ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 36

Settore Concorsuale: 09/G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF/04 - AUTOMATICA

LEARNING-DRIVEN AND DISTRIBUTED OPTIMAL CONTROL METHODS FOR LARGE-SCALE AND MULTI-AGENT SYSTEMS

Presentata da: Lorenzo Sforni

Coordinatore Dottorato

Michele Monaci

Supervisore

Giuseppe Notarstefano

Co-supervisore

Ivano Notarnicola

Esame finale anno 2024

Abstract

In recent years, the interest of the control community in large-scale systems has surged, driven by their capacity to encompass behaviors that integrate human, cyber, and physical resources. The inherent complexity of these dynamical systems, characterized by several interconnected units, represents a significant challenge when developing effective optimal control policies. This timely topic is addresses in this thesis from a threefold perspective: (i) tackling the complex nature of such settings by leveraging data-driven strategies over traditional model-based approaches, (ii) taking advantage of the patterns and interconnections typical of these large-scale applications at design level, and (iii) developing efficient and scalable centralized and distributed optimal control algorithms.

We start by developing *data-driven* strategies for the resolution of the Linear Quadratic Regulator (LQR) problem when the underlying dynamics are unknown. First, we propose an on-policy algorithm, where the system matrices are identified while simultaneously optimizing the control policy. Second, we design data-driven algorithms to generate feedback controllers that adhere to particular *structural constraints*, such as mirroring the inherent patterns of the large-scale system.

Subsequently, we develop a novel *first-order* framework for nonlinear optimal control tailored to large-scale systems. This centralized methodology is then extended to the *distributed optimal control* setting. Here, we tackle a novel problem formulation that leverages the aggregative optimization framework, an approach designed to encapsulate collective (aggregate) behaviors. We then propose a *learning-driven* version of the approach, based on a concurrent optimization and learning scheme. Eventually, the proposed methodology is extended to the *stochastic optimal control* scenario via a stochastic gradient descent scheme.

Finally, we consider the challenge of developing safe-by-design control policies within the *multi-layer control* formalism. This novel framework is tailored to analyzing the control architectures traditionally deployed in robotics applications, where a low-level, easy-to-deploy control policy interacts with a more advanced high-level planning strategy. In this context, we present an optimization-based trajectory generation strategy, ensuring compliance to *safety-critical constraints*, specifically designed for multi-layer control architectures.

Keywords: Large-scale Systems, Data-driven Control, First-order Methods, Reinforcement Learning, Gaussian Processes, Optimal Control, Nonlinear Systems, Linear Quadratic Regulator, Multi-layer Control Architectures, Safety-critical Control.

Contents

Notation

Given a matrix M, its right-pseudo inverse is $M^\dagger$. Given a square matrix M, its Frobenius norm is $\|M\|_F$ while $\mathrm{Tr}[M]$ is its trace. If M is positive definite (resp. semi-definite) we write $M > 0$ (resp. $M \geq 0$). Given two matrices $M_1 \in \mathbb{R}^{n\times n}$ and $M_2 \in \mathbb{R}^{n\times n}$, we define the inequality $M_1 \geq M_2$ sticking to the convention that their difference is a positive semidefinite matrix (i.e., $M_1 - M_2 \geq 0$). The $n \times m$ all-ones matrix is $\mathbf{1}_{n\times m}$, the $n \times m$ all-zeros matrix is $\mathbf{0}_{n\times m}$, the $n \times n$ identity matrix is I_n, and $\circ$ is the Hadamard product. The symbol $\otimes$ denotes the Kronecker product. We denote with $\mathrm{diag}(v_1, \ldots, v_n)$ the diagonal matrix whose i-th diagonal element is given by v_i. Given a symmetric, positive-definite matrix $Q \in \mathbb{R}^{n\times n}$, and $x \in \mathbb{R}^n$, we define the Q-norm of x as $\|x\|_Q = \sqrt{x^\top Q x}$ Given $x \in \mathbb{R}^n$ and $X \subseteq \mathbb{R}^n$, we define $w(x, X) := \inf_{y\in X} \|x - y\|$. Given d vectors $x_1 \in \mathbb{R}^{n_1}, \ldots, x_d \in \mathbb{R}^{n_d}$, their vertical stack is denoted by $\mathrm{col}(x_1, \ldots, x_d)$. Given a function $\ell : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_d} \to \mathbb{R}$, its (total) gradient at a given point $(\bar{x}_1, \ldots, \bar{x}_d)$ is denoted as $\nabla\ell(\bar{x}_1, \ldots, \bar{x}_d) := \mathrm{col}(\nabla_1\ell(\bar{x}_1, \ldots, \bar{x}_d), \ldots, \nabla_d\ell(\bar{x}_1, \ldots, \bar{x}_d))$, where $\nabla_k\ell(\bar{x}_1, \ldots, \bar{x}_d) \in \mathbb{R}^{n_k}$ denotes the partial derivative of ℓ with respect to its k-th variable for all $k \in \{1, \ldots, d\}$. We denote the i-th term of $\nabla_k\ell(\bar{x}_1, \ldots, \bar{x}_d) \in \mathbb{R}^{n_k}$, $i \in\in \{1, \ldots, n_k\}$ as $\left[\nabla_k\ell(\bar{x}_1, \ldots, \bar{x}_d)\right]_i$. Given a vector field $f : \mathbb{R}^n \to \mathbb{R}^m$, the gradient of f is $\nabla f(x) := [\nabla f_1(x) \ldots \nabla f_m(x)] \in \mathbb{R}^{n\times m}$. For a given $T \in \mathbb{N}$, we denote the discrete-time horizon as $[0, T] := \{0, 1, 2, \ldots, T\}$. A function $\ell : \mathbb{R}^n \to \mathbb{R}$ is said to be radially unbounded if $\forall c > 0$, $\exists r > 0$ such that, $\forall x \in \mathbb{R}^n$, $\|x\| > r \Rightarrow \ell(x) > c$. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with Ω sample space, $\mathcal{F}$ σ-algebra and $\mathbb{P}$ probability measure, we denote as w the random variable induced by the probability space, such that $\mathrm{w} : \Omega \to \mathbb{R}^p$. Each random variable is denoted in roman font. Each realization of w random variable is denoted as w. Given a random variable w we define the associated probability density function as p_{w}. With the symbol $w \sim p_{\mathrm{w}}$, we imply that w is a realization of the random variable w with probability density function p_{w}.

Introduction

Motivation and Challenges

The scope of control applications has continuously increased over the past few decades, moving from feedback control of a single device or system to optimization-based control and decision-making in large-scale systems, system-of-systems, and infrastructure systems [10]. These complex systems are of particular interest, due to their ability of capturing complex behaviors that integrate human, cyber and physical resources, e.g., sharing economy [67], smart infrastructure systems [204], and smart societies [200, 239].

The goal is to design control strategies to address the complexity of these large-scale systems while simultaneously optimizing a wider range of performance targets.
First, the uncertain, and complex nature of such problems makes the use of data-driven and learning-based control strategies appealing over traditional model-based approaches. The use of data for control design is gaining traction in the recent years thanks to the technological advantages which enabled the use of machine learning techniques in a range of previously inaccessible scenarios. See the recent surveys [72, 159, 196], for an overview of possible solutions and applications.
Second, the envisioned solutions should take advantage of the patterns and interconnections often arising in these large-scale dynamics. From a control point of view, this translates into the need for certain structural properties and limitations on the control policy. This topic is currently of high interest, e.g., for the design of fast and flexible controllers suited for multi-agent applications, e.g., in [89, 112, 146]. From an optimization perspective, the intrinsic interdependency among units in these large-scale systems paves the way to innovative problem formulations that accurately model collective behaviors and resource utilization. More in detail, it is worth considering the extension into the optimal control field of optimization frameworks capable of considering also some sort of aggregate information, e.g., the aggregative optimization framework [146], which proved to be capable of capturing a variety of applications, from cooperative robotics to network congestion control [21, 122, 176, 178].
Third, the proposed algorithmic strategies should exhibit beneficial scalability properties as the number of nodes and agents increases. Scalable optimization algorithms are a

timely research topic [102]. Among the others, first-order algorithms are well recognized effective tools in optimization over large-scale models, e.g., in neural network training, where the parameter tuning in performed via stochastic gradient descent [103]. The widespread diffusion of computationally enabled devices capable of cooperating by processing local data and information made attractive the alternative represented by the distributed computation paradigm. These solutions can be advantageous not only when the scale of the network and the number of agents make centralized approaches impracticable due to the amount of data to be shared, but also as a way to lower privacy risks when personal data and information are involved. This distributed computation model has been successfully addressed in a variety of frameworks, from consensus algorithms to distributed optimization and control. See, e.g., [165, 170, 244].

Additionally, this thesis investigates the design of optimization-based control policies within the multi-layer control framework. This emerging theoretical framework is specifically tailored to the analysis and development of rigorous control strategies intended for deployment in robotics applications, see, e.g., [68, 193]. In these practical scenarios, computational tractability and real-time implementation constraints often necessitate a layered architecture, where a low-level, real-time control policy interacts with a more advanced high-level planning strategy. Notably, these high-level approaches, while offering transformative potential, can introduce vulnerabilities leading to fragile failure modes, as discussed in recent surveys [10, 45]. This calls for the development of safe-by-design high-level control policies. In this thesis, we tackle this challenge through optimal control-based solutions.

Summary of Main Contributions

The methodologies developed in this thesis contribute to the field of optimal control of large-scale systems from a threefold perspective: (i) leveraging on data for optimal control design, (ii) developing control strategies specifically designed to take advantage of the patterns and interconnections typical of large-scale applications, (iii) designing scalable and efficient numerical optimal control algorithms, both in centralized and distributed settings. Furthermore, it is investigated the development of safe-by-design control strategies within the formalism of multi-layer control architectures, with a special focus on optimal control-based planning strategies.

First, this thesis studies the problem of designing control policies for unknown linear systems within the context of Linear Quadratic (LQ) optimal control. We start by proposing a data-driven on-policy strategy. The developed algorithmic solution is based on a simultaneous identification and optimization process, where the dynamics is identified via recursive least squares while the policy is improved using a gradient method. This results in a time-varying, iteratively updated, control policy, for which we proved convergence towards the optimal Linear Quadratic Regulator (LQR) solution. Afterwards, we address the challenge of designing data-driven control policies matching the inherent patterns of large-scale systems, e.g., mimic the communication graph of a network system. From an optimal control perspective, this results in the introduction of a structural (equality) constraint within the (unconstrained) LQR formulation. We remark that also in this setting, the dynamics is assumed to be unknown. This challenging scenario is tackled from a dual perspective, (i) we introduce a Reinforcement Learning framework based on a Q-learning scheme, where the desired policy structure is ensured via an LMI-based policy improvement step, (ii) we reformulate the gain design process as a direct policy search, proposing a data-driven approach for computing the optimal gain with the desired structure based on the combination of a gradient method and an augmented Lagrangian approach.

Second, the investigation shifts towards the design of numerical optimal control strategies for large-scale systems. Overall, we propose a computationally efficient first-order algorithmic framework, denoted as GoPRONTO, short for Generalized first-Order PROjectioN operator method for Trajectory Optimization. Initially formulated within a centralized, model-based context, this framework has been comprehensively extended to the challenging scenarios of distributed optimal control applications, and learning-based optimal control of partially unknown nonlinear systems. More in detail, the generalized GoPRONTO methodology consists of a class of first-order algorithms, that leverage the incorporation of a feedback policy into the problem formulation. This approach is referred to as the feedback embedding paradigm. Via this methodology, we can reinterpret the optimal control problem as a cost function minimization task, paving the way for advanced numerical optimal control strategies. These ideas are instrumental in extending the GoPRONTO approach to a distributed optimal control framework. We address the novel class of aggregative optimal control problems. Within this setting, the introduction of an aggregate optimization variable enables the modeling of behavioral patterns characteristic of multi-agent systems. The resulting strategy is a fully distributed optimal control algorithm that relies on inter-agent communication to efficiently reconstruct quantities that may not be available to individual agents. The GoPRONTO framework is further extended towards the design of optimal control strategies for partially unknown systems. Specifically, we introduce a data-driven optimal control strategy that assumes the ability to actuate control input sequences onto a real system while only an inaccurate description of the dynamics is available for the control design. This approach integrates trajectory optimization with Gaussian process regression, iteratively refining the model and performing optimization steps. Ultimately, the GoPRONTO framework is adapted for stochastic optimal control. The proposed strategy implements a Stochastic Gradient Descent (SGD) scheme, leveraging on the convenient reformulation enabled by the feedback embedding paradigm.

Finally, the focus moves on the design of safe trajectories for multi-layer control architectures. We propose an high-level, trajectory generation strategy, based on optimal control, that ensures satisfaction of safety-critical constraints. The constraints are enforced through Control Barrier Functions (CBFs), that represent the state-of-art approach for the generation of safe-by-design controllers. We extend the application of CBFs to nonlinear non-autonomous control systems, providing both sufficient and necessary safety guarantees, as CBFs are originally proven to ensure safety only for autonomous systems. As a result, the proposed multi-layer controller can simultaneously deliver optimal performance and ensure the satisfaction of safety constraints.

Organization and Chapter Contributions

The thesis organization follows the contribution scheme outlined in the previous section.

In Chapter 1, we formally present the optimal control frameworks central to this thesis. Initially, we define the Linear Quadratic Regulator (LQR) problem in the context of unknown dynamics, which is then specialized to generate a structured feedback policy. Next, we explore the nonlinear optimal control problem, considering its applications in large-scale systems, multi-agent scenarios, and applications with unknown dynamics. Lastly, we introduce the concept of safe optimal control.

In Chapter 2, we investigate the data-driven LQR framework, which we then extend to the constrained case. We start by developing an on-policy control scheme where the proposed policy is applied to the actual (unknown) linear system, while concurrently refined towards the optimal solution of the LQR problem. The proposed method relies on the so-called direct policy search reformulation of the LQR problem, which is an optimization problem with the control policy gain being the decision variable and parametrized in the system matrices. This optimization problem is addressed via a gradient-based method combined with an estimation procedure to deal with the missing knowledge of the system dynamics. In particular, the system matrices are progressively reconstructed via a Recursive Least Squares (RLS) mechanism that iteratively elaborates the state-input samples obtained from the actual, closed-loop system. The on-policy nature of the scheme stems from the fact that each state-input sample is gathered by actuating the (yet non-optimal) state feedback. To ensure persistency of excitation, a probing dithering signal is also fed into the (running) closed-loop dynamics. The stability properties of the resulting closed-loop system are by resorting to Lyapunov arguments and averaging theory for two-time-scale systems. We show the exponential stability of a properly defined steady state, in which: (i) the feedback policy is the optimal solution of the LQR problem; (ii) the estimates of the unknown matrices are exact; and (iii) the system state oscillates about the origin with an amplitude arbitrarily tunable setting the initial conditions magnitude of the dither. We then propose a Reinforcement Learning framework to solve the infinite-horizon LQR problem with unknown dynamics and with a user-defined sparsity of the feedback control law. To solve the problem, we propose a Q-learning approach able to enforce the desired controller structure. The developed algorithm has the form of a policy iteration scheme that progressively improves the cost by appropriate manipulation of the system Q-function. Specifically, at each time step we first solve a system of linear equations to evaluate the performance of a given controller and then solve a Linear Matrix Inequality (LMI) to compute a new feedback gain improving on the previous one and enforcing the desired sparsity. Under mild assumptions, we demonstrate that the developed algorithm produces at each iteration a feedback matrix with the imposed structure that also guarantees asymptotic stability of the system, and with a non-increasing associated Q-function. As a consequence, also every limit point of the computed feedback matrices is sparse and stabilizing. Finally, we leverage on a data-driven approach to design a structured, static linear feedback law, specifically tailored to LQR problems in which the system dynamics is unknown. We consider the LQR problem reformulated in terms of a direct policy search and include the sparsity constraints as an additional constraint in the resulting optimization problem. Hence, we tackle the challenge of gain design by casting it as a nonconvex constrained optimization problem. We propose an algorithm that combines an augmented Lagrangian method with a data-driven gradient method. The results of this chapter are based on [205, 206, 210].

In Chapter 3, we present a class of robustified first-order optimal control algorithms, specifically developed for large-scale systems. This methodology is also successfully extended to the design of optimal trajectories for multi-agent and uncertain nonlinear systems. We start by formally introducing this novel class of numerically-robust first-order algorithms for discrete-time optimal control, which we termed GoPRONTO. In our approach we combine the introduction of a nonlinear tracking system (the so-called embedded feedback) into the nonlinear optimal control problem, with a gradient-based resolution strategy. The innovative combination of these two approaches results in a novel first-order numerical optimization framework which enjoys several appealing features in the optimal control context. From the embedding of the feedback policy, our approach enjoys the highly desired properties of: (i) numerical robustness, even when dealing with unstable dynamics, and (ii) recursive feasibility, i.e., a state-input trajectory can be computed, at each iteration, via a closed-loop integration of the nonlinear dynamics. This property is of particular interest in the case of, e.g., unstable systems. From the gradient-based approach, our methodology inherits the simplicity of implementation of the descent-direction search, namely a costate equation update. This adaptable update rule makes our algorithmic strategy flexible enough to be extended to problems involving large-scale dynamics. As in other optimization domains with very large decision variables (e.g., neural network training) Newton's methods are impracticable while first-order approaches are preferred. We then show how this general framework gives rise to several first-order optimization algorithms that can speed up the resolution of the optimal control problem. Indeed, novel algorithms can be implemented by means of appropriate modifications in the structure of our algorithmic approach, e.g., by variations in the state-input curve update or by the evaluation of the costate equation and the system-linearization in different curves. We then develop a distributed scheme to address a novel class of nonlinear optimal control problems over networks of multi-agent systems. Such a class of problems is formulated by adopting the problem structure introduced in the context of aggregative optimization. We address the nonlinear aggregative optimal control problem according to a peer-to-peer paradigm that does not require any centralized unit accessing global data and making global decisions. Inspired by the aforementioned centralized GoPRONTO strategy, our multi-agent method uses, in each agent, a local feedback mechanism updated according to a distributed optimization-oriented scheme. Such an optimization scheme, due to the aggregative terms of the local cost functions, would require local knowledge of unavailable global quantities. Thus, we resort to a set of local, auxiliary variables named trackers that asymptotically reconstruct these global quantities through suitable consensus-based dynamics relying on inter-agent communication. Interlacing this mechanism with the optimization-oriented part of the algorithm gives rise to a complex interconnected system. In order to analyze its evolution, we exploit the adopted feedback-based algorithmic structure which allows for reformulating the multi-agent optimal control problem as an (unconstrained) nonconvex optimization problem. Moreover, the feedback-based nature ensures stability and, thus, allows for analyzing the overall distributed strategy with tools from system theory. In detail, we exploit LaSalle-based arguments to prove that the sequence of generated trajectories converges to the set of state-input trajectories satisfying the first-order necessary conditions for optimality of the considered optimal control problem. Then, we propose a learning-based optimization strategy to solve nonlinear finite-horizon optimal control problems with partially unknown dynamics. We propose a two-step iterative procedure, based on the embedded-feedback gradient-like update combined with a Gaussian process regression, in which the optimization process and the learning phase are concurrently performed while running experiments on the real system. Specifically, the unknown term in the dynamics is approximated through an iteratively refined Gaussian Process (GP) and at each iteration the gradient-like step is performed by taking derivatives of the nominal dynamics enhanced with the GP. The current optimal input estimate is then perturbed with the computed descent direction and actuated on the real system. During this experiment, novel measurements from the system evolution are collected and used in the learning phase. Under suitable technical conditions on the collected system trajectories, the proposed strategy is proved to converge to a neighborhood of a stationary point of the optimal control problem. In order to prove the result, the algorithmic updated is recast into a suitable gradient-with-error update with error uniformly bounded across iterations. Eventually, we extended the GoPRONTO framework of optimal control of stochastic systems. We propose a numerical procedure based on the Stochastic Gradient Descent algorithm. The descent direction is computed via a Gradient Estimator which is then showed to be unbiased. Convergence in expectation is show under mild assumptions. The results of this chapter are based on [208, 209, 212].

Finally, in Chapter 4, a continuous-time high-level planning approach tailored to multi-layer applications is proposed. Our method generates safe trajectories with adjustable time length, thus achieving balance between point-wise optimal safety filters and safe infinite-time optimal control. Since, safety is ensured through a CBF constraint, we provide a new characterization of CBFs as necessary and sufficient for the controlled invariance of safe sets. Utilizing this, we show that as the time horizon extends to infinity the constrained optimal control problem is retrieved, conversely as the time horizon is shrink to zero, a safety filter is obtained. The proposed strategy is then deployed within a multi-layer control architecture. The results of this chapter are based on [211].

As a complementary part of this thesis, we include Appendix A providing some basic concepts of numerical nonlinear optimization. Appendix B overviews few numerical methods for optimal control which laid the foundation of the algorithms developed in this work. Finally, Appendix C contains some useful results about averaging theory for for two-time-scale systems.

Chapter 1

Nonlinear and Learning-driven Optimal Control Frameworks

In this chapter, we introduce the different optimal control frameworks that are central to this thesis. We commence by defining the data-driven optimal control framework for unknown linear systems. This framework is then extended to the development of structured control policies for multi-agent and large-scale systems. Subsequently, we address scenarios where the dynamics are nonlinear, while specifying the optimal control problem to large-scale systems, multi-agent distributed control and data-driven control of unknown and uncertain dynamics. Finally, we introduce the concept of safe optimal control within a multi-layer control architecture.

