points at a potential sub-manipulation task whereas the edges can refer to a robot either making or breaking contact with an object.

The workspace path planning for the object manipulation is based on a standard grid-based Dijkstra's shortest path algorithm which is followed by robot end-effector path planning. The manipulation and transition controller are based on spatial-springs based impedance controllers [14, 115, 36] with varying controller parameter as desired by task performed. For cooperative manipulation, the trajectories of the robot's end-effector are generated from the workspace path planned for object manipulation [14]. The idea of sequential composition is used and the DoAs and goal sets for each robot controllers is found from overlapping workspaces of the nodes, the contact configuration of nodes, and the information from workspace path planning of the object. Using this information, switching conditions for controllers are defined and the control system is represented in the form of a control automaton. On execution, the overall manipulation task is achieved.

Figure 6.8 illustrates the simulation results for the left robotic arm of Figure 6.6 in

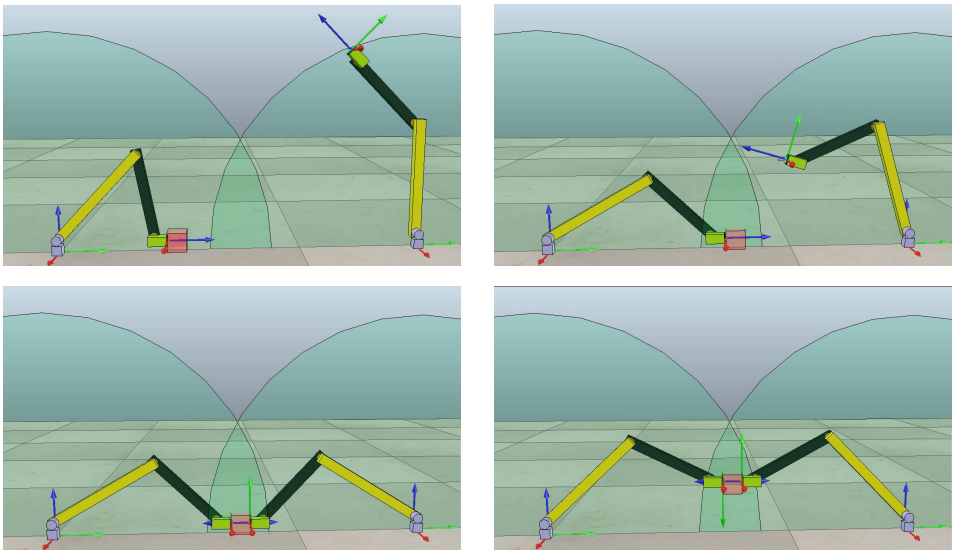

Figure 6.6: Snapshots of the simulated manipulation task in VREP (Left to right, top to bottom) for the task of pushing and cooperatively lifting an object. The shaded blue regions are the workspaces of the two planar robots which are partly overlapping. The red object is being first pushed into the overlapping workspace by one of the robots followed by both robots cooperatively lifting the object.

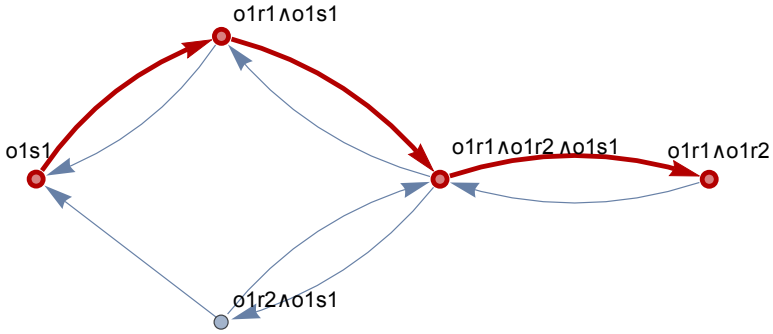

Figure 6.7: Contact graph for the example of pushing and cooperatively lifting an object after considering the geometries and physical limitations of the components. The shortest path from the initial to goal node is shown in red.

the example of pushing and cooperatively lifting an object. Similarly, Figure 6.9 depicts the simulation results for the right robotic arm. In the both figures, the first two plots illustrate the end-effector positions along the y and z axes. The desired end-effector trajectories generated via path planning are shown in red and the actual trajectories are given in blue. The last plots in the figures represent the end-effector force along the z axis for the left and right manipulators. This simulation has been done in VREP.

Five different controllers are used in the overall task: two for robot transition, one for pushing the robot, and two for cooperative manipulation (functioning simultaneously). The vertical dotted line denotes the time instance of controller switching: the first region indicates the make-contact controller of the left robot, the second region represents a push-controller for the same robot to bring the object into the overlapping workspace, the third region shows the make-contact controller of the right robot, and the last region represents a cooperative manipulation controller for each object that manipulates objects while adhering to closed chain constraints. Thus, the control of the two individual robotic arms leads to successful object manipulation while tracking their respective workspace paths.

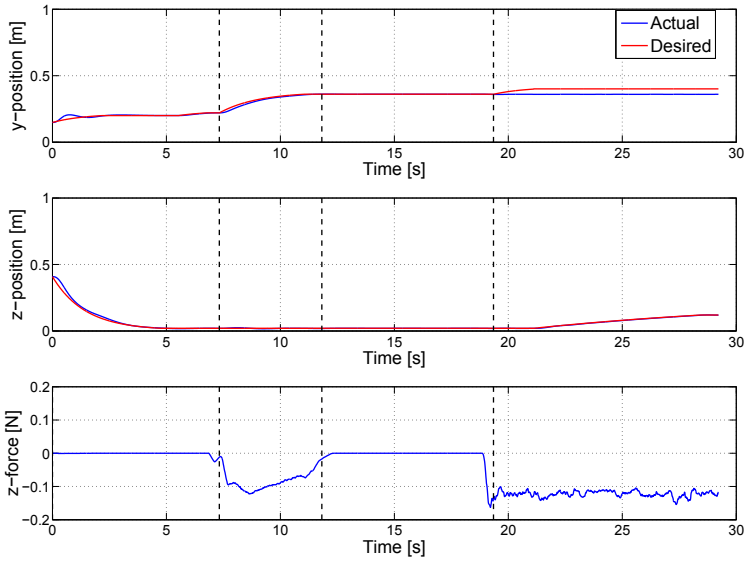

Figure 6.8: Simulation results for the left robotic arm of Figure 6.6 in the example of pushing and cooperatively lifting an object. The first two plots illustrate the end-effector positions along the y and z axes. The desired end-effector trajectories are shown in red and the actual trajectories are in blue. The last plot represents the end-effector force along the z axis.

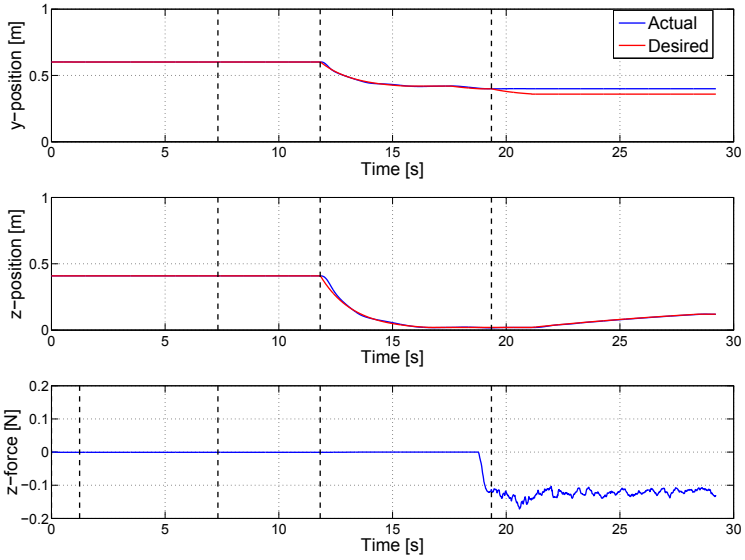

Figure 6.9: Simulation results for the right robotic arm of Figure 6.6 in the example of pushing and cooperatively lifting an object. The first two plots illustrate the end-effector positions along the y and z axes. The desired end-effector trajectories are shown in red and the actual trajectories are in blue. The last plot represents the end-effector force along the z axis.

6.4.2 Stacking Objects

This example consists of one robot, two objects, and two surfaces. The initial position of the two objects is on two separate surfaces while the final position is in a stacking position on one of the surfaces. This is an important example in AI planning. The sequence of manipulation matters as it involves stacking of objects. Along with sequential planning, the decision of manipulating the correct object first is also made using this planner.

The complete contact graph is shown in Figure 6.10, which is generated by applying the rules and assumptions of the robot contact language. The graph shows a set of all possible nodes along with the feasible edges between two nodes. The initial node for the task is $O_1S_1 \wedge O_2S_2$ and the goal node is $O_1O_2 \wedge O_2S_1$. The manipulation planner returns two paths from the initial to the goal node for this task. Considering the same weight for all edges on the graph, the shortest and the alternate paths on the graph is given in Figure 6.11, shown in red.

The manipulation plan is hence for R_1 to first pick and place O_2 from S_2 to S_1 followed by picking and placing O_1 from S_1 to O_2. As expected, the manipulation planner achieves task planning and gives out the correct order of object manipulation for their stacking process. The alternative path also produces a plan for the same manipulation task but with a longer path on the contact graph.