1.1 Data-driven Optimal Control of Linear Systems

As a first setting, we consider an optimal control framework in which our objective is to design optimal feedback *policies*, i.e., feedback control laws, for unknown linear systems. The problem is formalized as the so-called Linear Quadratic Regulator (LQR) problem, a cornerstone within the control systems community. Recently, this setting has received renewed attention for its benchmark role in the development of innovative reinforcement learning and data-driven strategies. In this setting, we consider a dynamics which is linear and time-invariant, i.e., in the form

$$x_{t+1} = A_\star x_t + B_\star u_t, \qquad x_0 = x_{\text{init}} \tag{1.1}$$

where $x_t \in \mathbb{R}^n$ are $u_t \in \mathbb{R}^m$ the state and the input of the system at time t, $x_{\text{init}} \in \mathbb{R}^n$ represents the initial condition, and $A_\star \in \mathbb{R}^{n\times n}$ and $B_\star \in \mathbb{R}^{n\times m}$ are the state and input matrices, respectively. The matrices $A_\star$, $B_\star$ are assumed to be *unknown*. Formally, our goal it to solve the optimal control problem

$$\begin{aligned} \min_{x_t,u_t,K} \quad & \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t && (1.2a) \\ \text{subj.to} \quad & x_{t+1} = A_\star x_t + B_\star u_t && (1.2b) \\ & u_t = K x_t && (1.2c) \end{aligned}$$

where $K \in \mathbb{R}^{m\times n}$ is the feedback gain, and $Q \in \mathbb{R}^{n\times n}$, $Q = Q^\top \geq 0$ and $R \in \mathbb{R}^{m\times m}$, $R = R^\top > 0$, are the state and input cost matrices respectively. Notice that constraint (1.2c) is explicitly introduced to highlight that our focus is on the design of control policies among the class of linear state feedback laws, i.e., in the form

$$u_t = K x_t. \qquad (1.3)$$

Importantly, the presence of constraint (1.2c) makes Problem (1.2) nonconvex. Overall, under the challenging assumption that the dynamics (1.1) is unknown, our goal it so find the optimal feedback gain $K^\star$ solution of (1.2), such that (i) the closed-loop system is asymptotically stable, i.e., all the eigenvalues of the closed-loop matrix $(A_\star + B_\star K^\star)$ are within the unit disk, (ii) the infinite horizon cost functional (1.2a) is minimized.

Through this dissertation, we will also consider a variant of Problem (1.2) tailored to the design of control policies for large-scale and distributed control systems. Indeed, in these scenarios, the design of the feedback control law (1.3) often requires the feedback gain K to satisfy some structural constraints, e.g., have a sparse structure mirroring the system's communication network. Specifically, certain entries in K are required to be zero, aligning with structural and communication constraints within the system. More formally, in a large-scale, multi-agent control setting, we seek for a solution of the *constrained* infinite-horizon Linear Quadratic Regulator problem

$$\begin{aligned} \min_{x_t,u_t,K} \quad & \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t && (1.4a) \\ \text{subj.to} \quad & x_{t+1} = A_\star x_t + B_\star u_t && (1.4b) \\ & u_t = K x_t && \\ & K \circ \mathcal{S} = 0 && (1.4c) \end{aligned}$$

where the constraint (1.4c) is meant to enforce the desired structure on the gain K, with $\mathcal{S}$ being a binary matrix modeling the desired structural profile. The challenge of solving Problem (1.4) is twofold, (i) the unknown system dynamics hampers the applicability of model-based optimal control methods, (ii) the design of a sparse feedback gain K calls for different solution approaches.

Motivating example: Distributed Control of Sparse Dynamics

We now give an exemplary application of the general set-up (1.4), consider a network of N dynamical systems (or *agents*) that communicate according to an undirected graph $\mathcal{G} = (V, \mathcal{E})$, where $V = \{1, \ldots, N\}$ is the set of agents and $\mathcal{E} \subseteq V \times V$ is the set of edges. An edge (i, j) belongs to $\mathcal{E}$ if and only if agents i and j communicate with each other, in which case we have also $(j, i) \in \mathcal{E}$. We denote as $\mathcal{N}_i = \{j : (i, j) \in \mathcal{E}\}$ the neighbors of each agent i. The systems have unknown linear dynamics with graph-induced coupling. In the linear case, this can be written as

$$x_{i,t+1} = A_{ii}x_{i,t} + \sum_{j \in \mathcal{N}_i} A_{ij}x_{j,t} + B_i u_{i,t}, \qquad \forall i \in V,$$

where $x_{i,t} \in \mathbb{R}^{n_i}$ is the i-th system's state at time $t \in \mathbb{N}$, $u_{i,t} \in \mathbb{R}^{m_i}$ is the input applied to the i-th system at time t, and $A_{ij} \in \mathbb{R}^{n_i \times n_j}$ and $B_i \in \mathbb{R}^{n_i \times m_i}$ are the (unknown) i-th system's matrices. The goal is to find a distributed linear feedback of the type

$$u_{i,t} = K_{ii}x_{i,t} + \sum_{j \in \mathcal{N}_i} K_{ij}x_{j,t},$$

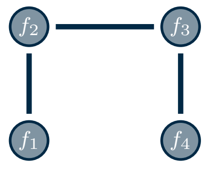
with each $K_{ij} \in \mathbb{R}^{m_i \times n_j}$. Namely, we search for a control law matching the communication graph, see e.g. [243]. This scenario is enclosed by Problem (1.4) as a special case, where the desired control is in the form (1.3) with the sparsity constraint $K \circ \mathcal{S} = 0$ and the (i, j)-th entry of $\mathcal{S}$ is equal to

$$\mathcal{S}_{(i,j)} := \begin{cases} 0, & \text{if } j \in \mathcal{N}_i, \\ 1, & \text{if } j \notin \mathcal{N}_i, \end{cases} \tag{1.5}$$

where $\mathbf{1}$ and $\mathbf{0}$ denote matrices of suitable dimension with all ones and zeros, and If the optimal, non-sparse gain $K^\star$, solution of Problem (1.2) was used, the controller would have the form

$$u_{i,t} = \sum_{j=1}^{N} K_{ij}^\star x_{i,t},$$

which would not satisfy the required graph-induced sparsity and would not be implementable in practice. Among the different engineering use cases, this general setup finds applications in controlling power networks [79] as well as in the formation control of vehicles [150]. This setting is graphically represented in Figure 1.1.



Communication network

$$\begin{matrix} * & * & * & * & 0 & 0 & 0 & 0 \\ * & * & * & * & * & * & 0 & 0 \\ 0 & 0 & * & * & * & * & * & * \\ 0 & 0 & 0 & 0 & * & * & * & * \end{matrix}$$

K

Figure 1.1: Distributed Control of Sparse Dynamics: the gain K is required to match the network structure.

1.2 Optimal Control of Nonlinear Systems

The second scenario we investigate in this thesis deals with the design of optimal trajectories for nonlinear dynamical systems. Specifically, we consider nonlinear, discrete-time systems described by

$$x_{t+1} = f(x_t, u_t), \qquad x_0 = x_{\text{init}} \tag{1.6}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the state and the input of the system at time t, respectively, while $x_{\text{init}} \in \mathbb{R}^n$ represents the fixed initial condition. The time t assumes only discrete values, i.e., $t \in \mathbb{N}$, $t \geq 0$. The map $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the vector field describing the nonlinear dynamics. Given a time-horizon T, we define the T-long sequences of states and inputs as

$$\boldsymbol{x} := \text{col}(x_1, \dots, x_T) \in \mathbb{R}^{nT}, \qquad \boldsymbol{u} := \text{col}(u_0, \dots, u_{T-1}) \in \mathbb{R}^{mT}.$$

A pair $(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ is said to be a *trajectory* of the system described by (1.6) if its components satisfy the constraint represented by the dynamics (1.6) for all $t \in [0, T-1]$ with initial condition x_{init}. To each state-input sequence $(\boldsymbol{x}, \boldsymbol{u})$ it is associated a user defined cost functional $\ell : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \to \mathbb{R}$ defined as

$$\ell(\boldsymbol{x}, \boldsymbol{u}) := \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{1.7}$$

where $\ell_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the stage cost and $\ell_T : \mathbb{R}^n \to \mathbb{R}$ represents the terminal cost.

Overall, our objective is to design a trajectory $(\boldsymbol{x}, \boldsymbol{u})$ of system (1.6), such that the cost functional (1.7) is minimized. More formally, we seek for a solution of the optimal control problem

$$\min_{\boldsymbol{x}, \boldsymbol{u}} \; \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{1.8a}$$

$$\begin{aligned} \text{subj.to } & x_{t+1} = f(x_t, u_t), \quad t \in [0, T-1] \\ & x_0 = x_{\text{init}}. \end{aligned} \tag{1.8b}$$

where $\boldsymbol{x}$ and $\boldsymbol{u}$ are the decision variables.

The resolution of Problem (1.8) becomes particularly challenging in a variety of modern engineering applications. Throughout this dissertation, we will introduce several variants of Problem (1.8), by suitably manipulating the structure of the dynamics (1.8b) and the cost functional (1.8a). Specifically, we will develop numerical algorithms tackling versions of Problem (1.8), tailored to (i) large-scale systems, (ii) distributed multi-agent systems, (iii) partially unknown dynamical systems, (iv) systems affected by stochastic uncertainties.

In the following paragraphs, we will introduce these settings, presenting them via both motivating examples and thorough formalizations.

Remark 1.1. For the sake of exposition, the algorithms presented in this thesis are tailored to *unconstrained* optimal control problems, i.e., neglecting the presence of state-input constraints. However, constraints can be addressed by adopting barrier function approaches (e.g., [114]). Although these approaches may influence the numerical properties and performance of the algorithm, they have demonstrated successful results in various settings (cf. the review paper [3] and references therein). We apply these methodologies in an example of constrained optimal control provided in Sec. 3.2.3. △

Motivating example: Optimal Control of Large-Scale Systems

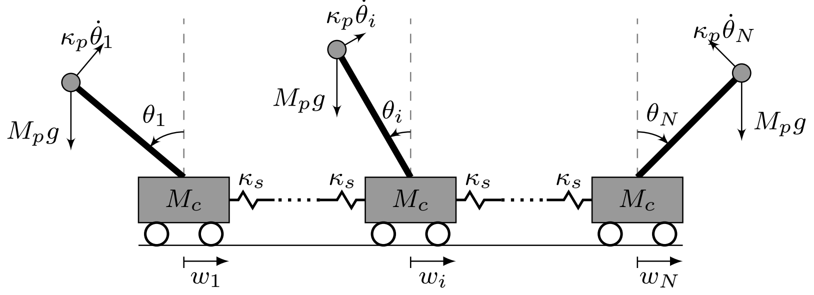


Figure 1.2: Scheme of the train of inverted pendulum-on-cart systems.

As a motivating example of optimal control of a large-scale system, consider the task of generating an optimal trajectory for a large-scale system made by a train of N inverted pendulums on carts as shown in Figure 1.2. For each system $i \in \{1, \ldots, N\}$, the nonlinear dynamics is given by

$$M_p l^2 \ddot\theta_i + \mathrm{f}_p \dot\theta_i - M_p l \sin(\theta_i)\ddot w - M_p l g \sin(\theta_i) = 0$$

$$(M_c + M_p)\ddot{w}_i + \mathrm{f}_c\dot{w}_i - \frac{1}{2}M_p l\cos(\theta)\ddot{\theta} + \frac{1}{2}M_p l\sin(\theta)\dot{\theta}^2 - \kappa_s w_{i+1} + \kappa_s w_{i-1} = u_i,$$

where θ_i is the angle measured from the vertical upward position and w_i is the the lateral position of the cart. Each system is controlled through a force u_i applied to the cart. Since the state space of the overall system scale linearly with the number of the subsystems, this scenario represents a challenging setting for numerical optimal control algorithms due to the large dimension of the decisions variables. Similar dynamical systems find applications in various engineering fields, such as in modeling the continuous surfaces of telescope mirrors which are actuated using a discrete number of actuators [158].

In this thesis, the design of the proposed optimal control algorithms draws inspiration from reinterpreting neural network training — notoriously large-scale models — within an optimal control context. Observing that neural network training primarily relies on first-order gradient-based algorithms, we introduce, in Chapter 3, a first-order algorithmic framework specifically devised for these large-scale systems.

1.2.1 Distributed Optimal Control of Multi-agent Cyber-physical Systems

We now specify Problem (1.8) to a multi-agent setting. We assume there exists a network of N systems, called *agents*, that have communication, computation, and control capabilities. The agents behavior evolves according to the heterogeneous nonlinear dynamics

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \qquad x_{i,0} = x_{i,\text{init}} \tag{1.9}$$

for all $i \in \{1, \ldots, N\}$, where $f_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^{m_i}$, $x_{i,t} \in \mathbb{R}^{n_i}$ is the state of agent i at time t, and $u_{i,t} \in \mathbb{R}^{m_i}$ is the control input applied to agent i at time t. The initial condition, for all $i \in \{1, \ldots, N\}$, is fixed and given by $x_{i,\text{init}} \in \mathbb{R}^{n_i}$. Our goal is to design a control law for the N dynamical systems (1.9) such that a global performance index is minimized. More formally, we aim at solving the optimal control problem

$$\min_{\substack{x_{i,0},\ldots,x_{i,T}\\ u_{i,0},\ldots,u_{i,T-1}\\ i=1,\ldots,N}} \sum_{i=1}^{N}\sum_{t=0}^{T-1} \ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t, u_t)) + \ell_{i,T}(x_{i,T}, \sigma_T(x_T)) \tag{1.10a}$$

$$\text{subj.to} \quad x_{i,t+1} = f_i(x_{i,t}, u_{i,t}) \quad t = 0, \ldots, T-1 \tag{1.10b}$$

$$x_{i,0} = x_{i,\text{init}} \quad i = 1, \ldots, N, \tag{1.10c}$$

where $\ell_{i,t} : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \times \mathbb{R}^q \to \mathbb{R}$ represents the so-called stage cost, while $\ell_{i,T} : \mathbb{R}^{n_i} \times \mathbb{R}^q \to \mathbb{R}$ is the terminal cost. The variables $\sigma_t(\cdot,\cdot)$ and $\sigma_T(\cdot)$ are the so-called *aggregative* variables, which couple the states and inputs of all agents. More formally, the aggregate variables, i.e., $\sigma_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$, for all $t \in [0, T-1]$, and $\sigma_T : \mathbb{R}^n \to \mathbb{R}^q$ couple the states and inputs of all the agents according to

$$\sigma_t(x_t, u_t) = \frac{1}{N}\sum_{i=1}^{N} \varphi_{i,t}(x_{i,t}, u_{i,t}), \qquad \sigma_T(x_T) = \frac{1}{N}\sum_{i=1}^{N} \varphi_{i,T}(x_{i,T}), \tag{1.11a}$$

where, $n = \sum_{i=1}^N n_i$ and $m = \sum_{i=1}^N m_i$, and we define the stacks of the states and inputs of all the agents at time t as

$$\begin{aligned} x_t &:= \text{col}(x_{1,t}, \ldots, x_{N,t}) \in \mathbb{R}^n \\ u_t &:= \text{col}(u_{1,t}, \ldots, u_{N,t}) \in \mathbb{R}^m. \end{aligned}$$

The functions $\varphi_{i,t} : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^q$, for all $t \in [0, T-1]$, and $\varphi_{i,T} : \mathbb{R}^{n_i} \to \mathbb{R}^q$, represent the contribution of agent i to the aggregative variable, for all and $i \in \{1, \ldots, N\}$. Importantly, the local cost function of agent i depends on both the local state and input and the aggregative variables coupling the states and the inputs of all the agents. Notice that the presence of the aggregate variable $\sigma(\cdot)$ allows Problem (1.10) to capture collective behaviors. This proved to be effective in a variety of applications, from cooperative robotics to energy management, see, e.g., [21, 122, 176, 178, 223]

Due to the possible huge number of agents, Problem (1.10) calls for a resolution via a distributed optimization algorithm, namely a numerical strategy leveraging on the computational power of each agent, while exchanging information within the network. Specifically, we assume the N agents to be able to communicate among each others via a network modeled as an undirected graph $\mathcal{G} = (V, \mathcal{E})$, where $V = \{1, \ldots, N\}$ is the set of agents and $\mathcal{E} \subseteq V \times V$ is the set of edges. An edge (i, j) belongs to $\mathcal{E}$ if and only if agents i and j communicate with each other, in which case we have also $(j, i) \in \mathcal{E}$. We denote as $\mathcal{N}_i = \{j : (i, j) \in \mathcal{E}\}$ the neighbors of each agent i.

Notice that, due to the coupling represented by the aggregative variables (1.11), each agent has only a partial knowledge of Problem (1.10) and must cooperate with each other in order to find a solution.

Motivating example: Distributed Surveillance via Heterogeneous Robotics Systems

As an example of application of optimal control of multi-agent systems, we consider a team of N heterogeneous quadrotors employed in a robotic surveillance scenario. Each quadrotor i has a continuous-time dynamics described by

$$\begin{aligned} \ddot{p}_{x,i} &= \tfrac{u_{i,1}+u_{i,2}}{M_i} \sin\theta_i \\ \ddot{p}_{y,i} &= \tfrac{u_{i,1}+u_{i,2}}{M_i} \cos\theta_i - g \\ \ddot{\theta}_i &= \tfrac{L_i}{2J_i}(u_{i,1} - u_{i,2}) \end{aligned}$$

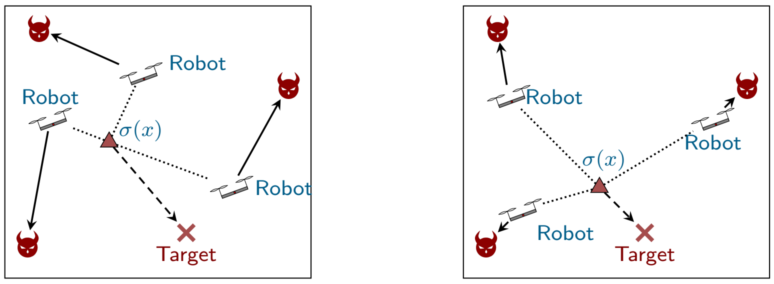


Figure 1.3: Surveillance scenario, each robot patrols an area invaded by malicious intruders, while keeping the formation barycenter over the target.

where $p_{x,i}, p_{y,i}$ is the position of the robot in the $x-y$ plane, θ_i its orientation, $u_{i,1}$ and $u_{i,2}$ are the control inputs, i.e., the thrust applied to the right and left motor. The mechanical parameters M_i, J_i, L_i denote the mass, the inertia and the width of each quadrotor. The goal of each robot is to patrol a pre-defined area trying to stay as close as possible to a given reference trajectory while keeping the formation barycenter over a (moving) target. This scenario can be captured by means of the distributed aggregative optimal control scenario formalized via Problem (1.10). Specifically, for each quadrotor i we can suitably define a local cost function such that (i) a local reference signal $(\boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}})$ is tracked, (ii) the formation barycenter, modeled via $\sigma_t(x_t, u_t)$, is maintained as close as possible to the target located at $b_t \in \mathbb{R}^2$ at time t. More formally, the stage cost, for each i and t, is defined as

$$\ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t)) = \|x_{i,t} - x_{i,t}^{\text{des}}\|_{Q_{i,t}}^2 + \|u_{i,t} - u_{i,t}^{\text{des}}\|_{R_{i,t}}^2 + \|\sigma(x_t) - b_t\|_{Q_{i,t}^{\sigma}}^2$$

for suitably chosen cost matrices $Q_{i,t}$, $R_{i,t}$ and $Q_{i,t}^{\sigma}$. The aggregative variables denoting the weighted center of mass of the team reads as

$$\sigma(x_t) := \sum_{i=1}^{N} H_i x_{i,t}$$

where $H_i = [I_2 \; 0_{2\times4}]$ for all $i = 1, \dots, N$. We define similarly the terminal cost function $\ell_{i,T}(x_{i,T}, \sigma_T(x_T))$. An illustrative concept of this framework is provided in Figure 1.3.

1.2.2 Learning-driven Optimal Control of Uncertain Nonlinear Systems

Finally, we show how Problem (1.8) can be suitably extended to the case where the dynamics (1.8b) is unknown. Throughout the dissertation, we will also address problems where the key challenge that the dynamics $f(\cdot)$ is composed by a nominal, known model (e.g., derived from first principles), and an unknown part as

$$f(x_t, u_t) = f_\star(x_t, u_t) + g_\star(x_t, u_t) \tag{1.12}$$

where $f_\star : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ models the nominal dynamics while $g_\star : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is unknown. We investigate nonlinear optimal control problems in which we look for trajectories of the unknown system (1.12) that minimize a performance criterion defined over a fixed, time horizon $[0, T]$. Formally, we aim to solve the problem

$$\min_{\substack{x_1,\dots,x_T \\ u_0,\dots,u_{T-1}}} \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{1.13a}$$

$$\begin{aligned} \text{subj.to } & x_{t+1} = f_\star(x_t, u_t) + g_\star(x_t, u_t), \quad t \in [0, T-1], \\ & x_0 = x_{\text{init}}, \end{aligned} \tag{1.13b}$$

with stage costs $\ell_t : \mathbb{R}^{mT} \to \mathbb{R}$, for all t, and terminal cost $\ell_T : \mathbb{R}^{mT} \to \mathbb{R}$. The main challenge of the optimal control problem (1.13) is that the dynamics is only partially known and the presence of the unknown term calls for novel learning techniques to be combined into an optimal control scheme.

Motivating example: Inverted Pendulum with Unknown Friction

Consider a canonical testbed for nonlinear control given by the inverted pendulum, shown in Figure 1.4, which is governed by the following nonlinear continuous-time dynamics

$$Ml^2\ddot{\theta} = +Mgl\sin(\theta) - \mathrm{f}_\ell l\dot{\theta} - \mathrm{f}_c l\dot{\theta}^3 + u,$$

where θ is the angle, $\dot{\theta}$ is the velocity and u is the input with the same sign of θ. Moreover, M is the mass, l is the pendulum length, g is the gravity acceleration, $\mathrm{f}_\ell = \mathrm{f}_{\ell,0} + \Delta\mathrm{f}_\ell$ is the linear friction coefficient and f_c is the cubic friction coefficient. We assume to partially know only the linear coefficient, while the cubic term f_c is totally not modeled. It is a realistic framework since not modeled friction-like effects are usually present in more complex applications as, e.g., drag forces in quadrotors or tyre frictions in cars. Importantly, we assume we can recover knowledge of the true dynamics by collecting measurements during experimental sessions. Exemplary applications can be found in the field of autonomous driving, see, e.g., [129] The uncertainties in the model dynamics make unpractical the traditional, model-based approaches, thus calling for novel, data-driven, optimal control algorithms.