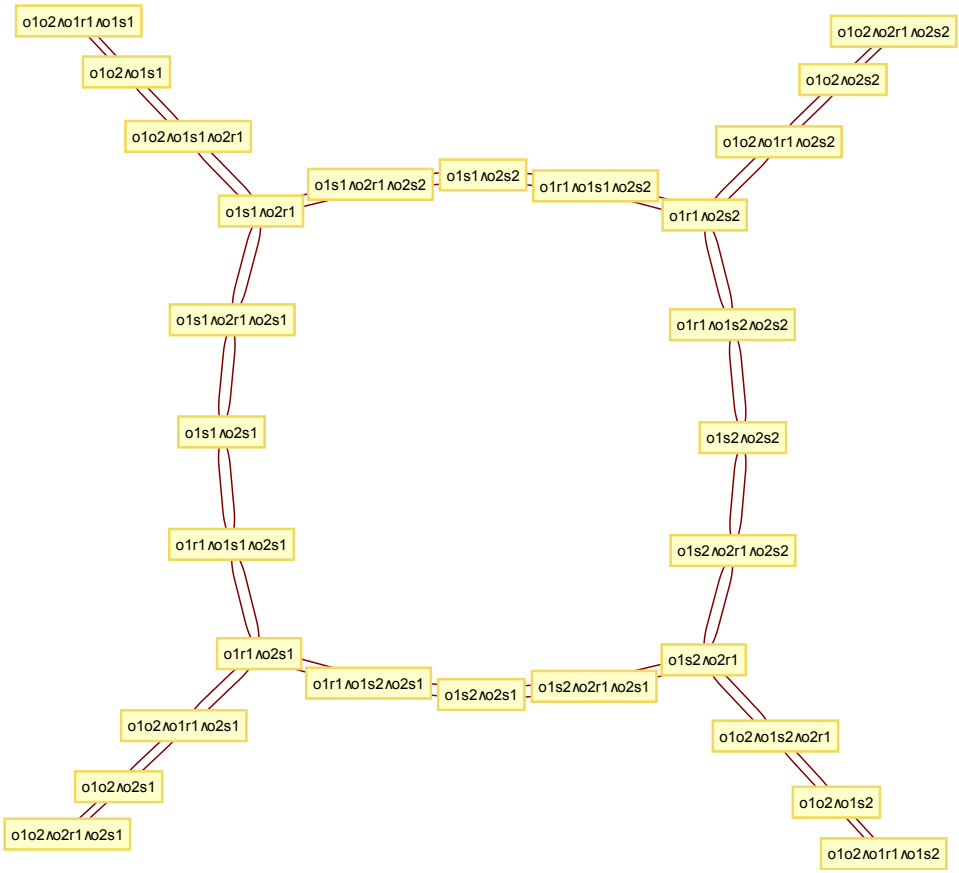

Figure 6.10: Contact graph for the example of stacking objects with one robot, two objects, and two surfaces.

6.4.3 Parallel Manipulation

This example consists of two robots, two objects, and one surface. The task for each of the two robots is to lift a separate object and manipulate them individually. This example is used to illustrate manipulations of two objects carried out in parallel using the parallelization rule.

The initial node for the given task is $O_1S_1 \wedge O_2S_1$ and the goal node is $O_1R_1 \wedge O_2R_2$. The shortest path for this task is depicted in Figure 6.12, represented by red arrows. On observing every node in this path, one can see that there is potential for parallelizing the task. The two robots R_1 and R_2 can lift and manipulate their respective objects O_1 and O_2 simultaneously. On applying the paralleling rule, the

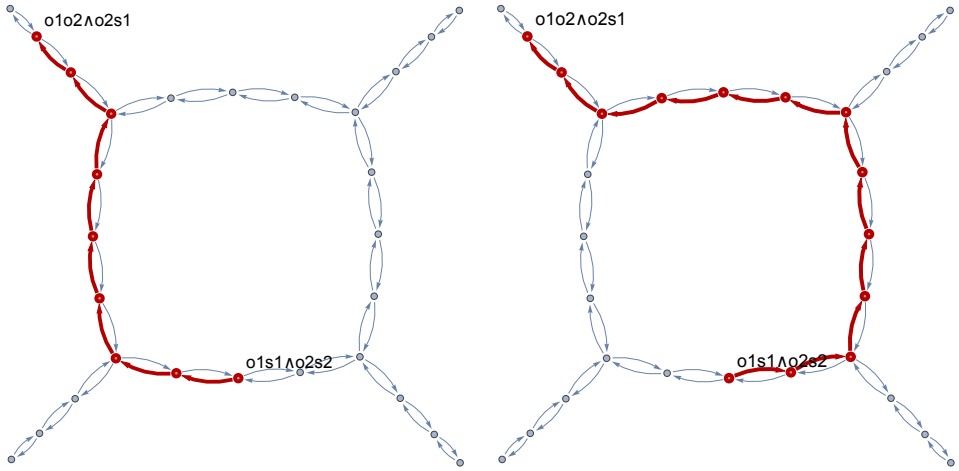

Figure 6.11: Contact graphs for the example of stacking objects with one robot, two objects, and two surfaces. The graph on the left shows the shortest path on the graph, shown in red, whereas the graph on the right shows an alternate path for the same task.

task can be divided into two parallel paths given by

$$\begin{gathered} O_1S_1 \parallel O_2S_1 \\ O_1S_1 \wedge O_1R_1 \parallel O_2S_1 \wedge O_2R_2 \\ O_1R_1 \parallel O_2R_2 \end{gathered} \tag{6.19}$$

where time optimizations can be carried out while performing the overall task.

6.5 Conclusions

This chapter proposed a robot contact language for robotic manipulation planning. This symbolic planner is based on the idea of change in contact between robots, object, and surfaces while using standard graph theory techniques. The derived symbolic rules of robot making and breaking contact with objects along with lifting and placing them from or on surfaces are used to generate a supervisory graph. This contact graph is enhanced using the available geometrical and other relevant information. The sub-tasks are attained by assigning different controllers and executing them in a hybrid manner. Simulation results show how tasks like pushing, lifting, and stacking objects can easily be planned using this robot contact language. Moreover, independent sub-tasks can also be planned to be carried out in parallel using the derived parallelization rules.

This contact-based approach is highly intuitive and the task-division conforms

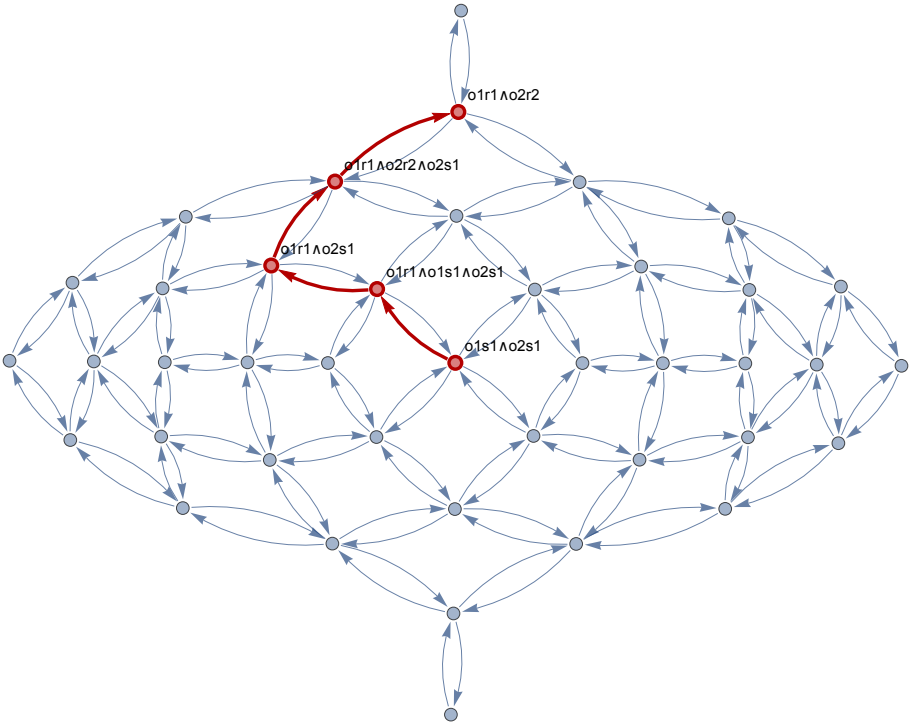

Figure 6.12: Contact graph for the example of parallel manipulation with two robots, two objects, and one surface. The shortest path is represented by red arrows. This task can be planned sequentially as well as in parallel using the parallelization rules.

with the need for a change in controller specification. However, it is not designed to be a generalized AI planner and focuses on the planning and optimization of object manipulations. Although the studied examples only focus on the robotic arms, the proposed language is also applicable to mobile manipulators and other robotic systems. This manipulation planner assumes full-world information, including robot workspaces and contact surfaces. It is also static in nature and can not deal with temporal or dynamic planning, like throwing and catching objects. On having these mentioned drawbacks, the planner can serve as a helpful tool in manipulation planning and scheduling. The problem is formulated as a graph search with a straightforward application of temporal and spatial constraints as edge weights.

7 CHAPTER

Conclusions and Future Research

This chapter provides a summary of the thesis as well as a conclusion from the contributions discussed throughout. It closes with some research directions for future work.

7.1 Conclusions

This thesis discussed automatic synthesis of supervisory controllers by means of learning and taking advantage of coordination between multiple control systems in the context of sequential composition. Estimating the DoA of controllers provides a notion of safe learning in a sense that the experiments are just allowed to search within the existing DoAs. Moreover, online monitoring the DoAs enables learning in a short amount of time by introducing a stopping criterion for the learning procedure. Additionally, a robot contact language has been proposed for the manipulation of multiple objects by multiple robots. This language is described with regards to the contact that is made or broken between a collection of involved components. In the following, a conclusion on the main contributions of this thesis is provided.

Estimating the domain of attraction. A fast sampling approach is proposed for estimating the DoAs of nonlinear systems in real-time. The sampling approach is fast and computationally effective in comparison with existing optimization-based methods. This technique is useful for real-time applications. According to the empirical evidence, it is guaranteed that this approach converges to the exact level set for a sufficiently large number of samples. It is shown that the convergence rate directly depends on the distribution function selected for sampling. By means of a more sophisticated distributed function the convergence rate of the sampling procedure can be increased. There is always a trade-off between

the speed of convergence and the computational cost imposed by the complexity of distribution function. The sampling approach is beneficial for the design of passivity-based learning controllers, where the system Hamiltonian can be used as a candidate Lyapunov function for approximating the controller's DoA.

Learning sequential composition control. One of the approaches that enables automatic synthesis of supervisory controllers is learning sequential composition control developed to cope with unmodeled situations using online learning. The standard sequential composition framework is augmented with a learning mode to add new controllers to the supervisory structure on a need basis. The learning process is safe since it is bounded to the union of existing DoAs. Estimating the DoA of the learned controller at every trial results in rapid learning by terminating the learning process once the DoA is sufficiently large to respect the control objective. In addition, the proposed control approach has been extended to PLTs algorithm as a feedback motion planning. It learns a collection of controllers safely for a class of systems. Although PLTs inherits the challenges present in RL, it is useful for motion planning as it learns in small regions of the state space instead of entire state space at once.

Cooperative sequential composition control. Another technique that enables automatic synthesis of supervisory controllers is cooperative sequential composition control. It composes two or multiple independent supervisory controllers to achieve new tasks that were not possible by each controller individually. If mathematical models for all the dynamics and interaction constraints are present, it is feasible to apply the composition rules to obtain a new cooperative control automaton with extra composed events. Each system receives only the relevant composed events for its own control automaton. The whole procedure is automatic without need to change anything in the original control automata. The new composed events are generated based on the prepare relation between the DoAs of composed controllers. Using this approach, a rich task can be accomplished by executing a composed event on the cooperative control simultaneously.

Robot contact language. A new approach to robotic manipulation planning is proposed, called robot contact language. This symbolic planner is based on the change in contact between robots, objects, and surfaces while using standard graph theory techniques. The derived symbolic rules of robot making and breaking contact with objects along with lifting and placing them from or on surfaces are used to generate a symbolic graph. This supervisory graph is enhanced using the available geometrical and other relevant information presents in manipulation. Standard graph search algorithms can then be applied to plan the manipulation task on an abstract level, which indirectly divides a complex manipulation task into sub-tasks based on contact. These sub-tasks can be obtained by assigning different controllers and executing in a hybrid manner. Simulation results show how tasks like pushing, lifting and stacking objects can easily be planned using this language. Moreover, independent sub-tasks can also be planned to be carried out

in parallel using the derived parallelization rules. This contact-based approach is not designed to be a generalized AI planner. It is highly intuitive and focuses on the planning and optimization of object manipulations. The proposed language is applicable to mobile manipulators and other robotic systems.

7.2 Recommendations for Future Work

This section recommends some research direction for future work. The suggestions are as follows.

- **Formal guarantee for the sampling method.** Although the empirical evidence suggests that the result of the proposed sampling method converges to the exact level set for a large number of samples, a formal guarantee for the convergence of sampling does not exist yet.
- **Distributed function for the sampling method.** The convergence rate of the sampling method depends on the distribution function selected for sampling. Using a more sophisticated function can considerably speed up the convergence rate. However, there is always a trade-off between the speed of convergence and the computational cost imposed by the complexity of the distribution function. This needs to carefully be studied so that an appropriate distribution function is chosen for the sampling procedure.
- **Non-Lyapunov methods for estimating the DoA.** In this thesis, we only implemented the sampling technique for the Lyapunov-based methods. However, the proposed sampling approach can be extended to non-Lyapunov methods as well. This enables estimating the DoAs of both model-based and non-model based controllers.
- **Generic learning experiments.** The design of a generic learning experiment in the context of learning sequential composition control to learn proper control laws in a short amount of time is a real challenge. This is due the fact that for each system the reward function, the learning rates and the parametrization of the value function must carefully be chosen. This thesis did not address the automatic choice of these parameters, but this could be a research line for future work.
- **Search in the control automaton.** In the current learning sequential composition control approach, the next controller of the sequence is selected randomly when there are multiple possibilities. However, a distance function can be defined to compute the smallest distance between all the goal sets reachable by the initial state via sequential composition and all the DoAs that can lead to the desired state. Once a pair of goal set and DoA is found,

the reward function of the learning module is set to give positive rewards in the directions that minimize the distance between the associated goal set and the desired DoA, using the distance function. This function can be complex to compute if the DoAs and goal sets are non-convex.

- **Interaction dynamics in cooperative manipulation.** In the cooperation of two or multiple systems, a good knowledge on the interaction dynamics is very beneficial. In this thesis, we manually calculated a simplified model for the interaction between two collaborating systems. However, the automatic computation of this interaction can effectively simplify the control synthesis. Future research includes using partial dynamical models where learning needs to happen in real-time towards achieving the interaction dynamics.

- **Extension of the robot contact language.** The proposed contact-based language focuses on the manipulation of objects with simple geometry while designing low-level controllers for oddly-shaped objects in the context of this planner is quite challenging. This even gets more severe if one also considers the objects dynamics. As such, the robot contact language should be studied further, mainly with respect to the low-level controllers. This could be very useful for the assembly lines in various industries, where contact between the robots and objects with irregular geometries is very trivial.

Bibliography

[1] Rachid Alami, Jean-Paul Laumond, and Thierry Siméon. Two manipulation planning algorithms. In *Proceedings of the workshop on Algorithmic foundations of robotics*, pages 109–125, 1995.

[2] Rajeev Alur, Thao Dang, and Franjo Ivančić. Progress on reachability analysis of hybrid systems using predicate abstraction. In *Hybrid Systems: Computation and Control*, pages 4–19. Springer, 2003.

[3] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[4] Francesco Amato, Carlo Cosentino, and Alessio Merola. On the region of attraction of nonlinear quadratic systems. *Automatica*, 43(12):2119–2123, 2007.

[5] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[6] Nora Ayanian and Vijay Kumar. Abstractions and controllers for groups of robots in environments with obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3537–3542, 2010.

[7] Robert Baier and Matthias Gerdts. A computational method for non-convex reachable sets using optimal control. In *Proceedings of the European Control Conference*, pages 23–26, 2009.

[8] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[9] Randal W Beard, Jonathan Lawton, and Fred Y Hadaegh. A coordination architecture for spacecraft formation control. *IEEE Transactions on Control Systems Technology*, 9(6):777–790, 2001.

[10] Antonio Bicchi, Alessia Marigo, and Benedetto Piccoli. Feedback encoding for efficient symbolic control of dynamical systems. *IEEE Transactions on Automatic Control*, 51(6):987–1002, 2006.

[11] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] Avrim L Blum and Merrick L Furst. Fast planning through planning graph analysis. *Artificial intelligence*, 90(1):281–300, 1997.

[13] Robert R Burridge, Alfred A Rizzi, and Daniel E Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999.

[14] Fabrizio Caccavale, Pasquale Chiacchio, Alessandro Marino, and Luigi Villani. Six-DOF impedance control of dual-arm cooperative manipulators. *IEEE/ASME Transactions on Mechatronics*, 13(5):576–586, 2008.

[15] Fabrizio Caccavale, Ciro Natale, Bruno Siciliano, and Luigi Villani. Six-DOF impedance control based on angle/axis representations. *IEEE Transactions on Robotics and Automation*, 15(2):289–300, 1999.

[16] Stephane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126, 2009.

[17] Christos G Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Springer Science & Business Media, 2008.

[18] G Chesi, A Garulli, A Tesi, and A Vicino. LMI-based computation of optimal quadratic Lyapunov functions for odd polynomial systems. *International Journal of Robust and Nonlinear Control*, 15(1):35–49, 2005.

[19] Graziano Chesi. Estimating the domain of attraction for uncertain polynomial systems. *Automatica*, 40(11):1981–1986, 2004.

[20] Graziano Chesi. Domain of attraction: estimates for non-polynomial systems via LMIs. In *Proceedings of the 16th IFAC World Congress*, 2005.

[21] Graziano Chesi. Estimating the domain of attraction via union of continuous families of Lyapunov estimates. *Systems & control letters*, 56(4):326–333, 2007.

[22] Graziano Chesi. Estimating the domain of attraction for non-polynomial systems via LMI optimizations. *Automatica*, 45(6):1536–1541, 2009.

[23] Graziano Chesi. *Domain of Attraction: Analysis and Control via SOS Programming*. Springer, 2011.

[24] Graziano Chesi. Rational Lyapunov functions for estimating and controlling the robust domain of attraction. *Automatica*, 49(4):1051–1057, 2013.

[25] David C Conner. *Integrating Planning and Control for Constrained Dynamical Systems*. ProQuest, 2007.

[26] David C Conner, Howie Choset, and Alfred A Rizzi. Flow-through policies for hybrid controller synthesis applied to fully actuated systems. *IEEE Transactions on Robotics*, 25(1):136–146, 2009.

[27] David C Conner, Hadas Kress-Gazit, Howie Choset, Alfred Rizzi, and George J Pappas. Valet parking without a valet. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 572–577, 2007.

[28] David C Conner, Alfred Rizzi, and Howie Choset. Composition of local potential functions for global robot control and navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3546–3551, 2003.