1.2.3 Optimal Control of Systems with Stochastic Uncertainties

A challenging instance of Problem (1.8) is represented by the scenario in which the original nonlinear dynamics (1.6) is affected by stochastic disturbances. In this setting, we consider discrete-time nonlinear dynamics subjected to stochastic disturbance in the

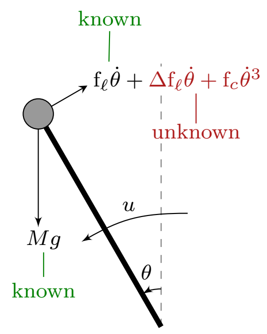

known
$f_\ell\dot{\theta} + \Delta f_\ell\dot{\theta} + f_c\dot{\theta}^3$
unknown
u
Mg
known
θ

Figure 1.4: Graphical representation of an inverted pendulum moving in an environment with partially unknown dynamics.

form

$$x_{t+1} = f(x_t, u_t, w_t) \tag{1.14}$$

in which $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$ and $x_t \in \mathbb{R}^n$ represent the state of the system at time t, $u_t \in \mathbb{R}^m$ the control input and $w_t \in \mathbb{R}^p$ is stochastic disturbance, realization of a properly defined random variable w_t. Considering a time-horizon T, we define the sequence of T random variables w_t as $\mathbf{w} := \mathrm{col}(\mathrm{w}_0, \ldots, \mathrm{w}_{T-1})$, which is a random variable itself. We then denote as $\boldsymbol{w} \in \mathbb{R}^{pT}$ a realization of $\mathbf{w}$.

Assuming a deterministic sequence of inputs $\boldsymbol{u} \in \mathbb{R}^{mT}$, for a given initial condition $x_0 = x_{\text{init}}$ and a specified time horizon T, one can see the resulting state trajectory $\boldsymbol{x}$ as a function of the random variable $\mathbf{w}$. This comes from the composition of each random variable w_t, for $t = 0, \ldots, T-1$, through the nonlinear dynamics (1.14). Hence, we resort to the minimization of the performance index (1.7) taken in expectation with respect to the random variable $\mathbf{w}$. Overall, we aim at solving the following stochastic optimal control problem

$$\min_{\boldsymbol{x},\boldsymbol{u}} \; \mathbb{E}_{\mathbf{w}}\left[\sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T)\right] \tag{1.15a}$$

$$\text{subj.to} \;\; x_{t+1} = f(x_t, u_t, w_t), \qquad x_0 = x_{\text{init}} \tag{1.15b}$$

$$w_t \sim p_{\mathrm{w}_t} \tag{1.15c}$$

where, the operator $\mathbb{E}_{\mathbf{w}}[\cdot]$ denotes the expected value taken with respect to $\mathbf{w}$ and p_{w_t} denotes the probability density function associated to w_t.

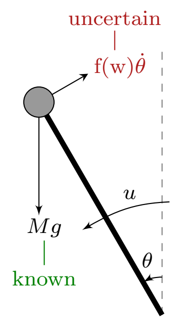


Figure 1.5: Graphical representation of an inverted pendulum moving in an environment with parametric uncertainties in dynamics. The friction f(w) is function of the random variable w.

Motivating example: Inverted Pendulum with Parametric Uncertainties

Consider again the inverted pendulum system, shown in Figure 1.5, which is governed by the following nonlinear continuous-time dynamics

$$Ml^2\ddot{\theta} = +Mgl\sin(\theta) - \mathrm{f}(w_t)l\dot{\theta} + u,$$

where θ is the angle, $\dot{\theta}$ is the velocity and u is the input with the same sign of θ. Moreover, M is the mass, l is the pendulum length, g is the gravity acceleration, $\mathrm{f}(w_t)$ is the linear friction coefficient. We assume the friction coefficient to be uncertain and normally distributed with (known) mean value f_0 and variance σ_f.

1.3 Safety-critical Optimal Control of Nonlinear Systems

Finally, we consider the task of generating *safe* trajectories for nonlinear systems accordingly to some optimality criteria. In this scenario, we consider the novel framework of layered control architectures.

A multi-layer control architecture consists of (at least) two, interconnected, dynamical systems with different roles and objectives. In this work, we consider an architecture where a trajectory generation layer is coupled with a low-level tracking controller. More formally, we frame this control scheme considering, (i) a *low-level* model, with dynamics

$$\dot{x}_\ell(t) = f_\ell(x_\ell(t), u_\ell(t)) \tag{1.16}$$

where $f_\ell : \mathbb{R}^n \to \mathbb{R}^n$ and state $x_\ell \in \mathbb{R}^{n_l}$, and input $u_\ell \in \mathbb{R}^{m_l}$. This dynamics usually captures the "true" system dynamics. And, (ii) an *high-level* reference model

$$\dot{x}_\mathrm{h}(t) = f_\mathrm{h}(x_\mathrm{h}(t), u_\mathrm{h}(t)) \tag{1.17}$$

where $f_\mathrm{h} : \mathbb{R}^{n_\mathrm{h}} \to \mathbb{R}^{n_\mathrm{h}}$ and $x_\mathrm{h} \in \mathbb{R}^{n_\mathrm{h}}$, high-level state, $u_\mathrm{h} \in \mathbb{R}^{m_\mathrm{h}}$, high-level control signal. In general the objective is to design an high-level control law, applied to the low-level dynamics via a projection map $\pi_{u_\mathrm{h}}(u_\mathrm{h}) = u_\ell$, while leveraging the information from the low-level dynamics, whose state is related to the high-level one via $\pi_{x_\ell}(x_\ell) = x_\mathrm{h}$. Additionally, in the considered control scenario, the state of the high-level model $x_\mathrm{h}(t)$ is constrained to satisfy some state constraints in the form

$$h(x_\mathrm{h}(t)) \geq 0, \qquad t \geq 0 \tag{1.18}$$

where $h : \mathbb{R}^{n_\mathrm{h}} \to \mathbb{R}$ is a continuously differentiable function. These constraints represent some safety conditions for dynamics (1.17), i.e., they encode a set $\mathbb{S}$ of *safe* states defined as the superlevel set of the function $h(x_\mathrm{h})$, i.e.,

$$\mathbb{S} := \{x_\mathrm{h}(t) \in \mathbb{R}^{n_\mathrm{h}} \mid h(x_\mathrm{h}(t)) \geq 0\} \subseteq \mathbb{R}^{n_\mathrm{h}}. \tag{1.19}$$

The concept of safety, associated to system (1.17), is framed in the context of enforcing invariance of the set of states $\mathbb{S}$. More formally, system (1.17) is said to be safe with respect to $\mathbb{S}$ if $\mathbb{S}$ is forward invariant, i.e., if, for every $x_\mathrm{h}(0) \in \mathbb{S}$, $x_\mathrm{h}(t) \in \mathbb{S}$ for all $t \geq 0$.

Nevertheless, our goal is to generate trajectories for a multi-layer system that are not only safe, but also optimize a user defined cost functional. More formally, our goal is to generate trajectories for the high-level system (1.17), solution of the following optimal control problem

$$\min_{\boldsymbol{x},\boldsymbol{u}} \int_0^\infty q(x_\mathrm{h}(\tau)) + u_\mathrm{h}(\tau)^\top u_\mathrm{h}(\tau)\,\mathrm{d}\tau \tag{1.20a}$$

$$\text{subj.to } \dot{x}_\mathrm{h} = f_\mathrm{h}(x_\mathrm{h}(t), u_\mathrm{h}(t)), \quad x_\mathrm{h}(0) = x_\mathrm{init} \tag{1.20b}$$

$$h(x_\mathrm{h}(t)) \geq 0 \tag{1.20c}$$

where $q : \mathbb{R}^{n_\mathrm{h}} \to \mathbb{R}$ positive-definite function, i.e., $q(x) > 0$ for all $x \neq 0$, and continuously differentiable, i.e., of class $\mathcal{C}^1$, and we denote as $\boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^{n_\mathrm{h}}$ the state trajectory solution of (1.17) with initial condition x_init, under a given control signal $u_\mathrm{h}(t)$ and time interval $[t_0, t_f]$, $t_0, t_f \in \mathbb{R}_{\geq 0}$, $t_f > t_0$. Similarly, $\boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^{m_\mathrm{h}}$ represents the corresponding input signal. The high-level trajectory solution of (1.20) represents then a reference trajectory for the low-level system (1.16). Then, the idea is to update the initial condition of (1.20) at discrete time instants, according to the state of the low-level dynamics(1.16).

Motivating example: Safe Exploration of Unknown Enviroments

As a motivating example, consider the scenario where an autonomous robot is tasked with exploring an unknown environment. In this setting, the robot must navigate while avoiding obstacles, relying on its onboard sensors for guidance. This illustrative

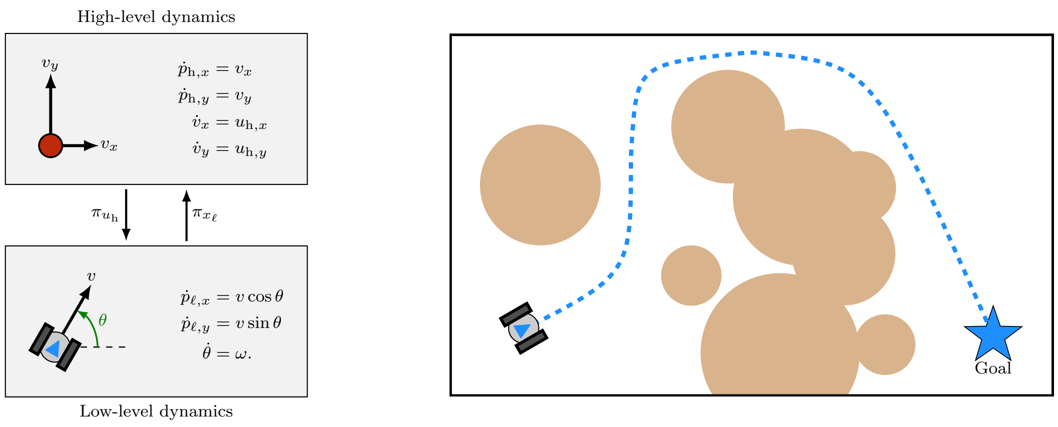

High-level dynamics
v_y
v_x
$\dot{p}_{\mathrm{h},x} = v_x$
$\dot{p}_{\mathrm{h},y} = v_y$
$\dot{v}_x = u_{\mathrm{h},x}$
$\dot{v}_y = u_{\mathrm{h},y}$
$\pi_{u_{\mathrm{h}}}$
π_{x_ℓ}
v
θ
$\dot{p}_{\ell,x} = v\cos\theta$
$\dot{p}_{\ell,y} = v\sin\theta$
$\dot{\theta} = \omega.$
Low-level dynamics
Goal

Figure 1.6: (Left) Representation of a two-layer control architecture: with low-level unicycle dynamics, and high-level double-integrator dynamics. The interface between the two layers is provided by the maps $\pi_{x_\ell}(x_\ell)$ and $\pi^{u_{\mathrm{h}}}(u_{\mathrm{h}}, x_{\mathrm{h}})$. (Right) Graphical representation of a differential-drive robot traversing an area with N obstacles modeled via N safety constraint $h_i(x_{\mathrm{h}}) = \|x_{\mathrm{h}} - x_{c,i}\|^2 - r_i$.

scenario is depicted in Figure 1.6. A differential-drive robot admits as low-level model representation the unicycle dynamics, i.e.,

$$\dot{p}_{\ell,x} = v\cos\theta, \tag{1.21}$$

$$\dot{p}_{\ell,y} = v\sin\theta, \tag{1.22}$$

$$\dot{\theta} = \omega, \tag{1.23}$$

with $x_\ell = (p_{\ell,x}, p_{\ell,y}, \theta) \in \mathbb{R}^3$, and $u_\ell = (v, \omega) \in \mathbb{R}^2$. Being (1.21) differentially flat, a valid high-level dynamics is

$$\dot{x}_{\mathrm{h}} = \begin{bmatrix} 0_2 & I_2 \\ 0_2 & 0_2 \end{bmatrix} x_{\mathrm{h}} + \begin{bmatrix} 0_2 \\ I_2 \end{bmatrix} u_{\mathrm{h}} \tag{1.24}$$

with $x_{\mathrm{h}} \in \mathbb{R}^4$, $u_{\mathrm{h}} \in \mathbb{R}^2$. The low level dynamics (1.21) is coupled with (1.24) via

$$\pi_{x_\ell}(x_\ell) = \begin{bmatrix} p_{\ell,x} & p_{\ell,x} & \dot{p}_{\ell,x} & \dot{p}_{\ell,x} \end{bmatrix}^\top \tag{1.25a}$$

$$\pi^{u_{\mathrm{h}}}(u_{\mathrm{h}}, x_{\mathrm{h}}) = \begin{bmatrix} \sqrt{x_{\mathrm{h},3}^2 + x_{\mathrm{h},4}^2}, & \frac{-x_{\mathrm{h},4}u_{\mathrm{h},1} + x_{\mathrm{h},3}u_{\mathrm{h},2}}{x_{\mathrm{h},3}^2 + x_{\mathrm{h},4}^2} \end{bmatrix}^\top \tag{1.25b}$$

Our goal then is to design a control law such that the overall system (i) reaches a desired goal pose, encoded in the cost function $q(x_{\mathrm{h}})$, (ii) avoids the obstacles along the way. The obstacles can be modeled as N circumferences centered at $x_{c,i}$, thus resulting in a safe set $\mathbb{S}$ defined as the intersection of N safety constraints in the form $h_i(x_{\mathrm{h}}) \geq 0$, $i = 1, \ldots N$. This scenario can be framed in the resolution of a specific instance of Problem (1.20).

Chapter 2

Reinforcement Learning and Data-driven Optimal Control for Large-Scale and Multi-Agent Systems

In this chapter, we explore novel approaches for the resolution of the Linear Quadratic Regulator (LQR) problem tailored to large-scale unknown dynamics.

We propose a set of algorithmic strategies based on *data-driven* approaches, which are then instrumental for the development of numerical solutions for the design of *structurally constrained* feedback control policies.

In Section 2.2, we propose an on-policy strategy for the resolution of the LQR problem. This approach involves identifying system matrices while simultaneously optimizing the control policy applied to the real system. This results in a time-varying control policy that is iteratively updated based on the latest system matrix estimates, aiming towards the optimal LQR solution. In Sections 2.3 and 2.4, we tackle the problem of designing of data-driven control strategies for large-scale and distributed control systems with structural constraints. In Section 2.3, we introduce a Reinforcement Learning framework based on a Q-learning scheme. This framework maintains the desired policy structure through an LMI-based policy improvement step. Meanwhile, in Section 2.4 we reformulate the gain design process as a direct policy search. We propose a data-driven approach for computing the optimal structured gain based on the combination of a gradient method and an augmented Lagrangian approach. The results of this chapter are based on [205–207, 210].

2.1 Literature Review

The Linear Quadratic Regulator problem is a cornerstone in the history of Control Theory, with a vast and extensive body of literature surrounding it. Next we review various model-based and data-driven approaches for resolving the LQR problem, including cases where structural constraints on the feedback gain are applied.

The model-based LQR solution has a long history in control and can be efficiently addressed resorting to dynamic programming and Riccati equations, see, e.g., [9, 29]. The extension of LQR problems to a data-driven scenario, however, relies manly on solution approaches where the optimal control problem is solved directly in the policy space, i.e., in terms of the feedback gain, see e.g., the Anderson-Moore algorithm or Kleinman policy iteration presented in [137, 143, 186]. Indeed, these approaches are early versions of policy iteration and policy gradient methods developed in reinforcement learning [29, 221]. In data-driven control and reinforcement learning, a fundamental distinction between online and offline methods exists. While online methods involve collecting system data while concurrently refining the controller, offline approaches leverage on a shwred data collection phase before developing the controller. Former derivations of Reinforcement Learning methods for LQ regulation trace back to [44]. Offline approaches can be further distinguished between direct, where data are used directly in the policy design phase, and indirect approaches, where a preliminary identification step is performed. Direct strategies often tackle the LQR problem by exploiting Persistently Exciting (PE) data together with semi-definite programming and Linear Matrix Inequalities (LMI) approaches, as introduced in [70]. These methodologies are thoroughly studied also in [70, 197, 233]. The work in [198] extends these concepts to unknown linear systems with switching time-varying dynamics. These LMI-based solutions also allowed for the design of control policies in case of noisy data, as explored in [71, 81]. The recent survey [73] also includes an extension to nonlinear systems. Instead, the work [75] proposes a safe-learning strategy for LQR via an indirect approach, i.e., the unknown dynamics is firstly estimated, so that the control gain is optimized on the estimated quantities. As for online methods, a further classification distinguishes between off-policy and on-policy algorithms. Off-policy algorithms pursue a value iteration approach, learning the optimal value function from data that may be independent of the current policy. Conversely, on-policy algorithms employ a policy iteration framework, evaluating the value function of the current policy using data stemming from the same policy. In the context of off-policy methodologies, we find iterative methods inspired by the Kleinman algorithm, involving either parameter identification or direct estimate of the policy [142, 155, 161, 172, 173, 184]. An off-policy adaptive value-iteration strategy is analyzed in [183]. The paper [155] investigates an off-policy Q-learning strategy, with an additional focus on the the computational complexity. The recent works [33, 182, 253] proposes iterative algorithms that at do not assume the existence of a stabilizing initial policies. Conversely, on-policy control techniques are proposed in the continuous-time framework in [124, 136, 160, 191, 192, 236]. In [183] stability guarantees on the learning dynamics are provided without the assumption of an initial stabilizing policy. In [39], an on-policy algorithm is proposed leveraging on model-reference adaptive control tools. Notice that, although these strategies are developed in a continuous-time framework, the policy is updated at discrete-time instants only, as soon as enough informative data have been collected. In [231] a gradient-based Q-learning approach is proposed. In [173] robustness of the policy iteration for continuous-time systems under additive, bounded disturbances is studied. The discrete-time framework has received significant attention in the learning community. In recent years [188], it is investigated the connection between the early works in the adaptive control field, e.g. [52] and on-policy reinforcement learning. A model-free approache for discrete-time LQR based based on reinforcement learning are studied and developed in [135]. Beyond these distinctions, in the recent years, the LQR problem has been addressed also via policy-gradient methods. The convergence properties of the (policy) gradient methods are thoroughly studied in [91] for discrete-time LQR. A model-free, gradient-based, strategy is proposed in [250]. While in [162], the sample complexity and convergence properties for the continuous-time case are examined. Recent works also explored the non-asymptotic performances of model-free LQR algorithms. Sub-linear regret result is given in [1]. Poly-logarithmic regret bounds are given in [4, 58]. Finally, the sample complexity for model-free LQR is studied in [74].

The rising interest towards the development of distributed control policies motivated the research for algorithmic strategies for the design feedback laws with a given structural profiles. Such structured control policies find application in control of power networks [79] as well as in formation control of vehicles [150]. A variety of methodologies have been proposed in the model-based framework, i.e., the system dynamics is supposed to be known. In a model-based scenario, LQR design methods to handle a sparsity constraints are proposed in [149, 151] in the continuous-time case. An LMI-based approach for the continuous-time case is proposed in [179]. In [249], constraints over the policy structure are introduced to handle statistical modeling, as well as sensor and actuator selection. The discrete-time setting is considered in [87], where sequential convex programming and LMIs are employed. The problem of designing sparse feedback matrices solving a minimum-gain eigenvalue pole placement problem is considered in [131]. Sparse LQR has been also used to design linear controllers for spatially distributed systems in [217] and [216]. Other approaches for distributed control in network systems have been developed in [40, 125, 235, 251] and [243]. All the mentioned approaches require exact knowledge of the model. However, in many cases an accurate model of the dynamical system may be impractical to obtain, which impacts on the reliability of the developed controller. Data-driven solutions represent an interesting way to compute a (optimal) control strategy when the dynamics is completely unknown. See, e.g., [71] and [23]. As a matter of fact, Reinforcement Learning methods are recently getting more and more attention from the control community. Distributed scenarios with Reinforcement Learning for LQR, are studied both in discrete [146] and in continuous-time context [126, 127]. In [146], the policy optimization methodology have been extended to the zero-th order optimization framework, i.e., without using gradient related information, see, e.g.. Another zeroth-order optimization algorithm for a distributed instance of the model-free LQR problem is proposed in [98]. Alongside policy gradient approaches, a different line of research has been devoted to the exploitation of data directly in the in the controller synthesis. In the continuous-time framework, LQR design methods to handle a sparsity constraint on the static feedback are proposed in [149, 151]. In [87] the discrete-time case is considered resorting to sequential convex programming and LMIs. In [112], a data-driven solution to the sparse H_2 feedback gain design is proposed for continuous-time systems. In [98], model-free methods for output-feedback finite-horizon LQR problems with sparsity feedback constraints are studied for distributed systems resorting to zero-th order optimization approaches. In [59], the problem of sparse feedback design is faced in the context of pole placement and eigenstructure assignment. The design of policies with a decentralized structure is considered also in [156] and [108]. Similar scenarios are also considered in the context of distributed reinforcement learning, see, e.g., [146].

2.2 On-policy Data-driven Linear Quadratic Regulator

We consider discrete-time, linear and time-invariant systems described by the dynamics

$$x_{t+1} = A_\star x_t + B_\star u_t, \qquad x_0 \sim p_{\mathrm{x}_0}, \tag{2.1}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ denote, respectively, the state and the input of the system at time $t \in \mathbb{N}$, while $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ represent, respectively, the state and the input matrix. As for the initial condition $x_0 \in \mathbb{R}^n$, we assumed that it is sampled from a random variable x_0, with (know) probability distribution p_{x_0}. We enforce the following properties on $(A_\star, B_\star)$.