[29] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[30] John J Craig and M Raibert. A systematic method of hybrid position/force control of a manipulator. In *Proceedings of the IEEE Computer Society's 3rd International Computer Software and Applications Conference*, pages 446–451, 1979.

[31] Daniel Cruz, James McClintock, Brent Perteet, Omar AA Orqueda, Yuan Cao, and Rafael Fierro. Decentralized cooperative control-a multivehicle platform for research in networked embedded systems. *IEEE Control Systems Magazine*, 27(3):58–78, 2007.

[32] Richard Dearden and Chris Burbridge. Manipulation planning using learned symbolic state abstractions. *Robotics and Autonomous Systems*, 62(3):355–365, 2014.

[33] Jaydev P Desai, James P Ostrowski, and Vijay Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[34] Christian Dornhege, Marc Gissler, Matthias Teschner, and Bernhard Nebel. Integrating symbolic and geometric planning for mobile manipulation. In *Proceedings of the IEEE International Workshop on Safety, Security & Rescue Robotics*, pages 1–6, 2009.

[35] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[36] Ernest D Fasse and Jan F Broenink. A spatial impedance controller for robotic manipulation. *IEEE Transactions on Robotics and Automation*, 13(4):546–556, 1997.

[37] J Alexander Fax and Richard M Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[38] Richard E Fikes and Nils J Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[39] Kenji Fujimoto and Toshiharu Sugie. Canonical transformation and stabilization of generalized Hamiltonian systems. *Systems & Control Letters*, 42(3):217–227, 2001.

[40] Roberto Genesio, Michele Tartaglia, and Antonio Vicino. On the estimation of asymptotic stability regions: State of the art and new proposals. *IEEE Transactions on Automatic Control*, 30(8):747–755, 1985.

[41] Malik Ghallab, Craig Knoblock, David Wilkins, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, and David E Smith. PDDL-the planning domain definition language. 1998.

[42] Antoine Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.

[43] Fabien Gravot, Stephane Cambon, and Rachid Alami. aSyMov: a planner that deals with intricate symbolic and geometric problems. In *Proceedings of the 11th International Symposium Robotics Research*, pages 100–110, 2005.

[44] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(6):1291–1307, 2012.

[45] Ivo Grondman, Maarten Vaandrager, Lucian Busoniu, Robert Babuska, and Erik Schuitema. Efficient model learning methods for actor–critic control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(3):591–602, 2012.

[46] O Hachicho. A novel LMI-based optimization algorithm for the guaranteed estimation of the domain of attraction using rational Lyapunov functions. *Journal of the Franklin Institute*, 344(5):535–552, 2007.

[47] O Hachicho and B Tibken. Estimating domains of attraction of a class of nonlinear dynamical systems with LMI methods based on the theory of moments. In *Proceedings of the 41st IEEE International Conference on Decision and Control*, pages 3150–3155, 2002.

[48] Roland Hafner and Martin Riedmiller. Reinforcement learning in feedback control. *Machine learning*, 84(1-2):137–169, 2011.

[49] Didier Henrion and Milan Korda. Convex computation of the region of attraction of polynomial control systems. *IEEE Transactions on Automatic Control*, 59(2):297–312, 2014.

[50] Neville Hogan. Impedance control: An approach to manipulation. In *Proceedings of the American Control Conference*, pages 304–313, 1984.

[51] Neville Hogan. Impedance control: An approach to manipulation: Part ii–implementation. *Journal of dynamic systems, measurement, and control*, 107(1):8–16, 1985.

[52] Mikael Johansson and Anders Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, 1998.

[53] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011.

[54] Vinutha Kallem, Adam T Komoroski, and Vipin Kumar. Sequential composition for navigating a nonholonomic cart in the presence of obstacles. *IEEE Transactions on Robotics*, 27(6):1152–1159, 2011.

[55] George Kantor and Alfred A Rizzi. Feedback control of underactuated systems via sequential composition: Visually guided control of a unicycle. In *Proceedings of the 11th International Symposium Robotics Research*, pages 281–290, 2005.

[56] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[57] Hassan K Khalil and JW Grizzle. *Nonlinear Systems*. Prentice hall Upper Saddle River, 2002.

[58] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[59] George Konidaris and Andre S Barreto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pages 1015–1023, 2009.

[60] James J Kuffner and Steven M LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 995–1001, 2000.

[61] Gerardo Lafferriere, Alan Williams, J Caughman, and JJP Veerman. Decentralized control of vehicle formations. *Systems & control letters*, 54(9):899–910, 2005.

[62] Steven M LaValle. *Planning Algorithms*. Cambridge university press, 2006.

[63] Jonathan RT Lawton, Randal W Beard, and Brett J Young. A decentralized approach to formation maneuvers. *IEEE Transactions on Robotics and Automation*, 19(6):933–941, 2003.

[64] JL Le Ny and George J Pappas. Sequential composition of robust controller specifications. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5190–5195, 2012.

[65] M Anthony Lewis and Kar-Han Tan. High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4(4):387–403, 1997.

[66] Daniel Liberzon. *Switching in Systems and Control*. Springer Science & Business Media, 2012.

[67] Stephen R Lindemann, Islam Hussein, and Steven M LaValle. Real time feedback control for nonholonomic mobile robots with obstacles. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 2406–2411, 2006.

[68] Stephen R Lindemann and Steven M LaValle. Smoothly blending vector fields for global robot navigation. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 3553–3559, 2005.

[69] Stephen R Lindemann and Steven M LaValle. Smooth feedback for car-like vehicles in polygonal environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3104–3109, 2007.

[70] Stephen R Lindemann and Steven M LaValle. Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions. *The International Journal of Robotics Research*, 28(5):600–621, 2009.

[71] Gabriel AD Lopes, Esmaeil Najafi, Subramanya P Nageshrao, and Robert 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., pages 53–74. Springer, 2015.

[72] Anirudha Majumdar, Ram Vasudevan, Mark M Tobenkin, and Russ Tedrake. Convex optimization of nonlinear feedback controllers via occupation measures. *The International Journal of Robotics Research*, 33:1209–1230, 2014.

[73] Matthew T Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 11(6):418–432, 1981.

[74] Luis G Matallana, Aníbal M Blanco, and J Alberto Bandoni. Estimation of domains of attraction: A global optimization approach. *Mathematical and Computer Modelling*, 52(3):574–585, 2010.

[75] Robin Milner. *Communication and Concurrency*. Prentice-Hall, Inc., 1989.

[76] Robert Murphey and Panos M Pardalos. *Cooperative control and optimization*, volume 66. Springer Science & Business Media, 2002.

[77] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[78] Roderick Murray-Smith and T Johansen. *Multiple model approaches to nonlinear modelling and control*. CRC press, 1997.

[79] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Hybrid control for navigation of shape-accelerated underactuated balancing systems. In *Proceedings of the 49th IEEE International Conference on Decision and Control*, pages 3566–3571, 2010.

[80] Umashankar Nagarajan, George Kantor, and Ralph Hollis. Integrated planning and control for graceful navigation of shape-accelerated underactuated balancing mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 136–141, 2012.

[81] Subramanya P Nageshrao, Gabriel AD Lopes, Dimitri Jeltsema, and R Babuska. Passivity-based reinforcement learning control of a 2-DOF manipulator arm. *Mechatronics*, 24(8):1001–1007, 2014.

[82] Subramanya Prasad Nageshrao, Gabriel AD Lopes, Dimitri Jeltsema, and R Babuska. Interconnection and damping assignment control via reinforcement learning. In *Proceedings of the 19th IFAC World Congress*, pages 1760–1765, 2014.

[83] Esmaeil Najafi, Robert Babuska, and Gabriel AD Lopes. An application of sequential composition control to cooperative systems. In *Proceedings of the 10th International Workshop on Robot Motion and Control*, pages 15–20, 2015.

[84] Esmaeil Najafi, Robert Babuska, and Gabriel AD Lopes. A fast sampling method for estimating the domain of attraction. Submitted to *Nonlinear Dynamics*, 2015.

[85] Esmaeil Najafi, Robert Babuska, and Gabriel AD Lopes. Learning sequential composition control. To appear in *IEEE Transactions on Cybernetics*, 2015.

[86] Esmaeil Najafi, Robert Babuska, and Gabriel AD Lopes. Cooperative sequential composition control. 2016.

[87] Esmaeil Najafi, Robert Babuska, and Gabriel AD Lopes. Probabilistic learning trees: a feedback motion planning. 2016.

[88] Esmaeil Najafi and Gabriel AD Lopes. Towards cooperative sequential composition control. Submitted to *the 55th IEEE International Conference on Decision and Control*. 2016.

[89] Esmaeil Najafi, Gabriel AD Lopes, and Robert Babuska. Reinforcement learning for sequential composition control. In *Proceedings of the 52nd IEEE International Conference on Decision and Control*, pages 7265–7270, 2013.

[90] Esmaeil Najafi, Gabriel AD Lopes, and Robert Babuska. Balancing a legged robot using state-dependent Riccati equation control. In *Proceedings of the 19th IFAC World Congress*, pages 2177–2182, 2014.

[91] Esmaeil Najafi, Gabriel AD Lopes, Subramanya P Nageshrao, and Robert Babuska. Rapid learning in sequential composition control. In *Proceedings of the 53rd IEEE International Conference on Decision and Control*, pages 5171–5176, 2014.

[92] Petter Ögren, Edward Fiorelli, and Naomi Ehrich Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, 2004.

[93] Romeo Ortega and Eloisa Garcia-Canseco. Interconnection and damping assignment passivity-based control: A survey. *European Journal of Control*, 10(5):432–450, 2004.

[94] Romeo Ortega, Arjan J Van der Schaft, Iven Mareels, and Bernhard Maschke. Putting energy back in control. *IEEE Control Systems Magazine*, 21(2):18–33, 2001.

[95] Christian Ott. *Cartesian Impedance Control: The Rigid Body Case*. Springer, 2008.

[96] Christian Ott, Oliver Eiberger, Werner Friedl, B Bauml, Ulrich Hillenbrand, Christoph Borst, A Albu-Schaffer, Bernhard Brunner, H Hirschmuller, and S Kielhofer. A humanoid two-arm system for dexterous manipulation. In *Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots*, pages 276–283. IEEE, 2006.

[97] Andrew Packard, Ufuk Topcu, Peter Seiler, and Gary Balas. Help on SOS. *IEEE Control Systems Magazine*, 30(4):18–23, 2010.

[98] Lynne E Parker. Current state of the art in distributed autonomous mobile robotics. In *Distributed Autonomous Robotic Systems 4*, pages 3–12. Springer, 2000.

[99] Sarangi Patel, Sang-Hack Jung, James P Ostrowski, Rahul Rao, and Camillo J Taylor. Sensor based door navigation for a nonholonomic vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3081–3086, 2002.

[100] Veronique Perdereau and Michel Drouin. Hybrid external control for two robot coordinated motion. *Robotica*, 14(02):141–153, 1996.

[101] Giordano Pola, Antoine Girard, and Paulo Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[102] Arthur E Quaid and Alfred Rizzi. Robust and efficient motion planning for a planar robot using hybrid control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4021–4026, 2000.

[103] Philipp Reist and Russ Tedrake. Simulation-based LQR-trees with input and state constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5504–5510, 2010.

[104] Alfred Rizzi. Hybrid control as a method for robot motion programming. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 832–837, 1998.

[105] Pritam Roy, Paulo Tabuada, and Rupak Majumdar. Pessoa 2.0: A controller synthesis tool for cyber-physical systems. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 315–316. ACM, 2011.

[106] Grzegorz Rozenberg and Arto Salomaa. *Handbook of Formal Languages: Beyonds words*. Springer Science & Business Media, 1997.

[107] Radu Bogdan Rusu, Nico Blodow, Zoltan Marton, Alina Soos, and Michael Beetz. Towards 3D object maps for autonomous household robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3191–3198, 2007.

[108] Ahmed Saleme, Bernd Tibken, S Warthenpfuhl, and Christian Selbach. Estimation of the domain of attraction for non-polynomial systems: A novel method. In *Proceedings of the 18th IFAC World Congress*, pages 10976–10981, 2011.

[109] Anuj Shah, Esmaeil Najafi, and Gabriel AD Lopes. Contact-based language for robotic planning and manipulation. Submitted to *the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2016.

[110] Anuj Shah, Esmaeil Najafi, and Gabriel AD Lopes. A robot contact language for manipulation planning. Submitted to *IEEE/ASME Transactions on Mechatronics*, 2016.

[111] JS Shamma and M Athans. Gain scheduling: Potential hazards and possible remedies. *IEEE Control Systems Magazine*, 12(3):101–107, 1992.

[112] O. Sprangers, R. Babuska, S.P. Nageshrao, and G.A.D. Lopes. Reinforcement learning for port-Hamiltonian systems. *IEEE Transactions on Cybernetics*, 45(5):1003–1013, 2015.

[113] Mike Stilman and James Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, 2008.

[114] Mike Stilman, Jan-Ullrich Schamburek, James Kuffner, and Tamim Asfour. Manipulation planning among movable obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3327–3332, 2007.

[115] Stefano Stramigioli. *From differentiable manifold to interactive robot control*. Delft University of Technology, 1998.

[116] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[117] Paulo Tabuada. An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, 53(6):1406–1418, 2008.

[118] Weehong Tan and Andrew Packard. Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming. *IEEE Transactions on Automatic Control*, 53(2):565–570, 2008.

[119] Herbert G Tanner and Dimitrios K Christodoulakis. Decentralized cooperative control of heterogeneous vehicle groups. *Robotics and autonomous systems*, 55(11):811–823, 2007.

[120] Herbert G Tanner, George J Pappas, and Vijay Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–455, 2004.

[121] Russ Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. 2009.

[122] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[123] Alberto Tesi, Francesca Villoresi, and Roberto Genesio. On the stability domain estimation via a quadratic Lyapunov function: Convexity and optimality properties for polynomial systems. *IEEE Transactions on Automatic Control*, 41(11):1650–1657, 1996.

[124] B Tibken and O Hachicho. Estimation of the domain of attraction for polynomial systems using multidimensional grids. In *Proceedings of the 39th IEEE International Conference on Decision and Control*, pages 3870–3874, 2000.

[125] Ufuk Topcu, Andrew Packard, and Peter Seiler. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44(10):2669–2675, 2008.

[126] Ufuk Topcu, Andrew K Packard, Peter Seiler, and Gary J Balas. Robust region-of-attraction estimation. *IEEE Transactions on Automatic Control*, 55(1):137–142, 2010.

[127] Masaru Uchiyama and Pierre Dauchez. A symmetric hybrid position/force control scheme for the coordination of two robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 350–356, 1988.

[128] Arjan Van der Schaft and Dimitri Jeltsema. *Port-Hamiltonian Systems Theory: An Introductory Overview*. Now Publishers Incorporated, 2014.

[129] A Vannelli and M Vidyasagar. Maximal Lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985.

[130] Mike Vande Weghe, Dave Ferguson, and Siddhartha S Srinivasa. Randomized path planning for redundant manipulators without inverse kinematics. In *Proceedings of the 7th IEEE-RAS International Conference on Humanoid Robots*, pages 477–482, 2007.

[131] Joel D Weingarten, Gabriel AD Lopes, Martin Buehler, Richard E Groff, and Daniel E Koditschek. Automated gait adaptation for legged robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2153–2158, 2004.

[132] Douglas Brent West. *Introduction to Graph Theory*. Prentice hall Upper Saddle River, 2001.

[133] David Williams and Oussama Khatib. The virtual linkage: A model for internal forces in multi-grasp manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1025–1030, 1993.

[134] Libo Yang and Steven M Lavalle. The sampling-based neighborhood graph: An approach to computing and executing feedback motion strategies. *IEEE Transactions on Robotics and Automation*, 20(3):419–432, 2004.

List of symbols

Chapter 2

x	state vector
u	control input
f	system dynamics
$\mathcal{X}$	state space
$\mathcal{U}$	action space
$\mathbb{R}^n$	n-dimensional Euclidean space
Φ_i	state-feedback controller i
f_i	dynamics of closed-loop system with controller Φ_i
x_i^*	equilibrium of closed-loop system f_i
$\mathcal{G}(\Phi_i)$	goal set of controller Φ_i
$\mathcal{D}(\Phi_i)$	DoA of controller Φ_i
t	time
x_0	initial state
$L(x)$	Lyapunov function
$\succeq$	prepare relation
s_i	mode i of control automaton
$\mathcal{S}$	finite set of discrete modes in control automaton
$\mathcal{D}(\Phi)$	union set of the DoAs of all existing controllers
M	Markov decision process
$\bar{f}$	state transition function
ρ	reward function
r_k	scalar reward at state x_k
T_s	sampling time
π	optimal policy
$V^\pi(x_k)$	discounted sum of expected instantaneous rewards at state x_k
γ	discount factor
$\hat{V}(x,\theta)$	approximated parameterized critic
θ	parameter vector for approximating critic
$\Psi_{\mathrm{c}}(x)$	basis function vector for approximating critic
$\hat{\pi}(x,\mu)$	approximated parameterized actor

μ	parameter vector for approximating actor
$\Psi_{\mathrm{a}}(x)$	basis function vector for approximating actor
δ	temporal difference
α_{c}	critic learning rate
∇_θ	partial differential with respect to θ
e_k	eligibility trace at time instance k
λ	trace decay rate
sat	saturation function
Δu_k	zero-mean white Gaussian noise at time instance k
α_{a}	actor learning rate
n_{t}	number of trials
n_{s}	number of samples
$J(x)$	skew-symmetric interconnection matrix
$R(x)$	symmetric dissipation matrix
$H(x)$	system Hamiltonian
$g(x)$	input matrix
y	collocated output
$H_{\mathrm{d}}(x)$	desired closed-loop Hamiltonian
x^*	equilibrium in which $H_{\mathrm{d}}(x)$ has a local minimum
u_{es}	energy shaping term of control law
u_{di}	damping injection term of control law
$H_{\mathrm{a}}(x)$	added energy term
$K(x)$	symmetric positive semi-definite damping injection matrix
$g^\perp(x)$	left annihilator matrix of $g(x)$
$\hat{H}_{\mathrm{d}}(x,\xi)$	approximated parameterized desired Hamiltonian
H_{di}	damping injection term of Hamiltonian
H_{es}	energy shaping term of Hamiltonian
ξ	parameter vector for approximating H_{es}
$\Psi_{\mathrm{es}}(x)$	basis function vector for approximating H_{es}
$g^\dagger(x)$	pseudo inverse matrix of $g(x)$
$F(x)$	system matrix
$\hat{K}(x,\psi)$	approximated parameterized damping injection matrix
ψ	parameter vector for approximating $\hat{K}(x,\psi)$
$\Psi_{\mathrm{di}}(x)$	basis function vector for approximating $\hat{K}(x,\psi)$
$J_{\mathrm{d}}(x)$	desired skew-symmetric interconnection matrix
$R_{\mathrm{d}}(x)$	desired symmetric dissipation matrix
$F_{\mathrm{d}}(x)$	desired system matrix
q	generalized position
p	generalized momentum
$\bar{n}$	$2\bar{n} = n$ where n is system dimension
x_{d}	desired state
$M(q)$	positive-definite mass-inertia matrix
Λ	positive-definite scaling matrix