Assumption 2.1. *The pair $(A_\star, B_\star)$ is controllable and unknown.* △

As it will be useful later, we collect the pair (A, B) in a single variable $\theta_\star \in \mathbb{R}^{(n+m)\times n}$ defined as

$$\theta_\star := \begin{bmatrix} A_\star^\top \\ B_\star^\top \end{bmatrix}. \tag{2.2}$$

We would like to design a controller for system (2.1) under Assumption 2.1 resorting to an LQR approach. Specifically, we consider an infinite horizon LQR problem

$$\min_{\substack{x_1, x_2, \ldots, \\ \mathrm{u}_0, \mathrm{u}_1, \ldots}} \ \frac{1}{2} \sum_{t=0}^{\infty} \left(\mathrm{x}_t{}^\top Q \mathrm{x}_t + \mathrm{u}_t{}^\top R \mathrm{u}_t \right) \tag{2.3a}$$

$$\text{subj.to } x_{t+1} = A_\star x_t + B_\star u_t, \qquad x_0 \sim \mathrm{x}_0, \tag{2.3b}$$

where the cost matrices $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are both symmetric and positive definite, i.e., $Q = Q^\top > 0$ and $R = R^\top > 0$. It is well-known that, when $(A_\star, B_\star)$ are known the optimal solution is given by a linear time-invariant policy $u_k = K_\star x_k$ with $K_\star \in \mathbb{R}^{m \times n}$ given by

$$K_\star = -(R + B_\star^\top P_\star B_\star)^{-1} B_\star^\top P_\star A_\star,$$

where $P_\star \in \mathbb{R}^{n \times n}$ solves the Discrete-time Algebraic Riccati Equation (DARE) associated to Problem (2.3), see [9].

Our goal is to devise a data-driven feedback policy for (2.1) while designing a control strategy that *concurrently*

(i) learns the unknown dynamics (2.1);

(ii) improves the policy towards the solution of Problem (2.3);

(iii) actuates the (real) system with the currently available state-feedback policy.

In the model-based case, cf. Appendix B.1, Problem (2.3) admits a reduced problem formulation that explicitly imposes the linear feedback structure to the optimal input and is amenable for gradient-based algorithmic solutions. Specifically, letting the feedback gain be $K \in \mathbb{R}^{m \times n}$ Problem (2.3) can be rewritten as

$$\min_{K \in \mathcal{D}} \ J(K, \theta_\star), \tag{2.4}$$

where $\theta_\star$ is defined in (2.2), the cost function $J : \mathcal{D} \times \mathbb{R}^{(n+m) \times n} \to \mathbb{R}$ is given by

$$J(K, \theta_\star) := \frac{1}{2} \operatorname{Tr} \sum_{t=0}^{\infty} (A_\star + B_\star K)^{t,\top} (Q + K^\top R K)(A_\star + B_\star K)^t, \tag{2.5}$$

and the set $\mathcal{D} \subset \mathbb{R}^{m \times n} := \{K \in \mathbb{R}^{m \times n} \mid J(K, \theta_\star) < \infty\}$ is the domain of J, i.e., the set over which J is well-defined. Notice that, the interior of $\mathcal{D}$ coincides with the set of stabilizing gains $\mathcal{K} := \{K \in \mathbb{R}^{m \times n} \mid A_\star + B_\star K \text{ is Schur}\} \subseteq \mathbb{R}^{m \times n}$, see [48, Lemma 3.2]. Hence, being the set of stabilizing gains open [49, Lemma IV.3] and connected [49, Lemma IV.6], and, if the pair $(A_\star, B_\star)$ were known, a gradient descent method could be used to solve Problem (2.4), where, at each iteration $k \in \mathbb{N}$, the estimate K_k of $K_\star$ is iteratively updated according to

$$K_{k+1} = K_k - \gamma \Gamma(K_k, \theta_\star), \tag{2.6}$$

where $\gamma > 0$ is the stepsize, while $\Gamma : \mathbb{R}^{m\times n} \times \mathbb{R}^{(n+m)\times n} \to \mathbb{R}^{m\times n}$ is the gradient of J with respect to K evaluated at $(K_k, \theta_\star)$, when $\mathbb{R}^{m\times n}$ is equipped with the Frobenius inner product. The gradient $\Gamma(K_k, \theta_\star)$, see also Lemma B.1 in Appendix B.1, can be computed as

$$\Gamma(K_k, \theta_\star) = \Big(RK_k + B_\star^\top P_k(A_\star + B_\star K_k)\Big) W_k. \tag{2.7a}$$

where $W_k \in \mathbb{R}^{n\times n}$ and $P_k \in \mathbb{R}^{n\times n}$ are solutions to the Lyapunov equations

$$\begin{aligned}&(A_\star + B_\star K_k)W_k(A_\star + B_\star K_k)^\top - W_k = -I_n\\ &(A_\star + B_\star K_k)^\top P_k(A_\star + B_\star K_k) - P_k = -(Q + {K_k}^\top R K_k).\end{aligned} \tag{2.7b}$$

Notice that under Assumption 2.1, Problem (2.4) is to be solved without resorting to the knowledge of $\theta_\star$, i.e., when $(A_\star, B_\star)$ are unknown. Therefore, in our framework, it is not possible to implement update (2.6).

2.2.1 On-Policy LQR via Recursive Least Squares Algorithm

In this section, we present the concurrent learning and optimization algorithm developed to solve Problem (2.3) under Assumption 2.1. The proposed on-policy strategy feeds the real system dynamics at each iteration k with the current feedback input including also an additive exogenous dithering signal w_k. Then, a new sample data from the system is collected and used to progressively improve the estimates (A_k, B_k) of the unknown (A, B) via a learning process inspired by Recursive Least Squares (RLS). In turn, (A_k, B_k) is used to refine the feedback gain K_k for (2.3), and the procedure is repeated. The overall scheme is shown in Figure 2.1.

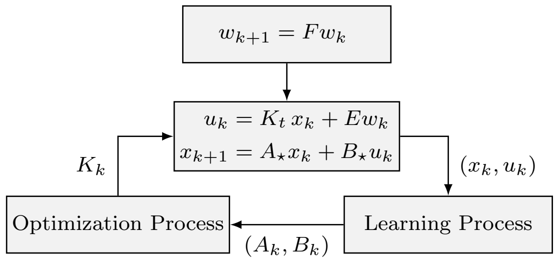


Figure 2.1: Representation of the concurrent learning and optimization scheme.

The overall strategy is reported in Algorithm 1 where, for notational convenience, we denote as $\theta_k \in \mathbb{R}^{(n+m)\times n}$ the estimate of $\theta_\star$ at iteration $k \in \mathbb{N}$. Consistently, $A_k \in \mathbb{R}^{n\times n}$ and $B_k \in \mathbb{R}^{n\times m}$ are the corresponding estimates of $A_\star$ and $B_\star$. Moreover, $H_k \in \mathbb{R}^{(n+m)\times(n+m)}$ and $S_k \in \mathbb{R}^{(n+m)\times m}$ denote two additional states of the learning process, $\lambda \in (0,1)$ is a forgetting factor, while γ is the stepsize as in (2.6). Finally, in order to prescribe the initialization K_0, we introduce the set $\mathcal{B}_{r^\star}(K_\star) \subset \mathbb{R}^{m\times n}$ constructively defined as follows. Being the matrix $A_\star + B_\star K_\star$ Schur, then, there must exist by continuity $r > 0$ such that $A_\star + B_\star K$ is Schur for all $K \in \mathcal{B}_r(K_\star)$. Therefore, let $r^\star > 0$ be the largest value allowed for r ensuring that $A_\star + B_\star K$ is Schur for all $K \in \mathcal{B}_{r^\star}(K_\star)$.

Algorithm 1 On-policy LQR for Unknown Systems

```
Initialization: $x_0 \in \mathbb{R}^n$, $H_0 \in \mathbb{R}^{(n+m)\times(n+m)}$, $S_0 \in \mathbb{R}^{(n+m)\times n}$, $\theta_0 \in \mathbb{R}^{(n+n)\times n}$, $K_0 \in \mathbb{R}^{m\times n}$ and $w_0 \in \mathbb{R}^{n_w}$.
for $k = 0, 1, 2 \dots$ do
    Data collection: generate
        $w_{k+1} = F w_k$
        $\mathrm{d}_k = E w_k$
    and apply
        $u_k = K_k x_k + \mathrm{d}_k$
        $x_{k+1} = A_\star x_k + B_\star u_k$
        $y_k = {x_{k+1}}^\top$
    Learning process: compute
        $H_{k+1} = \lambda H_k + \begin{bmatrix} x_k \\ u_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top$    (2.8a)
        $S_{k+1} = \lambda S_k + \begin{bmatrix} x_k \\ u_k \end{bmatrix} y_k$    (2.8b)
        $\theta_{k+1} = \theta_k - \gamma H_k^\dagger \left(H_k \theta_k - S_k\right).$    (2.8c)
    Optimization process: update
        $K_{k+1} = K_k - \gamma \Gamma(K_k, \theta_k).$    (2.9)
```

Next, we detail the main steps of the proposed algorithm.

Data collection Data from the controlled System (2.1) are recast in an identification-oriented form described by

$$\underbrace{{x_{k+1}}^\top}_{y_k} = \underbrace{\begin{bmatrix} {x_k}^\top & {u_k}^\top \end{bmatrix}}_{\mathrm{g}(x_k,u_k)^\top} \underbrace{\begin{bmatrix} A_\star^\top \\ B_\star^\top \end{bmatrix}}_{\theta_\star}. \tag{2.10}$$

Learning process The adopted learning strategy to compute an estimate of $\theta_\star$ relies on the interpretation of the least squares problem as an online optimization. Specifically, with the measurements (2.10) at hand, we consider, at each $k \in \mathbb{N}$, the online optimization problem

$$\min_{\theta \in \mathbb{R}^{(n+m)\times n}} \frac{1}{2} \sum_{i=0}^{k} \lambda^{k-i} \left\| \mathrm{g}(x_i, \mathrm{u}_i)^\top \theta - y_i \right\|^2, \tag{2.11}$$

where we recall that λ is a forgetting factor. We aim to solve (2.11) through an iterative algorithm that progressively refines a solution estimate $\theta_k \in \mathbb{R}^{(n+m)\times n}$. In particular, we update such an estimate θ_k according to a "scaled" gradient method with Newton's like scaling matrix. If applied to problem (2.11), the iteration would read

$$\theta_{k+1} = \theta_k - \gamma \left(\sum_{i=0}^{k} \lambda^{k-i} \mathcal{H}(x_i, \mathrm{u}_i) \right)^\dagger \left(\sum_{i=0}^{k} \lambda^{k-i} \left(\mathcal{H}(x_i, \mathrm{u}_i)\theta_k - \mathcal{S}(x_k, \mathrm{u}_i, y_i) \right) \right),$$

where $\mathcal{H} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{(n+m)\times(n+m)}$ and $\mathcal{S} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{(n+m)\times n} \times \mathbb{R}^n$ reads as

$$\mathcal{H}(x_i, \mathrm{u}_i) := \mathrm{g}(x_i, \mathrm{u}_i)\mathrm{g}(x_i, \mathrm{u}_i)^\top, \qquad \mathcal{S}(x_i, \mathrm{u}_i, y_i) := \mathrm{g}(x_i, \mathrm{u}_i) y_i.$$

To overcome the issue of storing the whole history of $\mathcal{H}(\cdot,\cdot)$ and $\mathcal{S}(\cdot,\cdot)$, we iteratively keep track of them through the matrix states $H_k \in \mathbb{R}^{(n+m)\times(n+m)}$ and $S_k \in \mathbb{R}^{(n+m)\times n}$ giving rise to (2.8).

Optimization process The estimate θ_k is concurrently exploited in the update of the feedback gain K_k, replacing the unavailable $\theta_\star$ into (2.6) giving rise to (2.9). To ensure sufficiently informative data, we equip our feedback policy with an additive dithering signal $\mathrm{d}_k \in \mathbb{R}^m$. Namely, we implement

$$u_k = K_k x_k + \mathrm{d}_k, \tag{2.12}$$

where d_k is the output of an exogenous system evolving according to a marginally stable linear discrete-time oscillator dynamics (see, e.g., [228]) described by

$$w_{k+1} = F w_k \tag{2.13a}$$

$$\mathrm{d}_k = E w_k, \tag{2.13b}$$

where $w_k \in \mathbb{R}^{n_w}$, with $n_w \geq n + m$, is the state of the exogenous system having $F \in \mathbb{R}^{n_w \times n_w}$ and $E \in \mathbb{R}^{m \times n_w}$ as state and output matrix, respectively. The matrix F is a degree of freedom to properly shape the oscillation frequency of w_k. The following assumption formalizes the requirements for the design of the exogenous System (2.13).

Assumption 2.2. *There exist $\alpha_1, \alpha_2, k_w, \mathbf{t}_\mathrm{d} > 0$ such that, if $w_0 \neq 0_{n_w}$, then the two following conditions hold true*

$$\alpha_1 I_{n_w} \leq \sum_{i=\bar{k}+1}^{\bar{k}+k_w} w_i w_i^\top \leq \alpha_2 I_{n_w}, \quad \text{for all } \bar{k} \in \mathbb{N} \tag{2.14a}$$

$$\operatorname{rank}\left(\begin{bmatrix} \mathrm{d}_0 & \mathrm{d}_1 & \dots & \mathrm{d}_{\mathbf{t}_\mathrm{d}-n-1} \\ \mathrm{d}_1 & \mathrm{d}_2 & & \mathrm{d}_{\mathbf{t}_\mathrm{d}-n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{d}_n & \mathrm{d}_{n+1} & \dots & \mathrm{d}_{\mathbf{t}_\mathrm{d}-1} \end{bmatrix}\right) = m(n+1). \tag{2.14b}$$

Moreover, the eigenvalues of F lie on the unit disk. △

The property (2.14a) is usually referred to as *persistency of excitation* of the signal w_k, see, e.g., [14]. As for the property in (2.14b), it corresponds to the *sufficient richness of order* $n+1$ in the early reference [14], while, more recently, such property has been referred to as *persistency of excitation of order* $n+1$ [70, 238].

The overall dynamics presented in Algorithm 1 can be written as

$$w_{k+1} = F w_k \tag{2.15a}$$

$$x_{k+1} = (A_\star + B_\star K_k) x_k + B_\star E w_k \tag{2.15b}$$

$$H_{k+1} = \lambda H_k + \begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix} \begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix}^\top \tag{2.15c}$$

$$S_{k+1} = \lambda S_k + \begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix} \begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix}^\top \theta_\star \tag{2.15d}$$

$$\theta_{k+1} = \theta_k - \gamma H_k^\dagger (H_k \theta_k - S_k) \tag{2.15e}$$

$$K_{k+1} = K_k - \gamma \Gamma(K_k, \theta_k), \tag{2.15f}$$

in which we have used the explicit expressions for y_k (cf. (2.10)) and u_k (cf. (2.12)). Next, we provide the main result, i.e., the convergence properties of System (2.15).

Theorem 2.1. *Consider System* (2.15) *and let Assumptions 2.1 and 2.2 hold. Then, for each* $(w_0, x_0, H_0, S_0, \theta_0, K_0) \in \mathbb{R}^n \times \mathbb{R}^{(n+m)\times(n+m)} \times \mathbb{R}^{(n+m)\times n} \times \mathbb{R}^{(n+m)\times n} \times \mathcal{B}_{r^\star}(K_\star)$ *such that* $w_0 \neq 0$, $A_0 + B_0K_0$ *is Schur, there exist* $\Pi_x \in \mathbb{R}^{n\times n_w}$, $\Pi_H \in \mathbb{R}^{(n+m)^2\times n_w^2}$, $\Pi_S \in \mathbb{R}^{(n+m)m\times n_w^2}$, $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, \bar{\gamma} > 0$ *such that, it holds*

$$\|x_k - \Pi_x w_k\| \leq a_1 \|x_0 - \Pi_x w_0\| \exp(-a_2 k) \tag{2.16a}$$

$$\begin{aligned}&\left\|H_k - \texttt{unvec}\left(\Pi_H \texttt{vec}\left(w_k w_k^\top\right)\right)\right\| \\ &\quad \leq a_3 \left\|H_0 - \texttt{unvec}\left(\Pi_H \texttt{vec}\left(w_0 w_0^\top\right)\right)\right\| \exp(-a_4 k)\end{aligned} \tag{2.16b}$$

$$\begin{aligned}&\left\|S_k - \texttt{unvec}\left(\Pi_S \texttt{vec}\left(w_k w_k^\top\right)\right)\right\| \\ &\quad \leq a_5 \left\|S_0 - \texttt{unvec}\left(\Pi_S \texttt{vec}\left(w_0 w_0^\top\right)\right)\right\| \exp(-a_6 k)\end{aligned} \tag{2.16c}$$

$$\left\|\begin{bmatrix}\theta_k - \theta_\star \\ K_k - K_\star\end{bmatrix}\right\| \leq a_7 \left\|\begin{bmatrix}\theta_0 - \theta_\star \\ K_0 - K_\star\end{bmatrix}\right\| \exp(-a_8 k), \tag{2.16d}$$

for all $\gamma \in (0, \bar{\gamma})$. △

The proof of Theorem 2.1 is provided in the following sections.

Notice that the initialization in Theorem 2.1 does not necessarily require the knowledge of (A, B). Indeed, one can compute a stabilizing controller K_0 in a data-based fashion, see, e.g., [233] and the discussion in [155].

Remark 2.1. Since the size of the ball $\mathcal{B}_r(K_\star)$ heavily depends on problem data, e.g., the choice of cost matrices and Lipschitz constants of $J(\cdot)$, it may generally be challenging to ensure that the initial gain K_0 belongs to $\mathcal{B}_r(K_\star)$. However, by appropriately selecting the step size γ, it is possible to expand $\mathcal{B}_r(K_\star)$ to ensure that $K_0 \in \mathcal{B}_r(K_\star)$. △

The result (2.16a) of Theorem 2.1 ensures that $\Pi_x w_k$ is a practically exponentially stable equilibrium for (2.15b). Indeed Theorem 2.1 allows us to choose the initial condition of the exogenous system w_0 so that x_k exponentially converges into the ball $\mathcal{B}_\rho(0_n)$ for any desired radius $\rho > 0$. More in details, since w_k evolves according to a marginally stable oscillating dynamics (cf. Assumption 2.2), it holds $\|w_k\| = \|w_0\|$ for all $k \in \mathbb{N}$. Thus, in order to make $\mathcal{B}_\rho(0_n)$ attractive for the trajectories of (2.15b), it is sufficient to choose w_0 such that

$$\|w_0\| \leq \frac{\rho}{\|\Pi_x\|}.$$

Furthermore, the result (2.16d) ensures that $(K_\star, \theta_\star)$ is exponentially stable for (2.15e) and (2.15f). Hence, we asymptotically (i) reconstruct the unknown system matrices (A, B) and (ii) compute the optimal gain matrix $K_\star$.

We finally point out that, as a byproduct, Theorem 2.1 formally ensures the well-posedness of Algorithm 1. In fact, if the closed-loop matrix $A_k + B_k K_k$ were not Schur (recall that $\theta_k = \text{col}(A_k^\top, B_k^\top)$), then the gradient $\Gamma(K_k, \theta_k)$ would not be computable, so that the update in (2.9) becomes ill-posed. However, in light of the exponential stability claimed in (2.16), one can invoke the converse Lyapunov theorem (see, e.g., [13, Theorem 2.1.1]) to guarantee the existence of a Lyapunov function for System (2.15). Notice that (i) such a Lyapunov function is bounded from below and from above by quadratic functions and (ii) its level sets are invariant. Then, one can show that the cost variation $J(K_{k+1}, \theta_{k+1}) - J(K_k, \theta_k)$ is always bounded along trajectories of (2.15). In turn, this guarantees that $A_k + B_k K_k$ is Schur for all $k \geq 0$ and, hence, the well-posedness of Algorithm 1.

2.2.2 Algorithm Analysis

In this section, we perform a thorough analysis of Algorithm 1 leveraging on the stability analysis of the associated closed-loop dynamics. We first write the algorithm dynamics with respect to suitable error coordinates. Second, we resort to the averaging theory to prove the exponential stability of the origin for the averaged system associated to the error dynamics. This result is then exploited to prove Theorem 2.1.

Closed-Loop Dynamics in Error Coordinates

As a preliminary step, System (2.15) is expressed into suitably defined error coordinates. First, we consider vectorized versions of the matrix updates in (2.15c)-(2.15d). To this end, let the new coordinates $H^{\text{vc}} \in \mathbb{R}^{(n+m)^2}$ and $S^{\text{vc}} \in \mathbb{R}^{(n+m)n}$ be defined as

$$\begin{cases} H_k \\ S_k \end{cases} \longmapsto \begin{cases} H_k^{\text{vc}} := \texttt{vec}\left(H_k\right) \\ S_k^{\text{vc}} := \texttt{vec}\left(S_k\right). \end{cases} \tag{2.17}$$

Therefore, (2.15c)-(2.15d) can be recast as

$$H_{k+1}^{\text{vc}} = \lambda H_k^{\text{vc}} + \texttt{vec}\left(\begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix}\begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix}^\top\right) \tag{2.18a}$$

$$S_{k+1}^{\text{vc}} = \lambda S_k^{\text{vc}} + \texttt{vec}\left(\begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix}\begin{bmatrix} x_k \\ K_k x_k + E w_k \end{bmatrix}^\top \theta_\star\right). \tag{2.18b}$$

Next, we will inspect (2.18) together with (2.15b) to provide the steady-state locus (see, e.g., [120, Ch. 12] for a formal definition) when the system is fed with the signal w_k, which evolves according to (2.15a). To this end, set $n_\chi := n + (n+m)^2 + (n+m)n$

and let $\chi \in \mathbb{R}^{n_\chi}$ be defined as

$$\chi := \begin{bmatrix} x \\ H^{\text{vc}} \\ S^{\text{vc}} \end{bmatrix}.$$

Then, using (2.18), the dynamics in (2.15a)-(2.15d) can be compactly expressed in the new coordinates as

$$w_{k+1} = F w_k \tag{2.19a}$$

$$\chi_{k+1} = \mathcal{A}_K(K_k)\chi_k + \phi(\chi_k, K_k, w_k) \tag{2.19b}$$

where, we introduced $\mathcal{A}_K : \mathbb{R}^{m\times n} \to \mathbb{R}^{n_\chi \times n_\chi}$ and $\phi : \mathbb{R}^{n_\chi} \times \mathbb{R}^{m\times n} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_\chi}$ be defined as

$$\mathcal{A}_K(K) := \begin{bmatrix} A_\star + B_\star K & 0 & 0 \\ 0 & \lambda I & 0 \\ 0 & 0 & \lambda I \end{bmatrix} \tag{2.20a}$$

$$\phi(\chi, K, w) := \begin{bmatrix} B_\star E w \\ \texttt{vec}\left(\begin{bmatrix} x \\ Kx+Ew \end{bmatrix} \begin{bmatrix} x \\ Kx+Ew \end{bmatrix}^\top \right) \\ \texttt{vec}\left(\begin{bmatrix} x \\ Kx+Ew \end{bmatrix} \begin{bmatrix} x \\ Kx+Ew \end{bmatrix}^\top \theta_\star \right) \end{bmatrix}. \tag{2.20b}$$

Notice that to keep the notation light, we use a hybrid notation with χ on the left-hand side and its (unvectorized) components (x, H, S) on the right-hand side.