$F_{ij}(x)$	entity ij of system matrix
ϑ	parameter vector for approximating elements of system matrix
$\Psi_{\mathrm{al}}(x)$	basis function vector for approximating elements of system matrix

Chapter 3

$\mathcal{M}$	closed invariant set including system origin
$\bar{x}^*$	non-zero equilibrium
$\bar{z}$	alternative variable for x
$N(x)$	numerator of rational Lyapunov function
$D(x)$	denominator of rational Lyapunov function
$R_i(x)$	homogeneous polynomial function of degree i representing $N(x)$
$Q_i(x)$	homogeneous polynomial function of degree i representing $D(x)$
$\mathcal{L}(c)$	sublevel set of Lyapunov function $L(x)$
c_*	maximum DoA level
$\mathcal{H}(x)$	set of states where $\dot{L}(x) < 0$ along with $\{0\}$
$\hat{c}_*$	upper bound of c_* in memoryless sampling
$\underline{c}_*$	lower bound of c_* in sampling with memory
$\bar{c}_*$	upper bound of c_* in sampling with memory
$\bar{\mathcal{E}}$	array of possible levels for DoAs approximated

Chapter 4

q	discrete state in hybrid automaton
H	hybrid automaton
Q	finite set of discrete states
X	set of continuous states
$\mathcal{F}$	vector field
Init	set of initial states
Inv	set of invariants
E	set of edges
G	guard condition
R	reset map
$\mathcal{P}(X)$	power set of X
A_ℓ	learning control automaton
ϵ	empty mode
s_ℓ	learning mode
s_q	mode q of control automaton
e_ℓ	learning event
e_p	event p of control automaton

(s_0, x_0)	initial mode
Φ	set of controllers
Φ_ℓ	learning controller
F	vector field
D	assigns to each mode its associated controller's DoA
G	assigns to each mode its associated controller's goal set
g	discrete event transition
Γ	active event function
$P(x_i, x_j)$	binary function that returns true or false
S_{ref}	reference event signal
x_0^i	initial state while controller Φ_i is active
x_{d}^i	desired state while controller Φ_i is active
$\bar{\mathcal{X}}$	desired range of state space to be controlled
x_0^*	overall desired equilibrium
$\bar{f}$	translated dynamics
$\bar{x}^*$	actual equilibrium
β	minimum distance between x^* and $\bar{x}^*$
c^*	minimum DoA level
$\mathcal{C}$	list of controllers
k	number of controller
c_k	DoA level of learned controller
δ	boundary of the DoA estimate
z	state chosen in a δ-boundary of the DoA

Chapter 5

x_i	state vector of system i
u_i	control input of system i
f_i	dynamics of system i
$\mathcal{X}_i$	state space of system i
$\mathcal{U}_i$	action space of system i
Φ_i^p	controller p of system i
Φ_i	set of controllers in system i
s_i^p	mode p of system i
$\mathcal{S}_i$	finite set of modes in system i
$L_i(x_i)$	Lyapunov function of system i
$\bar{\Phi}_{i,j,\ldots,r}^{p,q,\ldots,\ell}$	general form of composed controller
$\bar{A}$	general cooperative control automaton
A	initial control automaton
$\bar{E}$	finite set of composed events
$\bar{e}_k$	composed event k

$\bar{\mathcal{S}}_{i,j}$	set of modes in cooperative control automaton of systems i and j
(s_i^p, s_j^q)	composed mode in cooperative control automaton of systems i and j
$\bar{\Phi}_{i,j}^{p,q}$	composed controller generated by Φ_i^p and Φ_j^q
s_i^n	last mode of system i
s_j^m	last mode of system j
$\bar{x}$	composed state vector
$h_i(\bar{x})$	constraint equation of system i
$\bar{\Phi}_{i,j}$	set of composed controllers generated by systems i and j
$\bar{L}(\bar{x})$	composed Lyapunov function
$\bar{A}_{i,j}$	cooperative control automaton of systems i and j
$\bar{\mathcal{S}}_i$	subset of modes of system i
$\bar{E}_i$	set of composed events associated with $\bar{\mathcal{S}}_i$
E_i	set of individual events of system i
E_i'	set of individual and composed events of system i
$\bar{\sigma}$	elements of set $\bar{\Phi}_{i,j}$

Chapter 6

R_i	robot i
R	set of robots
O_i	object i
O	set of objects
S_i	surface i
S	set of surfaces
$\vert$	operation or
$\wedge$	operation and
N_i	contact configuration i
$\mathcal{N}$	set of valid contact configurations
$\mathcal{Q}$	set of other contact configurations present in a node
$\mathcal{P}$	set of contact configurations between all robots with an specific object
X_i	one of the components robot, object, or surface
$\vert\vert$	operation parallelization
$E_{i,j}$	edge from node i to node j in contact graph
$\mathcal{E}$	set of all possible edges in contact graph
w_i	weight of edge i in contact graph
$\mathcal{W}_i^X$	workspace of X_i
x_1	x position of the object in Cartesian space
x_2	y position of the object in Cartesian space
x_3	z position of the object in Cartesian space
y_0	initial y position of the robot base with respect to inertial frame
z_0	initial z position of the robot base with respect to inertial frame

r	radius of the robotic arm in extended configuration
$\mathcal{W}_j^N$	workspace of node N_j of contact graph
$\mathcal{W}_{k,l}^E$	workspace of edge $E_{k,l}$ of contact graph
$\Lambda(x)$	robot inertia
$\mu(x, \dot{x})$	Coriolis term in robot dynamic equation
F_{g}	gravitational matrix in robot dynamic equation
F_τ	end-effector input forces in Cartesian space
F_{ext}	end-effector external forces in Cartesian space
$\tilde{x}$	end-effector position error vector
D_{d}	damping parameter of the virtual spring
K_{d}	stiffness of the virtual spring
q	joint variable
τ	joint input torque
$J(q)$	Jacobian matrix of robot

List of Abbreviations

DoA	domain of attraction
RL	reinforcement learning
LTL	linear temporal logic
TD	temporal difference
PBC	passivity-based control
PH	port-Hamiltonian
DI	damping injection
EB-DI	energy balancing and damping injection
IDA-PBC	interconnection and damping assignment passivity-based control
EB-AC	energy balancing actor-critic
A-IDA-AC	algebraic interconnection and damping assignment actor-critic
SOS	sum of squares
PLTs	probabilistic learning trees
AI	artificial intelligent
PDDL	planning domain definition language
STRIPS	Stanford research institute problem solver
RRTs	rapidly exploring random trees
VREP	virtual robotics experimentation platform

Summary

Automatic Synthesis of Supervisory Control Systems

Esmaeil Najafi

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. Sequential composition focuses on the interaction between a collection of pre-designed controllers, where each of them is associated with a domain of attraction (DoA) and a goal set. By design, if the goal set of one controller lies in the DoA of another controller (this is called the prepare relation), the supervisor can instantly switch from the first controller to the second without affecting the stability of the system. As these controllers are designed offline, sequential composition cannot address unmodeled situations that might occur during runtime. Moreover, sequential composition has not been developed for cooperative settings where the collaboration of multiple systems is required in order to fulfill the control specifications.

This thesis studies automatic synthesis of supervisory control systems using the framework of sequential composition. First, a learning sequential composition control algorithm is developed so as to learn new controllers on demand, by means of reinforcement learning (RL). Once learning is completed, the supervisory control structure is augmented with the learned controllers. As a consequence, the supervisor is able to cope with unforeseen situations for which new controllers are required. Second, a cooperative sequential composition control algorithm is proposed to enable the coordination between a set of sequential composition controllers, without any change in their low-level structures. Finally, a robot contact language is designed for the manipulation of multiple objects by multiple robots. The main technical contributions of this thesis are outlined as follows.

Estimating the domain of attraction. To design a supervisory controller in the context of sequential composition, the DoAs of the low-level controllers and their goal sets have to be known. In this thesis, a fast sampling method is proposed for estimating the DoAs of nonlinear systems. This procedure is computationally effective, compared with the existing optimization-based techniques, and is useful for real-time applications. The sampling approach proposed has been used to

estimate the DoAs of stable equilibria in several nonlinear systems. Moreover, it has been applied to a passivity-based learning controller designed for a magnetic levitation system.

Learning sequential composition control. A learning control approach is proposed to enable the automatic synthesis of supervisory controllers. It augments the given pre-designed control system by learning new controllers online and on demand, using the actor-critic RL method. The learning process is always safe since the exploration in the course of learning the new controller only takes place within the DoAs of the existing controllers. The proposed approach has been implemented on two nonlinear systems: nonlinear mass-damper system and under-actuated inverted pendulum. This learning control technique has also been extended for situations where no controller exists initially and all controllers have to sequentially be synthesized so as to achieve the control objective. This algorithm is demonstrated on a simulated example of mobile robot navigation.

Cooperative sequential composition control. Multiple sequential composition controllers are composed via a cooperative sequential composition control technique to accomplish collaborative behavior. The sequential composition controllers communicate with each other to share their corresponding system dynamics and low-level structures. Using this data along with the interaction dynamics enables the computation of the DoAs of the resulting composed controllers. Based on the prepare relation, defined between the obtained DoAs and the goal sets, the original supervisors are augmented with new connections through their low-level controllers. The proposed control algorithm has been applied to the collaboration of an inverted pendulum with two second-order DC motors.