System (2.19) together with the exosystem (2.15a) is a cascade whose steady-state locus can be characterized by the nonlinear map $\chi^{\text{ss}} : \mathbb{R}^{n_w} \to \mathbb{R}^{n_\chi}$ defined as

$$\chi^{\text{ss}}(w) := \begin{bmatrix} \Pi_x w \\ \Pi_H \texttt{vec}\left(ww^\top\right) \\ \Pi_S \texttt{vec}\left(ww^\top\right) \end{bmatrix}, \tag{2.21}$$

where Π_x, Π_H, and Π_S are the same as in Theorem 2.1 (see (2.25) and (2.29) for their explicit definition). Formally, the following lemma holds true.

Lemma 2.1. *Let the assumptions of Theorem 2.1 hold true. Consider the map χ^{ss} defined in* (2.21)*, the feedback gain $K_\star$ solving* (2.3)*, the matrix* $\mathbf{f}$ *as in* (2.13)*, and the functions $\mathcal{A}_K$ and ϕ defined in* (2.20)*. Then, it holds*

$$\chi^{\text{ss}}(Fw) = \mathcal{A}_K(K_\star)\chi^{\text{ss}}(w) + \phi(\chi^{\text{ss}}(w), K_\star, w), \tag{2.22}$$

for all $w \in \mathbb{R}^{n_w}$. Moreover,

$$(\theta_\star{}^\top \otimes I_{n+m})\Pi_H = \Pi_S. \tag{2.23}$$

holds true. △

Proof. We note that (2.22) is formally obtained by setting $K_k = K_\star$ in System (2.19) (which compactly collects the updates (2.15a), (2.15b), and (2.18)). Hence, we start by inspecting (2.15a) and (2.15b) restricted to the manifold in which $K_k = K_\star$, which gives

$$w_{k+1} = Fw_k \tag{2.24a}$$
$$x_{k+1} = (A_\star + B_\star K_\star)x_k + B_\star E w_k. \tag{2.24b}$$

System (2.24) is a cascade, therefore its steady-state solution is $\mathrm{col}(w_k, x_k) = \mathrm{col}(I_{n_w}, \Pi_x)w_k$, with $\Pi_x \in \mathbb{R}^{n\times m}$ solution to the following Sylvester equation

$$\Pi_x F = (A_\star + B_\star K_\star)\Pi_x + B_\star E. \tag{2.25}$$

Being F marginally stable (cf. Assumption 2.2) and $A_\star + B_\star K_\star$ Schur (so that $\sigma(F) \cap \sigma(A_\star + B_\star K_\star) = \emptyset$) the solution Π_x exists and is unique. Then, we inspect the dynamics of System (2.18) restricted to the manifold in which $x_k = \Pi_x w_k$ and $K_k = K_\star$. Let the matrix $M \in \mathbb{R}^{(n+m)\times(n+m)}$ be defined as

$$M := \begin{bmatrix} \Pi_x \\ K_\star \Pi_x + E \end{bmatrix}, \tag{2.26}$$

then it holds

$$\mathtt{vec}\left(w_{k+1}w_{k+1}^\top\right) = \mathtt{vec}\left(Fw_k w_k^\top F^\top\right) \tag{2.27a}$$
$$H_{k+1}^{\mathrm{vc}} = \lambda H_k^{\mathrm{vc}} + \mathtt{vec}\left(Mw_k w_k^\top M^\top\right) \tag{2.27b}$$
$$S_{k+1}^{\mathrm{vc}} = \lambda S_k^{\mathrm{vc}} + \mathtt{vec}\left(Mw_k w_k^\top M^\top \theta_\star\right), \tag{2.27c}$$

where the first equation comes from the vectorization of (2.15a). By exploiting the vectorization properties[1], we can manipulate (2.27) to obtain the system

$$\mathtt{vec}\left(w_{k+1}w_{k+1}^\top\right) = (F \otimes F)\mathtt{vec}\left(w_k w_k^\top\right) \tag{2.28a}$$
$$H_{k+1}^{\mathrm{vc}} = \lambda H_k^{\mathrm{vc}} + (M \otimes M)\mathtt{vec}\left(w_k w_k^\top\right) \tag{2.28b}$$

[1]Given any $X_1 \in \mathbb{R}^{n_1\times n_2}$, $X_2 \in \mathbb{R}^{n_2\times n_3}$, and $X_3 \in \mathbb{R}^{n_3\times n_4}$, it holds $\mathtt{vec}\,(X_1X_2X_3) = (X_3^\top \otimes X_1)\mathtt{vec}\,(X_2)$.

$$S_{k+1}^{\text{vc}} = \lambda S_k^{\text{vc}} + ({\theta_\star}^\top M \otimes M)\texttt{vec}\left(w_k w_k^\top\right), \tag{2.28c}$$

which enjoys again a cascade structure. Thus, let $\Pi_H \in \mathbb{R}^{(n+m)^2 \times n_w^2}$ and $\Pi_S \in \mathbb{R}^{(n+m)n \times n_w^2}$ be the solution to the Sylvester equations associated to the cascade in (2.28) given by

$$\Pi_H(F \otimes F) = \lambda \Pi_H + M \otimes M \tag{2.29a}$$

$$\Pi_S(F \otimes F) = \lambda \Pi_S + ({\theta_\star}^\top M) \otimes M. \tag{2.29b}$$

Being F marginally stable (cf. Assumption 2.2) and $\lambda \in (0,1)$, then $\sigma(F \otimes F) \cap \sigma(\lambda I) = \emptyset$ and, thus, the solutions Π_S and Π_H to (2.29) exist and are unique. The proof of (2.22) follows by (i) noticing that (Π_x, Π_H, Π_S) are used to define χ^{ss} (cf. (2.21)), and (ii) plugging (2.25) and (2.29) into System (2.19).

As for (2.23), it can be shown using an algebraic manipulation of (2.29). Indeed, by premultiplying (2.29a) by ${\theta_\star}^\top \otimes I_{n+m}$ we can write

$$\begin{aligned}({\theta_\star}^\top \otimes I_{n+m})\Pi_H(F \otimes F - \lambda I) &= ({\theta_\star}^\top \otimes I_{n+m})(M \otimes M)\\ &\overset{(a)}{=} ({\theta_\star}^\top M) \otimes M\\ &\overset{(b)}{=} \Pi_S(F \otimes F - \lambda I),\end{aligned} \tag{2.30}$$

where in (a) we used the mixed-product property of the Kronecker operator[2], while (b) follows from (2.29b). So that the proof is complete. ■

It is worth noting that, despite the fact that System (2.19) is fed by K_k, the expression (2.22) involves $K_\star$. Indeed, as one may expect, the pair $(\theta_\star, K_\star)$ is an equilibrium of (2.15e)-(2.15f) when $\chi_k = \chi^{\text{ss}}(w_k)$. Thus, it makes sense to provide the steady state in terms of $K_\star$ rather than K_k Also, condition (2.23) turns out to be very useful to simplify the update (2.15e) when H_k and S_k lie in the steady-state locus. Indeed, when $\chi_k = \chi^{\text{ss}}(w_k)$ the driving term $-\gamma H_k^\dagger (H_k\theta_k - S_k)$ of the integrator dynamics in (2.15e) can be simplified by noticing that

$$\begin{aligned}(H_k\theta_k - S_k)\Big|_{\chi_k=\chi^{\text{ss}}(w_k)} &= \texttt{unvec}\left(\Pi_S \texttt{vec}\left(w_k w_k^\top\right)\right)\theta_k\\ &\quad - \texttt{unvec}\left(\left({\theta_\star}^\top \otimes I_{n+m}\right)\Pi_H \texttt{vec}\left(w_k w_k^\top\right)\right)\\ &\overset{(a)}{=} \texttt{unvec}\left(\Pi_H \texttt{vec}\left(w_k w_k^\top\right)\right)(\theta_k - \theta_\star),\end{aligned}$$

where in (a) we used a property of the vectorization operator.[3] From the above equality, one can notice that the mentioned driving term vanishes in steady state if $\theta_k = \theta_\star$, i.e.,

[2]Given any $X_1 \in \mathbb{R}^{n_1 \times n_2}$, $X_2 \in \mathbb{R}^{n_3 \times n_4}$, $X_3 \in \mathbb{R}^{n_2 \times n_5}$, and $X_4 \in \mathbb{R}^{n_4 \times n_6}$, it holds $(X_1 \otimes X_2)(X_3 \otimes X_4) = (X_1X_3) \otimes (X_2X_4)$.

[3]Given any two matrices $X_1 \in \mathbb{R}^{n_1 \times n_2}$ and $X_2 \in \mathbb{R}^{n_2 \times n_3}$, it holds $\texttt{vec}\,(X_1X_2) = (X_2^\top \otimes I_{n_1})\texttt{vec}\,(X_1)$.

when perfect estimation is achieved.

As it will become useful later, we also introduce $H_k^{\text{ss}} \in \mathbb{R}^{(n+m)\times(n+m)}$ defined as

$$H_k^{\text{ss}} := \texttt{unvec}\left(\Pi_H \texttt{vec}\left(w_k w_k^\top\right)\right), \tag{2.31}$$

which represents the steady-state value of the state H_k.

Before proceeding, let us collect also the remaining states in (2.15) in $z \in \mathbb{R}^{n_z}$, with $n_z := (n+2m)\times n$, defined as

$$z := \begin{bmatrix} \theta \\ K \end{bmatrix}.$$

In order to prove Theorem 2.1, we need to show the convergence of χ and z toward $\chi^{\text{ss}}(w)$ and $\operatorname{col}(\theta_\star, K_\star)$, respectively. Therefore, with Lemma 2.1 at hand, let us introduce error coordinates $\tilde{\chi} \in \mathbb{R}^{n_\chi}$ and $\tilde{z} \in \mathbb{R}^{n_z}$ given by the following change of coordinates

$$\begin{cases} w \\ \chi \\ z \end{cases} \longmapsto \begin{cases} w \\ \tilde{\chi} := \begin{bmatrix} I & 0 & 0 \\ 0 & \gamma I & 0 \\ 0 & 0 & \gamma I \end{bmatrix} (\chi - \chi^{\text{ss}}(w)) \\ \tilde{z} := z - \begin{bmatrix} \theta_\star \\ K_\star \end{bmatrix}. \end{cases} \tag{2.32}$$

For notational convenience, we will sometimes refer to the components of $\tilde{z}$ as $\operatorname{col}(\tilde{\theta}, \tilde{K})$. Finally, the closed-loop dynamics (2.15) in the new coordinates (2.32) reads as

$$\tilde{\chi}_{k+1} = \mathcal{A}(\tilde{z}_k)\tilde{\chi}_k + h(\tilde{z}_k, k) + \gamma g(\tilde{\chi}_k, \tilde{z}_k, k) \tag{2.33a}$$

$$\tilde{z}_{k+1} = \tilde{z}_k + \gamma f(\tilde{\chi}_k, \tilde{z}_k, k), \tag{2.33b}$$

where $\mathcal{A}(\tilde{z}) := \mathcal{A}_K(\tilde{K} + K_\star)$ and we introduced $h : \mathbb{R}^{n_z} \times \mathbb{N} \to \mathbb{R}^{n_z}$, $g : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_z} \times \mathbb{N} \to \mathbb{R}^{n_\chi}$, and $f : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_z} \times \mathbb{N} \to \mathbb{R}^{n_z}$ defined respectively as

$$h(\tilde{z}, k) := \begin{bmatrix} B_\star \tilde{K} \Pi_x w_k \\ 0 \\ 0 \end{bmatrix} \tag{2.34a}$$

$$g(\tilde{\chi}, \tilde{z}, k) := \begin{bmatrix} 0 \\ \gamma\phi(\tilde{x} + \Pi_x w_k, \tilde{K} + K_\star, w_k) \end{bmatrix} \tag{2.34b}$$

$$f(\tilde{\chi}, \tilde{z}, k) := -\begin{bmatrix} (\tilde{H} + H_k^{\text{ss}})^\dagger \left((\tilde{H} + H_k^{\text{ss}})\tilde{\theta} + (\tilde{H} - \tilde{S})\theta_\star\right) \\ \Gamma(\tilde{K} + K_\star, \tilde{\theta} + \theta_\star) \end{bmatrix}. \tag{2.34c}$$

Notice that for the sake of readability, in (2.34) we used the shorthands $\tilde{\chi} := \mathrm{col}(\tilde{x}, \tilde{H}, \tilde{S})$ and $\tilde{z} = \mathrm{col}(\tilde{\theta}, \tilde{K})$, while H_k^{ss} is defined in (2.31).

Some remarks are in order. We point out that with this transformation we obtained a dynamical system with two-time scales as the one described in Appendix C (cf. System (C.1)). As customary in the context of singularly perturbed systems, we distinguish between (i) the fast dynamics (2.33a) with state $\tilde{\chi}$, and (ii) the slow one (2.33b) with state $\tilde{z}$. Figure 2.2 shows the mentioned interconnected structure of System (2.33).

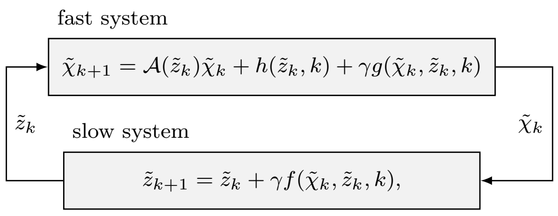

fast system
$\tilde{\chi}_{k+1} = \mathcal{A}(\tilde{z}_k)\tilde{\chi}_k + h(\tilde{z}_k, k) + \gamma g(\tilde{\chi}_k, \tilde{z}_k, k)$
$\tilde{z}_k$
slow system
$\tilde{\chi}_k$
$\tilde{z}_{k+1} = \tilde{z}_k + \gamma f(\tilde{\chi}_k, \tilde{z}_k, k),$

Figure 2.2: Block diagram describing System (2.33).

It is also worth noting that, in this reformulation, the effect of the exogenous/dithering signal w_k has been embedded in the time-dependency of h, g, and f. As for the equilibrium manifold, we observe that, for all $k \in \mathbb{N}$, it holds

$$h(0, k) = 0, \qquad g(0, 0, k) = 0, \qquad f(0, 0, k) = 0. \tag{2.35}$$

Finally, the matrix $\mathcal{A}(\tilde{z})$ is Schur for all $\tilde{K} \in \mathbb{R}^{m \times n}$ such that $(\tilde{K} + K_\star) \in \mathcal{B}_{r^\star}(K_\star)$.

Averaged System Analysis

Next, we carry out the stability analysis of the time-varying System (2.33) by leveraging on the averaging and singular perturbations theories (cf. Appendix C for further details). Indeed, since System (2.33) enjoys a two-time-scale structure (cf. the generic System (C.1)), we can study (2.33) by only investigating an auxiliary system typically termed as the *averaged system*. The latter is obtained by considering the slow dynamics (2.33b) in which (i) the fast state is frozen to its equilibrium, i.e., with $\tilde{\chi}_k = 0$ for all $k \geq 0$, and (ii) the vector field describing the dynamics is averaged with respect to time.

The following results are instrumental to properly formulate the averaged system.

Lemma 2.2. *Let the assumptions of Theorem 2.1 hold true. Then, the matrix H_k^{ss}, defined in* (2.31), *is invertible for all $k \in \mathbb{N}$.*

We recall that H_k^{ss} is the unvectorized version of the second block-component of $\chi^{\mathrm{ss}}(w_k)$ (see (2.21) and (2.31)).

Proof. We will prove the invertibility property of the matrix H_k^{ss} by investigating the evolution of H_k. The dynamics of H_k in (2.15c) restricted to the manifold in which

$x_k = \Pi_x w_k$ and $K_k = K_\star$ reads as

$$H_{k+1} = \lambda H_k + M w_k w_k^\top M^\top, \tag{2.36}$$

with M as in (2.26). The explicit solution of (2.36) is

$$H_k = \lambda^k H_0 + \underbrace{M \left(\sum_{i=0}^{k-1} \lambda^{k-1-i} w_i w_i^\top \right)}_{\mathcal{W}_k} M^\top. \tag{2.37}$$

Being $\lambda \in (0,1)$, the free evolution $\lambda^k H_0$ in (2.37) vanishes as $k \to \infty$. Hence, it does not impact on the invertibility of the steady-state solution.

Therefore, let us focus on the forced response $M\mathcal{W}_k M^\top$ only. We first notice that $\alpha_1 I_{n_w} \leq \sum_{i=\bar{k}+1}^{\bar{k}+k_w} w_i w_i^\top \leq \alpha_2 I_{n_w}$ for all $\bar{k} \in \mathbb{N}$ (cf. Assumption 2.2). Hence we can invoke [128, Lemma 1] to assert the positive definiteness of $\mathcal{W}_k$ for all $k \geq k_w$. Let us consider the Cholesky decomposition of $\mathcal{W}_k$ given by $\mathcal{W}_k = \mathcal{C}_k \mathcal{C}_k^\top$, with $\mathcal{C}_k \in \mathbb{R}^{n_w \times n_w}$ invertible. Then[4], for all $k \geq k_w$, we can write

$$\begin{aligned} \operatorname{rank}\left(M\mathcal{W}_k M^\top\right) &= \operatorname{rank}(M\mathcal{C}_k) \\ &\overset{(a)}{=} \operatorname{rank}(M), \end{aligned} \tag{2.38}$$

where in (a) we used the full-rankness of $\mathcal{C}_k$ and a property of the rank operator.[5] In order to compute $\operatorname{rank}(M)$, we consider again the dynamics in (2.15a) and (2.15b) restricted to the manifold in which $K_k = K_\star$, namely

$$w_{k+1} = F w_k \tag{2.39a}$$

$$x_{k+1} = (A_\star + B_\star K_\star) x_k + B_\star E w_k. \tag{2.39b}$$

Recalling that $\mathrm{d}_k = E w_k$ satisfies condition (2.14b) (cf. Assumption 2.2) and that the pair (A, B) is controllable (cf. Assumption 2.1), we can invoke [238, Cor. 2] to claim that

$$\operatorname{rank}\left(\begin{bmatrix} x_0 & \dots & x_{\mathbf{t}_\mathrm{d}-1} \\ \mathrm{d}_0 & \dots & \mathrm{d}_{\mathbf{t}_\mathrm{d}-1} \end{bmatrix}\right) = n + m, \tag{2.40}$$

for all $(x_0, \mathrm{d}_0) \in \mathbb{R}^n \times \mathbb{R}^m$. When the initial condition of (2.39b) lies in the invariant steady-state locus (cf. (2.25)), i.e., when $x_0 = \Pi_x w_0$, the condition in (2.40) simplifies to

$$\operatorname{rank}\left(M \begin{bmatrix} w_0 & \dots & w_{\mathbf{t}_\mathrm{d}-1} \end{bmatrix}\right) = n + m, \tag{2.41}$$

[4]Given any $X \in \mathbb{R}^{n \times m}$, it holds $\operatorname{rank}(XX^\top) = \operatorname{rank}(X) = \operatorname{rank}(X^\top)$.

[5]Given $X_1 \in \mathbb{R}^{n_1 \times n_2}$ and $X_2 \in \mathbb{R}^{n_2 \times n_3}$ it holds $\operatorname{rank}(X_1 X_2) = \operatorname{rank}(X_1)$ if $\operatorname{rank}(X_2) = n_2$.

with M as in (2.26). Equation (2.41) allows us to conclude that[6]

$$\mathrm{rank}(M) \geq n + m.$$

Moreover, being $\mathrm{rank}(M) \leq n + m$ by construction, the above inequality yields to $\mathrm{rank}(M) = n + m$, which, in turn, combined with (2.38), allows us to write

$$\mathrm{rank}\left(M\mathcal{W}_k M^\top\right) = n + m, \tag{2.42}$$

for all $k \geq k_w$. Next, we characterize the $\mathrm{rank}(M\mathcal{W}_k M^\top)$ after the transient phase. Being $\lambda \in (0, 1)$, it holds that $M\mathcal{W}_k M^\top$ exponentially converges to H_k^{ss}. Therefore, using a continuity argument there must exist $k_\infty \geq k_w$ such that also

$$\mathrm{rank}(H_k^{\mathrm{ss}}) = n + m,$$

for all $k \geq k_\infty$. Finally, being H_k^{ss} a static function of the periodic signal w_k, then H_k^{ss} is periodic as well so that its full-rankness is independent of k. Therefore, it must be that $\mathrm{rank}(H_k^{\mathrm{ss}}) = n + m$ for all $k \in \mathbb{N}$, and the proof follows. ■

Lemma 2.3. *Let the assumptions of Theorem 2.1 hold true. Consider f defined in* (2.34c). *Then, it holds*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} f(0, \tilde{z}, i) = -\begin{bmatrix} \tilde{\theta} \\ \Gamma(\tilde{K} + K_\star, \tilde{\theta} + \theta_\star) \end{bmatrix} \tag{2.43}$$

uniformly in $\bar{k} \in \mathbb{N}$ and for all $\tilde{z} = \mathrm{col}(\tilde{\theta}, \tilde{K}) \in \mathbb{R}^{n_z}$. △

Proof. We start recalling that, by Lemma 2.2, the invertibility of H_k^{ss} is established. Let us label the two components of f^{av} as

$$\begin{bmatrix} f_1^{\mathrm{av}}(\tilde{z}) \\ f_2^{\mathrm{av}}(\tilde{z}) \end{bmatrix} := \lim_{T\to\infty} \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} f(0, \tilde{z}, i).$$

As for $f_1^{\mathrm{av}}(z)$, we can write

$$f_1^{\mathrm{av}}(\tilde{z}) = -\lim_{T\to\infty} \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} (H_i^{\mathrm{ss}})^\dagger H_i^{\mathrm{ss}} \tilde{\theta} \overset{(a)}{=} -\tilde{\theta},$$

where in (a) we used Lemma 2.2 to guarantee the invertibility of H_k^{ss} for all $k \in \mathbb{N}$. As for $f_2^{\mathrm{av}}(z)$, its existence is trivially shown by observing that it does not depend on k.

[6]Given $X_1 \in \mathbb{R}^{n_1 \times n_2}$ and $X_2 \in \mathbb{R}^{n_2 \times n_3}$, it holds $\mathrm{rank}(X_1 X_2) \leq \min\{\mathrm{rank}(X_1), \mathrm{rank}(X_2)\}$.

Hence, it holds

$$f_2^{\text{av}}(\tilde{z}) := -\Gamma(\tilde{K} + K_\star, \tilde{\theta} + \theta_\star),$$

where we used $\tilde{z} = \text{col}(\tilde{K}, \tilde{\theta})$. ■

Lemma 2.3 provides a suitable approximation of the dynamics of $\tilde{z}$ in (2.33b) when (i) the convergence of the fast state $\tilde{\chi}$ to its equilibrium has already occurred and (ii) by averaging over time k the vector field $f(0, \tilde{z}, k)$. Specifically, under this approximation, Lemma 2.3 ensures that the two components of the driving term of the dynamics of $\tilde{z}$ are given by (i) a proportional term $-\gamma\tilde{\theta}$ and (ii) an approximate version of the correct gradient $\Gamma(\tilde{K} + K_\star, \theta_\star)$. Next, we will leverage averaging theory to prove the stability of the origin for System (2.33).

Once the averaged vector field has been characterized in Lemma 2.3, we can introduce $f^{\text{av}} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ given by

$$f^{\text{av}}(\tilde{z}) := \lim_{T\to\infty} \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} f(0, \tilde{z}, k) = -\begin{bmatrix} \tilde{\theta} \\ \Gamma(\tilde{K} + K_\star, \tilde{\theta} + \theta_\star) \end{bmatrix},$$

in which we used $\tilde{z} = \text{col}(\tilde{\theta}, \tilde{K})$. Then, we define the averaged system, with state $\tilde{z}_k^{\text{av}} \in \mathbb{R}^{n_z}$, associated to (2.33) as

$$\tilde{z}_{k+1}^{\text{av}} = \tilde{z}_k^{\text{av}} + \gamma f^{\text{av}}(\tilde{z}_k^{\text{av}}). \tag{2.44}$$

Exploding the expression of f^{av} (cf. (2.43)) and $\tilde{z}_k := \text{col}(\tilde{\theta}_k, \tilde{K}_k)$, the dynamics in (2.44) results in a cascade as depicted in Figure 2.3.

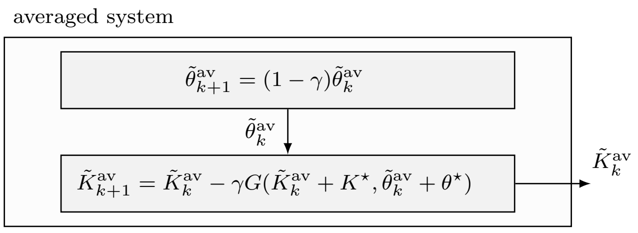

averaged system
$\tilde{\theta}_{k+1}^{\text{av}} = (1-\gamma)\tilde{\theta}_k^{\text{av}}$
$\tilde{\theta}_k^{\text{av}}$
$\tilde{K}_{k+1}^{\text{av}} = \tilde{K}_k^{\text{av}} - \gamma G(\tilde{K}_k^{\text{av}} + K^\star, \tilde{\theta}_k^{\text{av}} + \theta^\star)$
$\tilde{K}_k^{\text{av}}$

Figure 2.3: Block diagram describing System (2.44) with $\tilde{z}_k^{\text{av}} = \text{col}(\tilde{\theta}_k^{\text{av}}, \tilde{K}_k^{\text{av}})$.

The dynamics of $\tilde{\theta}_k^{\text{av}}$ is trivially exponentially convergent to zero, while in the following we will formally show that the dynamics of $\tilde{K}_k^{\text{av}}$ is input-to-state (ISS) exponentially stable (cf. [107]).

For the sake of compactness, let us also introduce the (averaged) estimates $A_k^{\text{av}} \in$

$\mathbb{R}^{n\times n}$ and $B_k^{\text{av}} \in \mathbb{R}^{n\times m}$ of the matrices A and B, defined as

$$\begin{bmatrix} A_k^{\text{av}} & B_k^{\text{av}} \end{bmatrix}^\top := \tilde{\theta}_k^{\text{av}} + \theta_\star, \tag{2.45}$$

where we recall that $\tilde{\theta}_k^{\text{av}}$ is the first component of $\tilde{z}_k^{\text{av}}$. Under the same assumptions of Theorem 2.1, the next result establishes exponential stability of the origin for System (2.44).

Proposition 2.1. *Let the assumptions of Theorem 2.1 hold true. Consider the averaged System* (2.44)*. Then, for all $\tilde{z}_0^{av} \in \mathbb{R}^{(n+2m)\times n}$ such that the corresponding $A_0^{av}+B_0^{av}(\tilde{K}_0^{av}+K_\star)$ and $A_\star + B_\star(\tilde{K}_0^{av}+K_\star)$ are Schur matrices, there exists $\bar{\gamma}^{av} > 0$ such that, for all $\gamma \in (0, \bar{\gamma}^{av})$, the origin of* (2.44) *is exponentially stable.* △

Proof. The proof resorts to a suitable Lyapunov candidate function whose increment along trajectories of System (2.44) will allow us to claim exponential stability of the origin. To ease the notation, we start by decomposing the state of (2.44) as $\tilde{z}_k^{\text{av}} := \operatorname{col}(\tilde{\theta}_k^{\text{av}}, \tilde{K}_k^{\text{av}})$. Then, we recall [48, Lemma 3.12] to guarantee that the cost function J, defined in (2.5), is gradient dominated, that is for all $K \in \mathcal{D}$ it holds

$$J(K,\theta_\star) - J(K_\star,\theta_\star) \leq \mu \left\|\Gamma(K,\theta_\star)\right\|^2, \tag{2.46}$$

for some $\mu > 0$, where Γ denotes the gradient of J. Now, let us consider the Lyapunov candidate function $V : \mathbb{R}^{m\times n} \times \mathbb{R}^{(n+m)\times n} \to \mathbb{R}$ defined as

$$V(\tilde{K}^{\text{av}}, \tilde{\theta}^{\text{av}}) := \kappa\left(J(\tilde{K}^{\text{av}} + K_\star, \theta_\star) - J(K_\star,\theta_\star)\right) + \tfrac{1}{2}\left\|\tilde{\theta}^{\text{av}}\right\|^2, \tag{2.47}$$

with $\kappa > 0$, whose specific value will be set later. Being $K_\star$ the unique minimizer of $J(\cdot,\theta_\star)$ [48], we note that V is positive definite. Now, given any $c > 0$, let us introduce the level set $\Omega_c \subset \mathbb{R}^{m\times m} \times \mathbb{R}^{(n+m)\times n}$ of V, defined as

$$\Omega_c := \left\{(\tilde{K}^{\text{av}}, \tilde{\theta}^{\text{av}}) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{(n+m)\times n} \mid J(\tilde{K}^{\text{av}} + K_\star,\theta_\star) - J(K_\star,\theta_\star) + \tfrac{1}{2}\left\|\tilde{\theta}^{\text{av}}\right\|^2 \leq c\right\}.$$

Let $c_0 > 0$ be the smallest number such that $(\tilde{K}_0^{\text{av}}, \tilde{\theta}_0^{\text{av}}) \in \Omega_{c_0}$ and define

$$\beta_1 := \max_{(\tilde{K}^{\text{av}},\tilde{\theta}^{\text{av}})\in\Omega_{c_0}} \left\{\left\|\frac{\partial\Gamma(\tilde{K}^{\text{av}}+K_\star, \tilde{\theta}^{\text{av}}+\theta_\star)}{\partial\tilde{K}^{\text{av}}}\right\|\right\} \tag{2.48a}$$

$$\beta_2 := \max_{(\tilde{K}^{\text{av}},\tilde{\theta}^{\text{av}})\in\Omega_{c_0}} \left\{\left\|\frac{\partial\Gamma(\tilde{K}^{\text{av}}+K_\star, \tilde{\theta}^{\text{av}}+\theta_\star)}{\partial\tilde{\theta}^{\text{av}}}\right\|\right\}. \tag{2.48b}$$

In light of [48, Proposition 3.10], it holds that $\frac{\partial\Gamma(\tilde{K}^{\text{av}}+K_\star, \tilde{\theta}^{\text{av}}+\theta_\star)}{\partial\tilde{K}^{\text{av}}}$ is a continuous function of $\tilde{K}^{\text{av}}$. Similarly, also continuity of $\frac{\partial\Gamma(\tilde{K}^{\text{av}}+K_\star, \tilde{\theta}^{\text{av}}+\theta_\star)}{\partial\tilde{\theta}^{\text{av}}}$ with respect to $\tilde{\theta}^{\text{av}}$ can be shown.

Hence, (β_1, β_2) are well posed, i.e., finite. We remark that [48, Corolllary 3.7.1] guarantees that, given any $c > 0$, the level set of the cost function J, namely $\{\tilde{K} \in \mathbb{R}^{m\times n} \mid J(\tilde{K}^{\text{av}} + K_\star, \theta_\star) - J(K_\star, \theta_\star) \leq c\} \subset \mathbb{R}^{m\times n}$, is compact and, thus, so is Ω_c.

Next, we show that Ω_{c_0} is (forward) invariant for System (2.44). To this end, assume that $(\tilde{K}^{\text{av}}_k, \tilde{\theta}^{\text{av}}_k) \in \Omega_{c_0}$ and let us prove the invariance of Ω_{c_0} using an induction argument.

Recall that, the cost $J(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star)$ is finite for all $\tilde{z}^{\text{av}}_k \in \Omega_{c_0}$, and, hence, iteration (2.44) is well-posed. The increment ΔV of V along trajectories of System (2.44) is given by

$$\begin{aligned}
&\Delta V(\tilde{K}^{\text{av}}_k, \tilde{\theta}^{\text{av}}_k) := V(\tilde{K}^{\text{av}}_{k+1}, \tilde{\theta}^{\text{av}}_{k+1}) - V(\tilde{K}^{\text{av}}_k, \tilde{\theta}^{\text{av}}_k)\\
&= \kappa\Big(J(\tilde{K}^{\text{av}}_{k+1} + K_\star, \theta_\star) - J(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\Big) - \gamma\,(1-\gamma/2)\left\|\tilde{\theta}^{\text{av}}_k\right\|^2\\
&\overset{(a)}{\leq} \kappa J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star), \theta_\star) - \kappa J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star), \theta_\star)\\
&\quad + \kappa J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star), \theta_\star) - \kappa J(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star) - \gamma\,(1-\gamma/2)\left\|\tilde{\theta}^{\text{av}}_k\right\|^2\\
&\overset{(b)}{\leq} \kappa J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star), \theta_\star) - \kappa J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star), \theta_\star)\\
&\quad - \gamma\kappa\left(1 - \gamma\frac{\beta_1}{2}\right)\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|^2 - \gamma\,(1-\gamma/2)\left\|\tilde{\theta}^{\text{av}}_k\right\|^2,
\end{aligned} \tag{2.49}$$

where in (a) we used the update of $\tilde{K}^{\text{av}}_{k+1}$ and added $\pm\kappa J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star), \theta_\star)$, in (b) we used the Taylor expansion of $J(\cdot,\cdot)$ about $(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)$ evaluated at $(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star), \theta_\star)$ and used (2.48a)..

Next, we manipulate the difference between the first two terms in (2.49). By expanding $J(\cdot,\cdot)$ about $(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)$ evaluated at $(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star), \theta_\star)$ and $(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star), \theta_\star)$ and using (2.48a) and the Cauchy-Schwarz inequality, we can write

$$\begin{aligned}
&J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star), \theta_\star) - J(\tilde{K}^{\text{av}}_k + K_\star - \gamma\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star), \theta_\star)\\
&\leq \gamma\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star) - \Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|\\
&\quad + \frac{\gamma^2\beta_1}{2}\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star)\right\|^2 + \frac{\gamma^2\beta_1}{2}\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|^2\\
&\overset{(a)}{\leq} \gamma\beta_2\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|\left\|\tilde{\theta}^{\text{av}}_k\right\| + \frac{\gamma^2\beta_1}{2}\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \tilde{\theta}^{\text{av}}_k + \theta_\star) \pm \Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|^2\\
&\quad + \frac{\gamma^2\beta_1}{2}\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|^2\\
&\overset{(b)}{\leq} \gamma\beta_2\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|\left\|\tilde{\theta}^{\text{av}}_k\right\| + \gamma^2\beta_1\beta_2\left\|\tilde{\theta}^{\text{av}}_k\right\|^2 + \gamma^2\beta_1\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|^2\\
&\quad + \frac{\gamma^2\beta_1}{2}\left\|\Gamma(\tilde{K}^{\text{av}}_k + K_\star, \theta_\star)\right\|^2,
\end{aligned} \tag{2.50}$$

where in (a) we exploited the Lipschitz continuity expressed on (2.48b) and added

$\pm\Gamma(\tilde{K}_k^{\text{av}} + K_\star, \theta_\star)$ inside the norm of the second term, while in (b) we exploited again the Lipschitz continuity and a standard property of the square norm.[7] Plugging the bound in (2.50) into (2.49) and restricting $\kappa \in (0,1)$, we get

$$\begin{aligned}\Delta V(\tilde{K}_k^{\text{av}}, \tilde{\theta}_k^{\text{av}}) \leq -\gamma\kappa\left(1-\gamma\frac{3\beta_1}{2}\right)\left\|\Gamma(\tilde{K}_k^{\text{av}} + K_\star, \theta_\star)\right\|^2 - \gamma\left(1-\gamma\frac{1+\beta_1\beta_2^2}{2}\right)\left\|\tilde{\theta}_k^{\text{av}}\right\|^2, \\ + \gamma\kappa\beta_2\left\|\Gamma(\tilde{K}_k^{\text{av}} + K_\star, \theta_\star)\right\|\left\|\tilde{\theta}_k^{\text{av}}\right\| \end{aligned}\tag{2.51}$$

where we used $\kappa\beta_1\beta_2^2 \leq \beta_1\beta_2^2$.

Let us consider two arbitrary scalars $\nu_1, \nu_2 \in (0,1)$. Then, for all $\gamma \in (0, \bar{\gamma}^{\text{av}})$ with $\bar{\gamma}^{\text{av}} := \min\left\{1, \frac{2\nu_1}{3\beta_1}, \frac{2\nu_2}{1+\beta_1\beta_2^2}\right\}$, the inequality (2.51) can be manipulated as

$$\begin{aligned}\Delta V(\tilde{K}_k^{\text{av}}, \tilde{\theta}_k^{\text{av}}) &\leq -\gamma\kappa\nu_1\left\|\Gamma(\tilde{K}_k^{\text{av}}+K_\star, \theta_\star)\right\|^2 + \gamma\kappa\beta_2\left\|\Gamma(\tilde{K}_k^{\text{av}}+K_\star, \theta_\star)\right\|\left\|\tilde{\theta}_k^{\text{av}}\right\| - \gamma\nu_2\left\|\tilde{\theta}_k^{\text{av}}\right\|^2 \\ &\overset{(a)}{=} -\gamma\begin{bmatrix}\left\|\Gamma(\tilde{K}_k^{\text{av}}+K_\star, \theta_\star)\right\| \\ \left\|\tilde{\theta}_k^{\text{av}}\right\|\end{bmatrix}^\top U(\kappa)\begin{bmatrix}\left\|\Gamma(\tilde{K}_k^{\text{av}}+K_\star, \theta_\star)\right\| \\ \left\|\tilde{\theta}_k^{\text{av}}\right\|\end{bmatrix},\end{aligned}\tag{2.52}$$

where in (a) we have simply rearranged the terms in a quadratic form with

$$U(\kappa) := \begin{bmatrix}\kappa\nu_1 & -\frac{\kappa\beta_2}{2} \\ -\frac{\kappa\beta_2}{2} & \nu_2\end{bmatrix}.$$

In light of the Sylvester criterion, the matrix $U(\kappa)$ is positive definite if and only if its determinant is positive. Therefore, to enforce the positive definiteness of $U(\kappa)$ we further restrict $\kappa \in (0, \bar{\kappa})$, with $\bar{\kappa} := \min\{\beta_2/(4\nu_1\nu_2), 1\}$. Let $\eta > 0$ be the smallest eigenvalue of $U(\kappa)$, then (2.52) can be bounded as

$$\begin{aligned}\Delta V(\tilde{K}_k^{\text{av}}, \tilde{\theta}_k^{\text{av}}) &\leq -\gamma\eta\left(\left\|\Gamma(\tilde{K}_k^{\text{av}} + K_\star, \theta_\star)\right\|^2 + \left\|\tilde{\theta}_k^{\text{av}}\right\|^2\right) \\ &\overset{(a)}{\leq} -\gamma\frac{\eta}{\mu}\Big(J(\tilde{K}_k^{\text{av}} + K_\star, \theta_\star) - J(K_\star, \theta_\star)\Big) - \gamma\eta\left\|\tilde{\theta}_k^{\text{av}}\right\|^2 \\ &\overset{(b)}{\leq} -\gamma\eta\min\left\{\tfrac{1}{\mu\kappa}, 1\right\}V(\tilde{\theta}_k^{\text{av}}, \tilde{K}_k^{\text{av}}),\end{aligned}\tag{2.53}$$

where (a) follows from the gradient dominance of J (cf. (2.46)), while (b) recovers the formulation of V (cf. (2.47)) by negleting a negative term. Being the right-hand side of (2.53) always non-positive, it holds

$$\begin{aligned}V(\tilde{K}_{k+1}^{\text{av}}, \tilde{\theta}_{k+1}^{\text{av}}) &\leq V(\tilde{K}_k^{\text{av}}, \tilde{\theta}_k^{\text{av}}) \\ &\overset{(a)}{\leq} J(\tilde{K}_k^{\text{av}}+K_\star, \theta_\star) - J(K_\star, \theta_\star) + \tfrac{1}{2}\left\|\tilde{\theta}_k^{\text{av}}\right\|^2,\end{aligned}\tag{2.54}$$

[7] Given any two vectors $v_1, v_2 \in \mathbb{R}^n$, it holds $\|v_1 - v_2\|^2 \leq 2\|v_1\|^2 + 2\|v_2\|^2$.

where (a) holds because $\kappa \leq 1$. In light of the definition of Ω_{c_0}, the inequality (2.54) guarantees that $(\tilde{K}^{\text{av}}_{k+1}, \tilde{\theta}^{\text{av}}_{k+1}) \in \Omega_{c_0}$ hence proving the invariance.

Since System (2.44) is initialized into Ω_{c_0}, then its trajectories satisfy (2.53). Hence, the exponential stability of the origin for System (2.44) is implied (cf. [109, Theorem 13.2]) and the proof follows. ■

Once this result has been posed, we can proceed with the proof of Theorem 2.1 in the next subsection.

Proof of Theorem 2.1

We will use Theorem C.1 given in Appendix C to guarantee the exponential stability of the origin for System (2.33). More specifically, in order to apply Theorem C.1, we need to verify

(i) the exponential stability of the origin for the associated averaged system,

(ii) the Lipschitz continuity of the vector field of (2.33),

(iii) that the origin is an equilibrium point of (2.33), and

(iv) that

$$\left\| \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} \Delta f(\tilde{z}, i) \right\| \leq \nu(T) \left\| \tilde{z} \right\|, \qquad \left\| \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} \frac{\partial \tilde{f}(\tilde{z}, i)}{\partial \tilde{z}} \right\| \leq \nu(T), \tag{2.55}$$

where $\Delta f(\tilde{z}, i) := f(0, \tilde{z}, i) - f^{\text{av}}(\tilde{z})$ and $\nu(k)$ is a nonnegative strictly decreasing function with the property $\nu(k) \to 0$ as $k \to \infty$.

As for condition (i), it follows from Proposition 2.1. Condition (ii) is satisfied by using the quantities defined in (2.48) as the required Lipschitz constants of the vector field of (2.33). Condition (iii) can be verified by means of (2.35). Finally, in order to check condition (iv) (cf. (2.55)), note that

$$\tilde{f}(\tilde{z}, k) = (H^{\text{ss}}_k)^{\dagger} H^{\text{ss}}_k \tilde{\theta} - \tilde{\theta} \overset{(a)}{=} 0, \tag{2.56}$$

where in (a) we use the fact that H^{ss}_k is invertible for all $k \in N$ (cf. Lemma 2.2). Therefore, the conditions in (2.55) are satisfied and, thus, we can apply Theorem C.1. This result guarantees the existence of $\bar{\gamma} > 0$ such that, for all $\gamma \in (0, \bar{\gamma})$, the origin is an exponentially stable equilibrium point for System (2.33). The proof follows backtracking to the original coordinates (x, θ, K).