Robot contact language. This thesis concludes by studying the synthesis of supervisory control systems for robotic planning and manipulation. The problem of dividing a manipulation task is addressed to obtain an appropriate sequence of subtasks with regard to the contact-based task division. A robot contact language is defined for robotic manipulation based on making and breaking contact between the components involved, namely robots, objects, and surfaces. This planner is modular enough to deploy geometrical and physical information of the components and translate supervisory planning to low-level robot controllers. This robot language is validated in three case studies, each with a specific control objective.

All the above-mentioned contributions enable the automatic synthesis of a class of supervisory control systems that employ the paradigm of sequential composition.

Summary in Dutch

Automatische Synthese van Supervisie-Regelsystemen

Esmaeil Najafi

Sequentiële compositie is een effectieve supervisie-regelmethode voor het aanpakken van regelproblemen van niet lineaire dynamische systemen. Het gebruikt sequentieel een set regelaars toe om een regelspecificatie te realiseren, die niet te realiseren is met een enkele regelaar. Sequentiële compositie focust op de interactie tussen een collectie van voor-ontworpen regelaars, waarbij iedere regelaar is geassocieerd met een domein van aantrekking (DoA) en een doel set. Deze sets zijn zo ontworpen, zodat als de doel set van een regelaar in het DoA van een andere regelaar ligt (dit is de voorbereidende relatie genaamd), de supervisor instantaan schakelt van de eerste regelaar naar de tweede regelaar, zonder de stabiliteit van het systeem te beïnvloeden. Gezien deze regelaars offline ontworpen zijn, kan sequentiële compositie geen rekening houden met niet-gemodelleerde situaties die zich mogelijk voordoen tijdens de looptijd. Bovendien is sequentiële compositie nog niet ontwikkeld voor coöperatieve opstellingen, waar de samenwerking van meerdere systemen gewenst is voor het voldoen van de regeling specificaties.

In dit proefschrift wordt de automatische synthese van supervisie-regelsystemen bestudeerd met behulp van het framework van sequentiële compositie. Allereerst zal een zelflerend sequentieel compositie regelalgoritme worden ontwikkeld voor het leren van nieuwe regelaars op aanvraag, door middel van reinforcement learning (RL). Zodra het leren afgerond is, zal de supervisie-regelaar structuur worden aangevuld worden met de geleerde regelaars. Als gevolg kan de supervisor omgaan met onvoorziene situaties, waarvoor nieuwe regelaars nodig zijn. Ten tweede, een coöperatieve sequentiële compositie regelalgoritme is voorgesteld om de coördinatie tussen een set sequentiële compositie regelaars mogelijk te maken, zonder enige wijziging in hun low-level structuur. Tot slot wordt er een robot contact taal ontworpen voor de manipulatie van meerdere objecten door meerdere robots. De belangrijkste technische bedrijven van dit proefschrift worden als volgt uiteengezet.

Het schatten van het domein van attractie. Voor het ontwerpen van een supervisie regelaar in de context van een sequentiële compositie, moeten de DoAs en doel sets

van de low-level regelaars bekend zijn. In dit proefschrift wordt een snelle sampling methode voorgesteld ter schatting van de DoAs van niet-lineaire systemen. Deze procedure is computationeel effectief vergeleken met bestaande optimalisatie gebaseerde technieken en is geschikt voor real-time applicaties. De voorgestelde sampling aanpak is gebruikt voor het schatten van DoAs van stabiele evenwichten in verschillende niet-lineaire systemen. Bovendien is het toegepast op een passiviteit-gebaseerde zelflerende regelaar voor een magnetisch levitatie systeem.

Zelflerend sequentieel compositie regeling. Een zelflerend regel aanpak is voorgesteld voor automatische synthese van supervisie regelaars. Het vult de voor-ontworpen regelsystemen aan door nieuwe regelaars online en op aanvraag te leren, met behulp van de actor-critic RL methode. Het leerproces is altijd veilig, gezien de exploratie gedurende het leren van de nieuwe regelaar alleen plaats vindt in de DoAs van de bestaande regelaars. De voorgestelde aanpak is geïmplementeerd in twee niet-lineaire systemen: een niet-lineaire massa-demper systeem en een onder-geactueerde geïnverteerde pendulum. Deze zelflerende techniek is ook uitgebreid voor situaties waarin initieel geen regelaar bestaat en alle regelaars sequentieel gesynthetiseerd moeten worden om de regeldoelstelling te halen. Dit algoritme is gedemonstreerd op een gesimuleerd voorbeeld van mobiele robot navigatie.

Coöperatieve sequentiële compositie regeling. Meerdere sequentiële compositie regelaars worden samengesteld door middel van een coöperatieve sequentiële compositie regeltechniek om samenwerkend gedrag te bewerkstelligen. De sequentiële compositie regelaars communiceren onderling om hun bijbehorende systeem dynamica en low-level structuren te delen. Deze data wordt samen met de interactie dynamica gebruikt voor de berekening van de DoAs van de resulterende samengestelde regelaars. Gebaseerd op de voorbereidende relatie tussen de verkregen DoAs en de doel sets worden de originele supervisors aangevuld met nieuwe connecties door hun low-level regelaars. Het voorgestelde regel algoritme is toegepast op de samenwerking van geïnverteerde pendulums met twee tweede-orde dc-motoren.

Robot contact taal. Dit proefschrift wordt geconcludeerd met de studie naar de synthese van supervisionaire regelsystemen voor robotische planning en manipulatie. Het probleem van het onderverdelen van manipulatie taken wordt aangepakt om een geschikte volgorde van deeltaken te verkrijgen met betrekking tot de contact-gebaseerde deeltaken. Een robot contact taal is gedefinieerd voor robotische manipulatie, gebaseerd op het maken van verbreken van contact tussen de betrokken componenten, namelijk robots, objecten en oppervlakken. Deze planner is modulair genoeg om geometrische en fysische informatie van de componenten te gebruiken en om supervisonary planning om te zetten naar low-level robot regelaars. Deze robot taal is gevalideerd in drie casestudies, ieder met een specifieke regelaar doelstelling.

Alle bovengenoemde contributies maken automatische synthese van een klasse van supervisory regelsystemen mogelijk die gebruik maken van het paradigma van sequentiële compositie.

Summary in Persian

طراحی اتوماتیک سیستم‌های سوپروایزری کنترل

اسماعیل نجفی

روش ترکیب متوالی کنترلرها یکی از متدهای مناسب در سیستم‌های سوپروایزری کنترل برای بررسی مسائل کنترلی سیستم‌های دینامیک غیرخطی می‌باشد. در این روش مجموعه‌ای از کنترلرها بصورت سلسله‌وار برای کسب هدف کنترلی فعال می‌شوند، در حالیکه ارضای هدف کنترلی معمولا با بکارگیری تنها یک کنترلر پیچیده امکان‌پذیر نمی‌باشد. روش ترکیب متوالی کنترلرها به بررسی تعامل میان مجموعه‌ای از کنترلرهای از پیش طراحی شده می‌پردازد بطوری که هر یک از این کنترلرها دارای یک محدوده جذب و یک نقطه تعادل می‌باشد. اگر نقطه تعادل یکی از کنترلرها در محدوده جذب کنترلر دیگری قرار گیرد (رابطه ترتیب)، سیستم سوپروایزری سریعاً از کنترلر اول به کنترلر دوم سوئیچ خواهد کرد، بدون اینکه پایداری سیستم تحت تاثیر قرار گیرد. از آنجا که این کنترلرها از پیش طراحی شده‌اند، روش ترکیب متوالی کنترلرها توانایی مدیریت موقعیت‌های تصادفی که ممکن است در حین کارکرد سیستم اتفاق بیفتد را ندارد. بعلاوه روش ترکیب متوالی کنترلرها در حال حاضر برای سیستم‌های همکار که نیاز به همکاری چندین سیستم برای کسب هدف کنترلی می‌باشد، ارائه نشده است.

در این رساله طراحی اتوماتیک سیستم‌های سوپروایزری کنترل با بکارگیری چهارچوب روش ترکیب متوالی کنترلرها مورد مطالعه قرار گرفته است. در ابتدا یک الگوریتم کنترلی ترکیب متوالی یادگیرنده ارائه می‌شود بطوری که توانایی یادگیری کنترلرهای جدید را براساس نیاز و با استفاده از روش‌های یادگیری ماشین دارا می‌باشد. هنگامیکه فرآیند یادگیری به اتمام می‌رسد، کنترلرهای جدید به ساختار سیستم سوپروایزری کنترل اضافه می‌شوند. در نتیجه سیستم کنترلی قادر خواهد بود تا شرایط غیر قابل پیش‌بینی را که نیاز به کنترلرهای جدید است را مدیریت کند. در ادامه یک الگوریتم کنترلی ترکیب متوالی همکار ارائه می‌شود که قابلیت ایجاد همکاری میان مجموعه‌ای از کنترلرهای ترکیب متوالی را دارا می‌باشد، بدون اینکه تغییری در ساختار داخلی هیچ یک از کنترلرها ایجاد شود. در نهایت یک زبان رباتیکی با رویکرد تماس، به منظور تعامل مجموعه‌ای از ربات‌ها با مجموعه‌ای از اشیا، ارائه می‌شود. مهمترین دستاوردهای این رساله به شرح ذیل می‌باشند.