2.2.3 Numerical Simulations

In this section, we provide some numerical simulations to corroborate our theoretical findings. We consider the linear model of a highly maneuverable aircraft derived from the linearization of its longitudinal dynamics at an altitude of 3000 [ft] and a velocity of 0.6 [Mach], see [130]. The resulting linear time-invariant dynamics in continuous-time reads as

$$\dot{x} = \begin{bmatrix} -0.0151 & -60.5651 & 0 & -32.174 \\ -0.0001 & -1.3411 & 0.9929 & 0 \\ 0.00018 & 43.2541 & -0.86939 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x + \begin{bmatrix} -2.516 & -13.136 \\ -0.1689 & -0.2514 \\ -17.251 & -1.5766 \\ 0 & 0 \end{bmatrix} u, \quad (2.57)$$

where the state $x \in \mathbb{R}^4$ represents the forward velocity, the attack angle, the pitch rate and the pitch angle, while the inputs $u \in \mathbb{R}^2$ are the elevator and flaperon angles. The discrete-time system matrices $A_\star$ and $B_\star$ of (2.1) are discretized from the continuous-time system (2.57) using a Zero Order Hold on the input with sampling time $T_s = 0.05$ [s]. Notice that the resulting matrix $A_\star$ has one eigenvalue outside the unit disk, i.e., it is not Schur. The cost matrices $Q \in \mathbb{R}^{4\times 4}$ and $R \in \mathbb{R}^{2\times 2}$ are randomly generated, while ensuring that $Q = Q^\top \geq 0$ and $R = R^\top > 0$. We set $\gamma = 10^{-4}$ and $\|w_0\| = 0.01$.

Exogenus System Design Procedure

Before providing the results of the numerical simulations, we propose a procedure tailored to design an exogenous system such that Assumption 2.2 is guaranteed. For the sake of completeness, we consider the general case with the state dimension being n and the input dimension being m. First, we set $q := n + 1$, $n_w = qm$, and $F :=$ blkdiag$(F_1, \ldots, F_q)$, where, for all $i \in \{1, \ldots, q\}$,

$$F_i := \begin{bmatrix} \cos(\omega_i) & \sin(\omega_i) \\ -\sin(\omega_i) & \cos(\omega_i) \end{bmatrix},$$

for some given $\omega_i > 0$ such that ω_i and ω_j are uncorrelated for each pair $i, j \in \{1, \ldots, q\}$ with $i \neq j$. By choosing an initial condition w_0 that satisfies

$$([w_0]_{2i+1})^2 + ([w_0]_{2i+2})^2 \neq 0, \quad 0 \leq i \leq q-1, \quad (2.58)$$

the chosen structure of F guarantees (2.14a) according to [171, Th.2]. As for (2.14b), we achieve it by selecting E such that $\begin{bmatrix} E^\top & (EF)^\top & \ldots & (EF^n)^\top \end{bmatrix}$ is nonsingular.

Aircraft Control

We start by considering the LTI system (2.57). We run Algorithm 1 with the exogenus signal generated via the procedure detailed above. In Figure 2.5 (left) it is possible to observe the evolution of the normalized cost error $|J(K_k, \theta_\star) - J^\star|/J^\star$, with $J^\star := J(K_\star, \theta_\star)$ and $\theta_\star := [A_\star \ \ B_\star]^\top$, in logarithmic scale. On the right of Figure 2.5 it is depicted the evolution of the normalized estimation error $\|\theta_k - \theta_\star\| \,/\, \|\theta_\star\|$ in logarithmic scale. Notice that, in both cases, convergence to the optimal cost $J^\star$ and true parameters $\theta_\star$ is achieved. Finally, in Figure 2.4 the state trajectory of the closed-loop system is depicted. The initial condition x_0 is sampled from a normal distribution with mean value 10 for each state. Notice that, after a transient, the states oscillate about the origin due to the exogenous system.

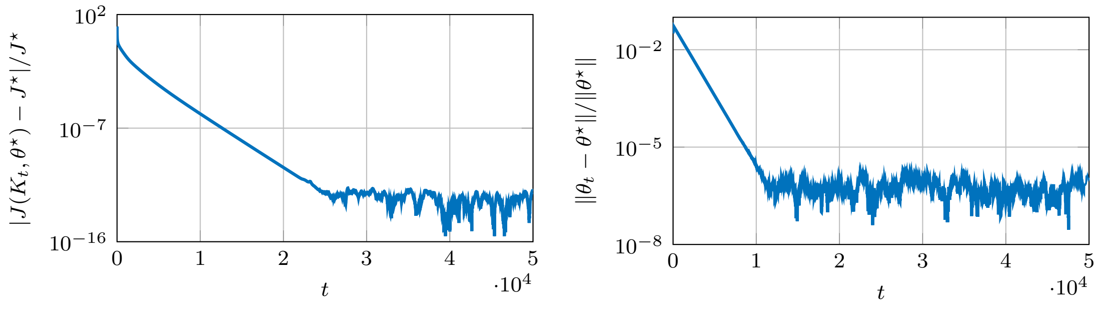


Figure 2.4: (left) Evolution of the normalized cost error $|J(K_k, \theta_\star) - J^\star|/J^\star$. (right) Evolution of the normalized estimation error about $\|\theta_k - \theta_\star\| \,/\, \|\theta_\star\|$ (left).

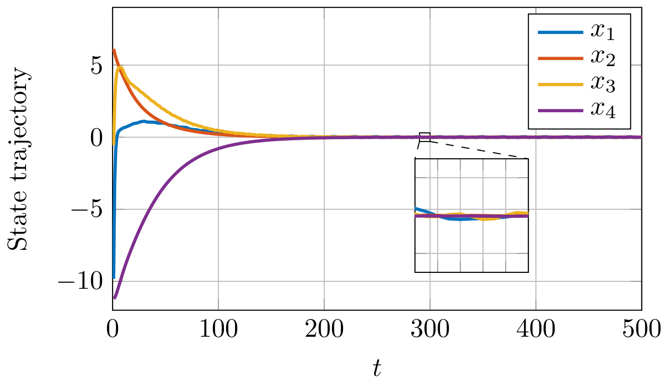


Figure 2.5: State trajectory of the closed-loop system. The states x_1, x_2, x_3, x_4 correspond, respectively, the forward velocity, the attack angle, the pitch rate and the pitch angle.

Aircraft Control with Drifting Parameters

To better highlight the capabilities of our algorithm, we also consider the case where the system matrices $A_\star$, $B_\star$, slowly change over time. The new time-varying state and input matrices are denoted as $A_\star^k$ and $B_\star^k$, respectively. More in detail, the time-varying system matrices $A_\star^k$ and $B_\star^k$ smoothly evolve from $A_\star$ and $B_\star$ toward a new pair of matrices A_+ and B_+, according to the update law

$$\begin{aligned} A_\star^k &= (1-\sigma(k))A_\star + \sigma(k)A_+ \\ B_\star^k &= (1-\sigma(k))B_\star + \sigma(k)B_+, \end{aligned}$$

for all $k \geq 0$, with $\sigma(k)$ being a sigmoid function defined as $\sigma(k) = 1/(1+\exp(\frac{k-t^{\text{trigger}}}{\alpha}))$, where $\alpha \in \mathbb{R}$ determines the transition width and $t^{\text{trigger}} \in \mathbb{N}$ defining the center of the transition. We select $t^{\text{trigger}} = 1.5 \cdot 10^5$ and $\alpha = 5 \cdot 10^3$, while the entries a_{ij}^+ and $b_{i\ell}^+$ of A_+ and B_+ are randomly generated according to

$$a_{ij}^+ = \begin{cases} a_{ij} & \text{if } a_{ij} = 0 \\ a_{ij} + \sigma v_{ij}^A & \text{otherwise} \end{cases}, \qquad b_{i\ell}^+ = \begin{cases} b_{i\ell} & \text{if } b_{ij} = 0 \\ b_{i\ell} + \sigma v_{i\ell}^B & \text{otherwise} \end{cases},$$

for all $i, j \in \{1, \ldots, n\}$ and $\ell \in \{1, \ldots, m\}$, where v_{ij}^A and $v_{i\ell}^B$ are random variables normally distributed and $\sigma = 0.1$ is the chosen variance. In Figure 2.6, we compare $J(K_k, \theta_\star)$ and $J_k^\star$. In Figure 2.7 (right), it is possible to observe the evolution of the normalized cost error $|J(K_k, \theta_\star) - J_k^\star|/J_k^\star$, with $J_k^\star := J(K_\star, \theta_\star)$ and $\theta_\star := \begin{bmatrix} A_\star^k & B_\star^k \end{bmatrix}^\top$, in logarithmic scale. Finally, in Figure 2.7 (left) it is depicted the evolution of the normalized estimation error $\|\theta_k - \theta_\star\| \,/\, \|\theta_\star\|$ in logarithmic scale. Notice that, in both cases, convergence to the optimal cost $J_k^\star$ and true parameters $\theta_\star$ is achieved. As one may expect, in the neighborhood of the inflection point $k \approx t^{\text{trigger}}$, both error quantities increase. However, we note that our policy shows its adaptability by quickly recovering convergence toward the optimal gain and exact estimation.

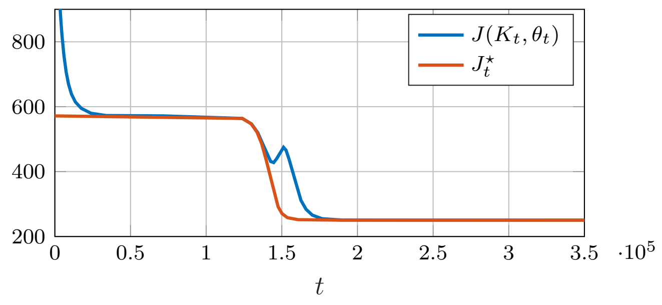


Figure 2.6: Comparison between $J(K_k, \theta_\star)$ and $J_k^\star$.

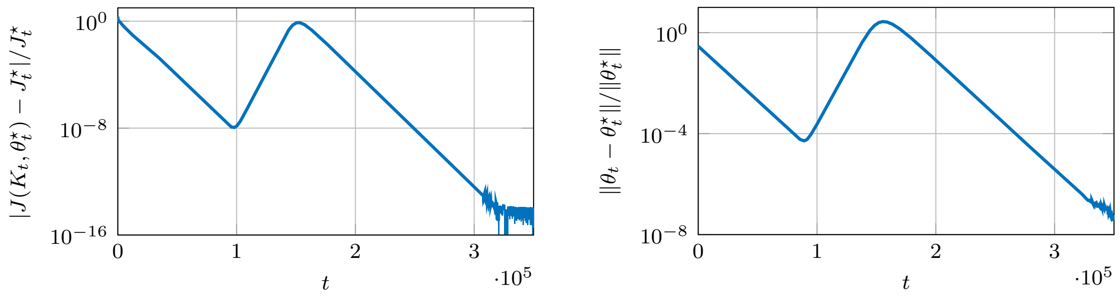


Figure 2.7: (Left) Evolution of the normalized cost error $|J(K_k, \theta_\star) - J_k^\star|/J_k^\star$. (right) Evolution of the normalized estimation error $\|\theta_k - \theta_\star\| \,/\, \|\theta_\star\|$ (left).

2.3 Structured Policy Design via Reinforcement Learning

In this section, we develop a Reinforcement Learning strategy for the design of structured feedback policies for large-scale networked systems using data. Specifically, we consider discrete-time linear systems of the form

$$x_{t+1} = A_\star x_t + B_\star u_t, \qquad x_0 = x_{\text{init}} \tag{2.59}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ denote the system's state and input at time $t \in \mathbb{N}$, while $A_\star \in \mathbb{R}^{n\times n}$ and $B_\star \in \mathbb{R}^{n\times m}$ are the system's matrices and $x_{\text{init}} \in \mathbb{R}^n$ the initial condition. We assume that the matrices $A_\star$ and $B_\star$ are *not known* and that the objective is to develop a linear state feedback of the form

$$u_t = K x_t, \tag{2.60}$$

where $K \in \mathbb{R}^{m\times n}$, such that the closed-loop system is asymptotically stable. Clearly, this is not directly possible since we do not know the system matrices. Furthermore, for the design of such controller, we aim at imposing a certain sparsity on K (that is, some entries must be zero) and to optimize an infinite-horizon quadratic cost. This can be formulated as the optimal control problem

$$\begin{aligned} \min_{x_t,u_t,K} \quad & \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t \\ \text{subj.to} \quad & x_{t+1} = A_\star x_t + B_\star u_t \\ & u_t = K x_t \\ & K \circ \mathcal{S} = 0 \end{aligned} \tag{2.61}$$

where $Q = Q^\top > 0 \in \mathbb{R}^{n\times n}$ and $R = R^\top > 0 \in \mathbb{R}^{m\times m}$ are the cost matrices associated to state and input, the symbol $>$ indicates that the matrices are positive definite and $\circ$

denotes item-by-item matrix multiplication. The binary matrix $\mathcal{S} \in \{0,1\}^{m\times n}$ models the desired sparsity in the sense that its (i,j)-th entry $\mathcal{S}_{(i,j)}$ is

$$\mathcal{S}_{(i,j)} := \begin{cases} 0, & \text{if } K_{(i,j)} \text{ is free;} \\ 1, & \text{otherwise.} \end{cases}$$

The challenge of solving problem (2.61) is twofold. We do not know the system dynamics, which hampers applicability of model-based optimal control methods. Moreover, even if this was the case, the design of a sparse K calls for different solution approaches. In order to tackle both problems, we will rely on a suitably designed Q-learning algorithm. Before describing the proposed approach, in the remainder of this section we show a motivating example and introduce the necessary preliminaries on Q-learning.

Remark 2.2 (Model-based non-sparse LQ solution)**.** If the sparsity constraint $K \circ \mathcal{S} = 0$ was not present in problem (2.61), the problem could be treated with LQ techniques [29]. In the model-based case, the solution to the problem is known to be a linear feedback $u_t = K_\star x_t$ with

$$K_\star = -(R + B_\star^\top P_\star B_\star)^{-1}(B_\star^\top P_\star A_\star), \tag{2.62}$$

where $P_\star$ is the (unique) positive definite solution of the Algebraic Riccati Equation associated to problem (2.61), i.e.,

$$P_\star = Q + A_\star^\top P_\star A_\star - A_\star^\top P_\star B_\star (R + B_\star^\top P_\star B_\star)^{-1} B_\star^\top P_\star A_\star.$$

The optimal gain matrix $K_\star$ cannot be guaranteed to be sparse. As such, in general $K_\star \circ \mathcal{S} \neq 0$. $\triangle$

Review of Reinforcement Learning and Q-learning in an LQ framework

Let us introduce the needed formalism for Q-learning techniques applied to LQ frameworks, see e.g. [29]. This technique is a Reinforcement Learning scheme for solving optimal control problems without any knowledge on the system dynamics. Formally, consider a linear system of the form (2.59) and consider a quadratic stage cost of being at a state x and applying an input u,

$$\ell(x,u) = x^\top Q x + u^\top R u.$$

The objective is to find an optimal input sequence $\mathbf{u}$ to minimize the infinite-horizon cost associated to an initial state x_k. To tackle this problem with Reinforcement Learning approaches, one typically considers an input *policy* $u_t = K x_t$, which is a linear state feedback, and defines the so-called *Value function* associated to the gain matrix K, which associates to the gain matrix K, its infinite horizon cost under dynamics (2.59). More specifically, assume to have a given gain matrix K, and initial state x_k. The state trajectory for all $t > k$, under the feedback control policy $u_t = Kx_t$, can be written as

$$x_t = (A_\star + B_\star K)^{t-k} x_k. \tag{2.63}$$

We can evaluate the cost of the control policy $u_t = Kx_t$ cost starting from x_k via the Value Function $\mathcal{V} : \mathbb{R}^n \times \mathbb{R}^{m\times n} \to \mathbb{R}$, defined as

$$\begin{aligned}
\mathcal{V}(x_k; K) &:= \sum_{t=k}^{\infty} x_t^\top (Q + K^\top R K) x_t && (2.64\text{a})\\
&\overset{(a)}{=} x_k^\top \left[\sum_{t=k}^{\infty} (A_\star + B_\star K)^{t-k,\top} (Q + K^\top R K)(A_\star + B_\star K)^{t-k} \right] x_k && (2.64\text{b})\\
&\overset{(b)}{=} x_k^\top \left[\sum_{\tau=0}^{\infty} (A_\star + B_\star K)^{\tau,\top} (Q + K^\top R K)(A_\star + B_\star K)^{\tau} \right] x_k && (2.64\text{c})
\end{aligned}$$

where in (a) we leveraged on (2.63), and in (b) we performed a change of indexes in the summation. If K is stabilizing, then $\sum_{\tau=0}^{\infty} (A_\star + B_\star K)^{\tau,\top} (Q + K^\top R K)(A_\star + B_\star K)^{\tau}$ is finite and equal to the solution P^K of the Lyapunov equation

$$(A_\star + B_\star K)^\top P^K (A_\star + B_\star K) - P^K + Q + K^\top R K = 0. \tag{2.65}$$

Hence, we can write the Value Function associated to an initial state x_k as

$$\mathcal{V}(x_k; K) = x_k^\top P^K x_k. \tag{2.66}$$

This function represents the cost obtained by starting system (2.59) at a certain initial state x_0 and applying the policy $u_t = Kx_t$.

Similarly, we can define the *Q-function* under policy K, denoted as $\mathcal{Q} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{n\times m} \to \mathbb{R}$, as the cost of starting from x_k, using an arbitrary input u_k at time k, and following $u_t = Kx_t$ from $t > k$ onward. The Q-function is defined as

$$\begin{aligned}
\mathcal{Q}(x_k, u_k; K) &= x_k^\top Q x_k + u_k^\top R u_k + \sum_{t=k+1}^{\infty} x_t^\top (Q + K^\top R K) x_t && (2.67\text{a})\\
&= x_k^\top Q x_k + u_k^\top R u_k\\
&\qquad + x_{k+1}^\top \left[\sum_{t=k+1}^{\infty} (A_\star + B_\star K)^{t-k-1,\top} (Q + K^\top R K)(A_\star + B_\star K)^{t-k-1} \right] x_{k+1}\\
&= x_k^\top Q x_k + u_k^\top R u_k
\end{aligned}$$

$$
\begin{aligned}
&+ x_{k+1}^\top \left[\sum_{\tau=0}^{\infty} (A_\star + B_\star K)^{\tau,\top} (Q + K^\top R K)(A_\star + B_\star K)^\tau \right] x_{k+1} \\
&= x_k^\top Q x_k + u_k^\top R u_k + x_{k+1}^\top P^K x_{k+1} \\
&= x_k^\top Q x_k + u_k^\top R u_k + (A_\star x_k + B_\star u_k)^\top P^K (A_\star x_k + B_\star u_k) \\
&= x_k^\top Q x_k + u_k^\top R u_k + \mathcal{V}(A_\star x_k + B_\star u_k; K) \qquad (2.67b)
\end{aligned}
$$

which can be written, more compactly, as

$$
\mathcal{Q}(x_k, u_k; K) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \underbrace{\begin{bmatrix} Q + A_\star^\top P^K A_\star & A_\star^\top P^K A_\star \\ \star & R + B_\star^\top P^K B_\star \end{bmatrix}}_{=:\Theta^K} \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \qquad (2.68)
$$

We highlight that the recursive relationship (2.67b) is known in literature as the *Bellman's equation* which holds for all $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ and relates the Value Function with the Q-function. Moreover, it also holds

$$
\mathcal{Q}(x, Kx; K) = \mathcal{V}(x; K).
$$

Building on this information, the generic Q-learning algorithm is obtained as a form of so-called policy iteration, which is an iterative scheme composed by two phases: *(i)* policy evaluation, *(ii)* policy improvement. Specifically, policy evaluation is performed exploiting an observed trajectory of the system in order to estimate the Q-function associated to a certain gain matrix K (more details on this step will be given next), while policy improvement is performed by finding a new gain K' minimizing the obtained cost. Since the dynamics is unknown, Q-learning assumes knowledge of a trajectory $\{x_t, u_t\}_{t=0}^T$ of length $T = \frac{1}{2}(n+m+1)(n+m)$ (cf. also [154]), where we recall that n and m are the dimension of state and input respectively. This trajectory may be obtained by either forward simulation of the system dynamics (2.59) or by direct observation on the real system. The following table summarizes the generic iteration of Q-learning.

Algorithm 2 An iteration of Q-learning

Input: gain matrix K, trajectory $\{x_t, u_t\}_{t=0}^{T}$

Policy evaluation: fit Q-function for given K from $\{x_t, u_t\}_{t=0}^{T}$

$$\mathcal{Q}(x,u;K) = \begin{bmatrix} x \\ u \end{bmatrix}^\top \begin{bmatrix} \Theta_{xx}^K & \Theta_{xu}^K \\ \Theta_{ux}^K & \Theta_{uu}^K \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}$$

Policy improvement: compute new feedback policy

$$\operatorname*{argmin}_u \mathcal{Q}(x,u;K) = \underbrace{-(\Theta_{uu}^K)^{-1}\Theta_{ux}^K}_{:=K'} x, \tag{2.69}$$

Output: new gain matrix K'

Note that nothing can be said on the sparsity of K'. Indeed, in general it does not hold $K' \circ \mathcal{S} = 0$. In the next section, we introduce a suitably designed Q-learning approach that guarantees sparsity of the computed policy.

2.3.1 LMI-based Design Strategy for Structured Q-learning: Algorithm Description and Analysis

In this section, we present our Structured Off-policy Q-learning scheme and provide a mathematical analysis.