***تخمین محدوده جذب:* برای طراحی یک سیستم سوپروایزری کنترل در چهارچوب روش ترکیب متوالی کنترلرها نیاز است که محدوده جذب و نقطه تعادل هر یک از کنترلرها تعیین گردد. در این رساله یک روش نمونه‌برداری سریع برای تخمین محدوده جذب سیستم‌های غیرخطی ارائه می‌شود. این روش از نظر محاسباتی در مقایسه با روش‌های موجود بهینه‌سازی**

بسیار سریع‌تر بوده و برای کاربردهای زمان واقعی بسیار مناسب است. چگونگی عملکرد روش ارائه شده در تخمین محدوده جذب نقاط تعادل، در سیستم‌های غیرخطی مختلفی مورد بررسی قرار گرفته است. همچنین این روش برای محاسبه محدوده جذب کنترلری که با رویکرد انرژی برای سیستم تعلیق مغناطیسی طراحی شده، بکار گرفته شده است.

سیستم کنترلری ترکیب متوالی یادگیرنده: یک روش یادگیری ماشین ارائه شده است که امکان طراحی اتوماتیک سیستم‌های سوپروایزری کنترل را فراهم می‌کند. در این روش کنترلرهای از پیش طراحی شده با کنترلرهایی که بصورت هم‌زمان و براساس نیاز با استفاده از روش یادگیری ماشین عامل-منتقد یاد گرفته شده‌اند، ترکیب می‌شوند. فرآیند یادگیری کنترلرهای جدید همواره پروسه‌ای ایمن می‌باشد، چرا که جستجو جهت یادگیری کنترلر جدید تنها در محدوده جذب کنترلرهای موجود صورت می‌گیرد. روش ارائه شده بر روی دو سیستم جرم و دمپر غیرخطی و نیز پاندول معکوس پیاده‌سازی شده است. این تکنیک کنترلی یادگیرنده برای موقعیت‌هایی که هیچ کنترلری از ابتدا وجود نداشته نیز گسترش یافته است. در روش ارائه شده تمامی کنترلرها از ابتدا به شکلی بصورت سلسله‌وار طراحی می‌شوند که در نهایت امکان کسب هدف کنترلی فراهم شود. چگونگی عملکرد این الگوریتم کنترلی جهت هدایت و کنترل یک ربات متحرک مورد مطالعه قرار گرفته است.

سیستم کنترلری ترکیب متوالی همکار: چندین کنترلر ترکیب متوالی در چهارچوب یک تکنیک ترکیب متوالی همکار به تعامل پرداخته تا توانایی کسب اهداف کنترلی که نیاز به همکاری هم‌زمان چندین سیستم می‌باشند، فراهم گردد. کنترلرهای ترکیب متوالی با یکدیگر ارتباط برقرار کرده تا دینامیک سیستم مربوطه و ساختار کنترلی خود را به اشتراک بگذارند. بکارگیری این اطلاعات به همراه دینامیک تعامل میان سیستم‌های تعامل کننده، امکان محاسبه محدوده جذب کنترلرهای ترکیبی ایجاد شده را فراهم می‌کند. بر اساس رابطه ترتیب که بین محدوده‌های جذب و نقاط تعادل بدست آمده تعریف می‌شود، رابطه‌های جدیدی میان کنترلرهای هر یک از سیستم‌ها ایجاد می‌شوند که به سیستم سوپروایزری کنترل ابتدایی اضافه می‌گردند. الگوریتم کنترلی ارائه شده جهت همکاری یک پاندول معکوس و دو موتور DC درجه دو بکار گرفته شده است.

زبان ربات با رویکرد تماس: این رساله با مطالعه طراحی سیستم‌های سوپروایزری کنترل به منظور برنامه‌ریزی و تعامل در سیستم‌های رباتیکی به اتمام می‌رسد. مساله تقسیم یک وظیفه تعاملی به یک سلسله زیر وظیفه تعاملی با رویکرد تماس میان اجزای تعامل‌کننده، مورد بررسی قرار گرفته است. یک زبان رباتیکی با هدف تعامل میان سیستم‌های رباتیکی، براساس ایجاد و قطع تماس میان اجزای تعامل‌کننده که شامل ربات‌ها، اشیا و سطوح می‌باشند، تعریف شده است. این زبان برنامه‌ریز قادر است که اطلاعات فیزیکی و هندسی اجزای تعامل‌کننده را بکار گرفته و برنامه‌ریزی سوپروایزری را در قالب اهداف کنترلی سطح پایین برای ربات‌ها تعریف نماید. عملکرد مناسب این زبان رباتیکی در قالب سه مساله مختلف، هر یک با هدف کنترلی خاص، مورد بررسی قرار گرفته است.

به کمک دستاوردهای ارائه شده در این رساله، امکان طراحی اتوماتیک کلاس خاصی از سیستم‌های سوپروایزری کنترل که از روش کنترلرهای ترکیب متوالی بهره می‌گیرند، فراهم آمده است.

List of Publications

Journal Papers

1. E. Najafi, R. Babuska, and G. A. Lopes, "Learning sequential composition control", to appear in *IEEE Transactions on Cybernetics*, 2015.

2. E. Najafi, R. Babuska, and G. A. Lopes, "A fast sampling method for estimating the domain of attraction", submitted to *Nonlinear Dynamics*, 2015.

3. A. Shah, E. Najafi, and G. A. Lopes, "A robot contact language for manipulation planning", submitted to *IEEE/ASME Transactions on Mechatronics*, 2016.

Book Chapter

1. G. A. Lopes, E. Najafi, S. P. Nageshrao, and R. Babuska, "Learning complex behaviors via sequential composition and passivity-based control", *Handling Uncertainty and Networked Structure in Robot Control*, L. Busoniu and L. Tamas Eds., pp. 53–74, Springer, 2015.

Conference Proceedings

1. E. Najafi, G. A. Lopes, and R. Babuska, "Reinforcement learning for sequential composition control", in *Proceedings of the 52nd IEEE International Conference on Decision and Control*, pp. 7265–7270, Florence, Italy, Dec. 2013.

2. E. Najafi, G. A. Lopes, and R. Babuska, "Balancing a legged robot using state-dependent Riccati equation control", in *Proceedings of the 19th IFAC World Congress*, pp. 2177–2182, Cape Town, South Africa, Aug. 2014.

3. E. Najafi, G. A. Lopes, S. P. Nageshrao, and R. Babuska, "Rapid learning in sequential composition control", in *Proceedings of the 53rd IEEE International Conference on Decision and Control*, pp. 5171–5176, Los Angeles, California, USA, Dec. 2014.

4. E. Najafi, R. Babuska, and G. A. Lopes, “An application of sequential composition control to cooperative systems", in *Proceedings of the 10th International Workshop on Robot Motion and Control*, pp.15-20, Poznan, Poland, July 2015. **Candidate for the Young Author Best Paper Award (selected top 5)**.

5. A. Shah, E. Najafi, and G. A. Lopes, “Contact-Based Language for Robotic Planning and Manipulation", submitted to *the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Daejeon, Korea, Feb. 2016.

6. E. Najafi, and G. A. Lopes, “Towards Cooperative Sequential Composition Control", submitted to *the the 55th IEEE International Conference on Decision and Control*, Las Vegas, NV, USA, Mar. 2016.

Peer-Reviewed Abstracts

1. E. Najafi, G. A. Lopes, and R. Babuska, “Automatic Synthesis of Sequential Composition of Controllers", in *Book of Abstracts of the 32nd Benelux Meeting on Systems and Control*, Houffalize, Belgium, p. 138, Mar. 2013.

2. E. Najafi, G. A. Lopes, and R. Babuska, “Domains of attraction of learning controllers in adaptive sequential composition", in *Book of Abstracts of the 33rd Benelux Meeting on Systems and Control*, Heijen, the Netherlands, p. 72, Mar. 2014.

About the Author

Esmaeil Najafi was born on 11 August 1985 in Tehran, Iran. He received his BSc in Mechanical Engineering from K. N. Toosi University of Technology, Tehran, Iran in 2007. He did his BSc thesis titled Design a Fuzzy Controller for Vehicle Power Transmission System under the supervision of Prof. Ali Ghaffari and Dr. Reza Kazemi. He continued his education at the same university and obtained his MSc in Mechanical Engineering, Dynamics and Control in 2010. He did his MSc thesis titled Impedance Control of Stewart Platform for Adaptive Tracking in Rehabilitation under the supervision of Dr. Ali Nahvi and Dr. Farid Najafi. Having excellent teaching skills, he was active in lecturing mechanical engineering and control theory courses to undergraduate students during the years 2009-2011.

Since November 2011, Esmaeil has been working on his PhD project titled Automatic Synthesis of Supervisory Control Systems at the Delft Center for Systems and Control, Delft University of Technology, the Netherlands. He did his PhD under the supervision of Prof. dr. Robert Babuška and Dr. Gabriel A.D. Lopes. His PhD research has mainly dealt with supervisory control systems, machine learning, robotics, and mechatronics. Esmaeil Najafi obtained the DISC certificate for fulfilling the graduate course program of the Dutch Institute of Systems and Control. He also received the certificate of the TU Delft Graduate School for fulfilling the Doctoral Education Program.

During his PhD, Esmaeil supervised an MSc thesis at Delft University of Technology and was teaching assistant for the graduate course Control Methods for Robotics for two consecutive years. In addition, he has been teaching the undergraduate courses Robot Control, Practical Intelligent Methods, and Linear Algebra at The Hague University of Applied Sciences as a mechatronics lecturer and supervising four BSc theses in the fields of control, robotics, and mechatronics.

Esmaeil Najafi's research interests include supervisory control systems, switching control systems, hybrid systems, machine learning, robotics, mechatronics, and control of energy systems.

E-mail: najafi.e@gmail.com