Let us now introduce the proposed scheme for the data-driven derivation of structured feedback policies for LQR. The algorithm is inspired to the Q-learning scheme (Algorithm 2), where the policy improvement step is modified such that the newly computed policy obeys the desired sparsity pattern while granting recursive stability to the scheme. Let $k \in \mathbb{N}$ be an iteration index and let $K^k \in \mathbb{R}^{m\times n}$ be the feedback gain obtained at a certain iteration k of the algorithm. Initially, a system trajectory is collected and any initial stabilizing matrix K_0 with the desired sparsity is found. Then, the proposed algorithm follows an iterative procedure that repeats a policy evaluation step, consisting in computing Θ^k as the solution of a linear system of equations (2.70), with

$$\Theta^k = \begin{bmatrix} \Theta_{xx}^k & \Theta_{xu}^k \\ {\Theta_{xu}^k}^\top & \Theta_{uu}^k \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)},$$

and a policy improvement step, consisting in the solution of an LMI (2.71). The proposed Q-learning scheme is summarized in Algorithm 3.

Algorithm 3 Structured-policy Q-learning

```
Off-policy data gathering: collect $\{x_t, u_t\}_{t=0}^{T}$
Find any initial stabilizing gain $K_0$ with $K_0 \circ S = K_0$.
for $k = 0, 1, \dots$ do
  Policy evaluation: find $\Theta^k$ as solution to linear system
```

$$\begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top \Theta \begin{bmatrix} x_t \\ u_t \end{bmatrix} = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} x_{t+1} \\ K^k x_{t+1} \end{bmatrix}^\top \Theta \begin{bmatrix} x_{t+1} \\ K^k x_{t+1} \end{bmatrix} \quad \text{for } t = 0, \dots, T-1 \tag{2.70}$$

```
  Policy improvement: let $\Phi^k := 2\Theta^k_{xu} K^k + {K^k}^\top \Theta^k_{uu} K^k$, then find $K^{k+1}$ as the solution to LMI
```

$$\begin{aligned} &\text{find } K \\ &\text{subj.to } \begin{bmatrix} \Phi^k - 2\Theta^k_{xu} K & (\Theta^k_{uu} K)^\top \\ \Theta^k_{uu} K & \Theta^k_{uu} \end{bmatrix} \geq 0 && (2.71a) \\ &K \circ \mathcal{S} = 0 && (2.71b) \end{aligned}$$

As regards the computational load of the algorithm, at each iteration only a linear system of equations and an LMI must be solved, both of which are easily and quickly performed with open-source or commercial solvers. We once again stress that the proposed algorithm is an off-policy scheme, that is, the computed policies need not be implemented on the real system while they are being learned. This is useful in situations where it is not possible to actually run the system many times, and indeed our algorithm only requires offline observation of a single trajectory.

Next we state the theoretical results for Algorithm 3. To this end, we now introduce the needed assumptions. The first one is standard for LQR.

Assumption 2.3. *Pair $(A_\star, B_\star)$ is assumed to be controllable and $(A_\star, C)$ is assumed to be observable with C such that $Q = C^\top C$.* △

In order for the approach to work properly, the trajectory must satisfy a *persistence of excitation* property, which informally speaking allows for the identification of the modes of the system and ensures that the policy evaluation step can be performed correctly. This is formalized next.

Assumption 2.4. *The trajectory $\{x_t, u_t\}_{t=0}^{T}$, with $T = \frac{1}{2}(n+m+1)(n+m)$, is obtained with a Persistently Exciting (P.E.) input sequence $\{u_t\}_{t=0}^{T}$ of order $n+1$, namely is such that* (2.14b) *hold.* △

The third assumption regards stability and sparsity of the initial condition provided to the algorithm.

Assumption 2.5. *The initial feedback matrix K_0 is stabilizing and satisfies $K_0 \circ \mathcal{S} = 0$.* $\triangle$

With these assumptions in place, we can state the main theoretical result.

Theorem 2.2. *Let Assumptions 2.3, 2.4 and 2.5 hold. Consider the sequence of control policies $\{K^k\}_{k \geq 0}$ generated by Algorithm 3. Then, for all $k \in \mathbb{N}$, the matrix $A_\star + B_\star K^k$ is Schur and it holds $K^k \circ \mathcal{S} = 0$. Moreover, every limit point $\bar{K}$ of $\{K^k\}_{k \geq 0}$ is such that $A_\star + B_\star \bar{K}$ is Schur and $\bar{K} \circ \mathcal{S} = 0$.* $\triangle$

Before giving the proof of Theorem 2.2 we state some preparatory results. From now on, we use $\mathcal{Q}^k(x,u) = \mathcal{Q}(x,u;K^k)$ as a shorthand to denote the Q-function associated to the policy $u = K^k x$ of iteration k.

The following two lemmas ensure that the algorithm is well posed if an initial sparse stabilizing gain is known and formalize the properties of policy evaluation and policy improvement.

Lemma 2.4 (Policy evaluation)**.** *Let Assumptions 2.3 and 2.4 hold and assume that at a certain iteration k the gain K^k is stable, i.e., $A + BK^k$ is Schur. Then, the system of equations* (2.70) *admits a unique solution Θ^k and the Q-function associated to the gain K^k is equal to*

$$\mathcal{Q}^k(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}^\top \underbrace{\begin{bmatrix} \Theta^k_{xx} & \Theta^k_{xu} \\ {\Theta^k_{xu}}^\top & \Theta^k_{uu} \end{bmatrix}}_{\Theta^k} \begin{bmatrix} x \\ u \end{bmatrix}. \tag{2.72}$$

Proof. A proof of the existence and uniqueness of the solution to the system of equations (2.70) can be found in [154, Theorem 2]. As regards the second point, we proceed as follows. Recall from Section 2.3 the Bellman equation for the Q-function (2.67b) associated to the gain K^k,

$$\mathcal{Q}^k(x,u) = x^\top Q x + u^\top R u + \mathcal{Q}^k(Ax + Bu, K^k(A_\star x + B_\star u)) \ \ \forall x, u.$$

Using the fact that the Q-function is quadratic (2.68), it follows that there exist matrices Θ^k_{xx}, Θ^k_{xu} and Θ^k_{uu} such that

$$\begin{aligned} &\begin{bmatrix} x \\ u \end{bmatrix}^\top \begin{bmatrix} \Theta^k_{xx} & \Theta^k_{xu} \\ {\Theta^k_{xu}}^\top & \Theta^k_{uu} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \\ &\quad = \begin{bmatrix} x \\ u \end{bmatrix}^\top \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} A_\star x + B_\star u \\ K^k(A_\star x + B_\star u) \end{bmatrix}^\top \begin{bmatrix} \Theta^k_{xx} & \Theta^k_{xu} \\ {\Theta^k_{xu}}^\top & \Theta^k_{uu} \end{bmatrix} \begin{bmatrix} A_\star x + B_\star u \\ K^k(A_\star x + B_\star u) \end{bmatrix} \end{aligned}$$

for all x, u, with In particular, choosing in the previous equation x, u along the trajectory obtained with the P.E. input, we obtain exactly the system of equation (2.70), which shows that indeed the solution of the system is

$$\Theta^k = \begin{bmatrix} \Theta^k_{xx} & \Theta^k_{xu} \\ {\Theta^k_{xu}}^\top & \Theta^k_{uu} \end{bmatrix},$$

and thus the Q-function associated to K^k is (2.72). ■

Lemma 2.5 (Policy improvement)**.** *Let Assumption 2.3 hold and assume that at a certain iteration k the (sparse) gain K^k is stable, i.e., $A+BK^k$ is Schur. Then, the policy improvement step* (2.71) *is feasible and produces a new gain K^{k+1} such that*

$$\mathcal{Q}^k(x, K^{k+1}x) \leq \mathcal{Q}^k(x, K^k x), \qquad \forall x \in \mathbb{R}^n, \tag{2.73}$$

and satisfying the sparsity pattern $K^{k+1} \circ \mathcal{S} = 0$. △

Proof. Let K^{k+1} denote the solution of the policy improvement step (2.71). We will later show that there always exists at least one such solution. By construction, K^{k+1} satisfies the desired sparsity pattern, i.e., $K^{k+1} \circ S = K^{k+1}$. Moreover, it holds

$$\begin{bmatrix} \Phi^k - 2\Theta_{xu}K^{k+1} & (\Theta^k_{uu}K^{k+1})^\top \\ (\Theta^k_{uu}K^{k+1}) & \Theta^k_{uu} \end{bmatrix} \geq 0 \tag{2.74}$$

with $\Phi^k = 2\Theta^k_{xu}K^k + {K^k}^\top \Theta^k_{uu}K^k$. By Schur complement (see, e.g., [42]), matrix inequality (2.74) is equivalent to

$$\Theta^k_{uu} \geq 0 \tag{2.75a}$$
$$(\Theta^k_{uu}K^{k+1})^\top (I - {\Theta^k_{uu}}^{-1}\Theta^k_{uu}) = 0 \tag{2.75b}$$
$$\Phi^k - 2\Theta^k_{xu}K^{k+1} - (\Theta^k_{uu}K^{k+1})^\top {\Theta^k_{uu}}^{-1}(\Theta^k_{uu}K^{k+1}) \geq 0 \tag{2.75c}$$

In particular, condition (2.75c) can be rewritten as

$$\Phi^k - 2\Theta^k_{xu}K^{k+1} - {K^{k+1}}^\top \Theta^k_{uu}K^{k+1} \geq 0 \tag{2.76}$$

By adding and subtracting Θ^k_{xx} and changing the sign, we can write

$$\Theta^k_{xx} + 2\Theta^k_{xu}K^{k+1} + {K^{k+1}}^\top \Theta^k_{uu}K^{k+1} - \Theta^k_{xx} - 2\Theta^k_{xu}K^k - {K^k}^\top \Theta^k_{uu}K^k \leq 0 \tag{2.77}$$

where the term Φ^k has been expanded.

As a consequence, the quadratic form associated to the matrix in the left-hand side of (2.77) satisfies

$$\begin{aligned}&x^\top \Theta_{xx}^k x + x^\top 2\Theta_{xu}^k K^{k+1}x + x^\top {K^{k+1}}^\top \Theta_{uu}^k K^{k+1}x\\ &\quad - x^\top \Theta_{xx}^k x - 2x^\top \Theta_{xu}^k K^k x^\top - x^\top {K^k}^\top \Theta_{uu}^k K^k x^\top \le 0,\end{aligned}$$

for all x. Using (2.72), this is equivalent to

$$\mathcal{Q}^k(x, K^{k+1}x) \le \mathcal{Q}^k(x, K^k x), \qquad \forall x \in \mathbb{R}^n. \tag{2.78}$$

As regards solvability of the policy improvement step (2.71), note that there always exists at least a sparse matrix K satisfying (2.78), namely $K^{k+1} = K^k$. Thus all the steps in the proof can be repeated backward to conclude that K^k is a feasible solution to (2.71). ■

As a consequence of the previous lemma, we have the following fundamental result, which formalizes the non-increasing infinite-horizon cost property of our algorithm.

Proposition 2.2. *Consider the same assumptions of Lemma 2.5 and consider the gain K^k of a certain iteration k and the updated gain K^{k+1}. Then, the Q-functions associated with these two feedbacks satisfy*

$$\mathcal{Q}^{k+1}(x,u) \le \mathcal{Q}^k(x,u), \qquad \forall x, u. \tag{2.79}$$

Proof. We begin by noting a basic property. Indeed, combining the definition of Q-function in (2.67a) with (2.73) we see that

$$\begin{aligned}\mathcal{Q}^k(x_0, K^{k+1}x_0) &= x_0^\top (Q + {K^{k+1}}^\top R K^{k+1})x_0 + \sum_{t=1}^{\infty} x_t^\top (Q + {K^k}^\top R K^k) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^k)x_t}\\ &\le \mathcal{Q}^{K^k}(x_0, K^k x_0)\\ &= \sum_{t=0}^{\infty} x_t^\top (Q + {K^k}^\top R K^k) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^k)x_t}\end{aligned} \tag{2.80}$$

To prove the proposition, one notes that the summation in the left-hand side of (2.80) is equal to $Q^k(x_1, K^k x_1)$ for a suitable x_1 and applies recursively the inequality. Namely

$$\begin{aligned}\mathcal{Q}^k(x_0, u) &= x_0^\top Q x_0 + u^\top R u + \sum_{t=1}^{\infty} x_t^\top (Q + {K^k}^\top R K^k) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^k)x_t}\\ &\overset{(2.80)}{\ge} x_0^\top Q x_0 + u^\top R u + x_1^\top (Q + {K^{k+1}}^\top R K^{k+1}) x_1\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{t=2}^{\infty} x_t^\top (Q + K^{k\top} R K^k) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^k)x_t} \\
&\overset{(2.80)}{\geq} x_0^\top Q x_0 + u^\top R u + \sum_{t=1}^{2} x_t^\top (Q + K^{k+1\top} R K^{k+1}) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^{k+1})x_t} \\
&+ \sum_{t=3}^{\infty} x_t^\top (Q + K^{k\top} R K^k) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^k)x_t}
\end{aligned}
$$

and so on. In the limit, we obtain for all x_0 and u

$$
\begin{aligned}
\mathcal{Q}^k(x_0, u) &\geq x_0^\top Q x_0 + u^\top R u + \sum_{t=1}^{\infty} x_t^\top (Q + K^{k+1\top} R K^{k+1}) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^{k+1})x_t} \\
&= \mathcal{Q}^{k+1}(x_0, u).
\end{aligned}
$$

The proof follows. ■

We are now ready to prove Theorem 2.2.

Proof of Theorem 2.2

Since K_0 is stabilizing by assumption, using Lemma 2.4 the system (2.70) is well posed and the Q-function at iteration k can be written as (2.72). Lemma 2.5 ensures that the policy improvement step (2.71) is well posed and produces a new gain K^1 satisfying the sparsity constraint, i.e., $K^1 \circ \mathcal{S} = 0$. Moreover, by Proposition 2.2, it follows that

$$
\mathcal{Q}^1(x, u) \leq \mathcal{Q}^0(x, u), \qquad \forall x, u. \tag{2.81}
$$

Expanding the previous inequality using the definition of Q-function, we obtain

$$
\begin{aligned}
&\sum_{t=0}^{\infty} x_t^\top (Q + K^{1\top} R K^1) x_t \Big|_{x_{t+1}=(A_\star + B_\star K^1)x_t} \\
&\quad \leq \sum_{t=0}^{\infty} x_t^\top (Q + K_0^\top R K_0) x_t \Big|_{x_{t+1}=(A_\star + B_\star K_0)x_t} < +\infty,
\end{aligned} \tag{2.82}
$$

where in the last inequality we used the fact that K_0 is stabilizing and that x_t goes to zero exponentially (and thus the infinite-horizon cost is finite). Thus, the generic term of the series in the left-hand side goes to zero:

$$
\lim_{t\to\infty} x_t^\top (Q + K^{1\top} R K^1) x_t = 0. \tag{2.83}
$$

Since $Q + K^{1\top} R K^1$ is positive definite, then $\lim_{t\to\infty} x_t = 0$, that is, K^1 is stabilizing.

Repeating the same arguments by induction, we conclude that K^k is stable, i.e., $A+BK^k$ is Schur, and sparse, i.e, $K^k \circ \mathcal{S} = 0$, for all $k \geq 0$. To conclude, consider a limit point $\bar{K}$ (if it exists) and consider a subsequence of indices $\{k_n\}_{n\geq 0}$ such that $\lim_{n\to\infty} K^{k_n} = \bar{K}$.

Since each K^k is sparse, then $K^{k_n} \circ \mathcal{S} = 0$ for all n, and by taking the limit as $n \to \infty$ we obtain $\bar{K} \circ \mathcal{S} = 0$. Moreover, by iterating the reasoning as in (2.82) for all k, it follows that the infinite-horizon cost corresponding to $\bar{K}$ is finite, and therefore $\bar{K}$ is stabilizing, i.e., $A + B\bar{K}$ is Schur. $\triangle$

Remark 2.3. To guarantee existence of limit points of $\{K^k\}_{k\geq 0}$, a sufficient condition is to ensure boundedness of the sequence. This can be done by adding a constraint of the type $-M \leq K_{ij} \leq M$ for all i, j to LMI (2.71), with $M > 0$ sufficiently large.

2.3.2 Numerical Simulations

In this section we present computations on a distributed control scenario. Let us consider a network of $N = 10$ agents with coupled dynamics,

$$x_{i,t+1} = A_{ii}x_{i,t} + \sum_{j\in\mathcal{N}_i} A_{ij}x_{j,t} + B_i u_{i,t}, \quad i = 1, \dots, N.$$

The state and input matrices are $A_{ii} = \begin{bmatrix} 1.1 & 0.1 \\ 0.1 & -1.1 \end{bmatrix}$, $A_{ij} = \begin{bmatrix} 0.5 & 0 \\ 0 & -0.5 \end{bmatrix}$, and $B_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, thus, $n_i = 2$, $m_i = 2$ and $n = 20$, $m = 20$. Interaction among the systems occurs according to a randomly generated Erdős-Rényi graph with edge probability $p = 0.05$. The obtained system is unstable, indeed the maximum eigenvalue of the overall system matrix A is equal to 2. We consider a quadratic stage cost with a sparsity matching the graph, i.e., $\ell(x, u) = \sum_{i=1}^{N} \ell_i(x, u)$ with

$$\ell_i(x, u) = x_i^\top \left(Q_i x_i + \sum_{j\in\mathcal{N}_i} Q_{ij}x_j \right) + u_i^\top R_i u_i,$$

for $i = 1, \dots, N$, where $Q_i = 5 \cdot I_2$, $R_i = 0.5 \cdot I_2$ and $Q_{ij} = I_2$, for all $j \in \mathcal{N}_i$. This can be easily recast as $\ell(x, u) = x^\top Qx + u^\top Ru$ with appropriate matrices $Q \in^{20\times 20}$ and $R \in^{20\times 20}$. We next show simulation results of running our Algorithm 3 in two different cases, namely *(i)* without imposing sparsity on K, and *(ii)* by imposing a sparsity matching the interaction graph.

Results Without Sparsity Constraints

First of all, we show that Algorithm 3, in absence of sparsity constraints, is capable of computing the optimal $K_\star$, solution of the infinite-horizon LQ problem. We first compute $K_\star$ by using the system model as described in Remark 2.2 and then run Algorithm 3 without sparsity constraints, i.e. by imposing S as matrix of ones. The starting matrix K_0 is initialized randomly upon checking that it stabilizes the system. The offline data gathering is performed with a random sequence of inputs with full-rank Hankel matrix (cf. Assumption 2.4).

In Figure 2.8 (left) we plot the error (in norm) between $K_\star$ and the K^k computed along the algorithmic evolution. Then we consider the infinite-horizon cost error of each policy $u = K^k x$. Specifically, we sample random initial state x_0 and we consider $\mathcal{V}^\star(x_0) = \mathcal{V}(x_0; K_\star)$, the infinite-horizon cost associated to $K_\star$, and $\mathcal{V}^k(x_0) = \mathcal{V}(x_0; K^k)$, the infinite-horizon cost associated to K^k. In Figure 2.8 (right) we plot the error between these two quantities as the algorithm evolves. The two graphs highlight that, when no sparsity is imposed on K, the algorithm converges to the optimal $K_\star$ and reaches a minimal infinite-horizon cost.

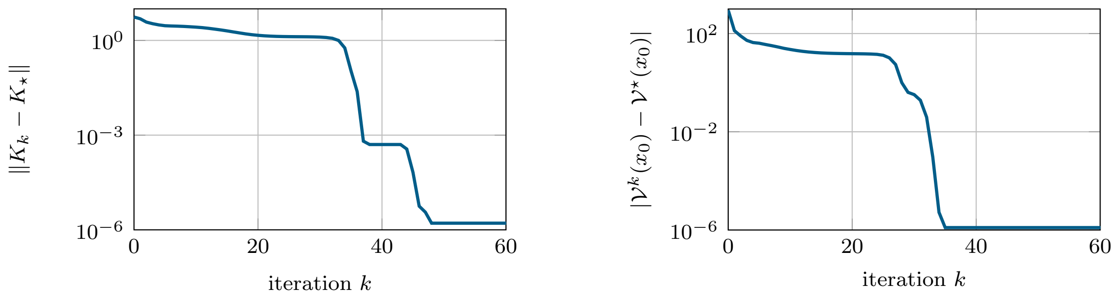


Figure 2.8: Results for non-sparse case. Left: norm of the difference between gain K^k computed by Algorithm 3 at iteration k, and the optimal gain $K_\star$. Right: infinite-horizon cost error of policies $u = K^k x$ computed by Algorithm 3 with respect to optimal policy.

Results With Sparsity Constraints

Now we turn our attention to the design of a sparse distributed controller of the form (2.60). We run the algorithm with the same settings as before, with the only difference that now we impose the sparsity constraint $K \circ \mathcal{S} = 0$. Namely, we impose $K \circ S$, where the ij-th element of $\mathcal{S} \in^{20\times 20}$.

$$\mathcal{S}_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{if } (i,j) \notin \mathcal{E}. \end{cases}$$

The sequence of gains $\{K^k\}_{k\geq 0}$ converges to a limit matrix $\bar{K}$, which has a structure matching the interaction graph among the systems, as represented in Figure 2.9.

Also in this case, we consider the infinite-horizon cost error of the gains K^k computed by the algorithm, with the same x_0 as before. In Figure 2.10, we plot the error between $V(x_0; K_\star)$ and $\mathcal{V}(x_0; K_\star)$. A comparison with Figure 2.8 (right) reveals that, although now there are sparsity constraints to be respected, the infinite-horizon cost error is not particularly large if compared to the non-sparse case.

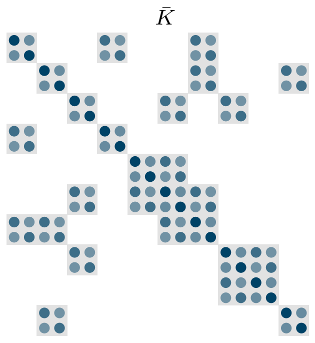


Figure 2.9: Representation of the sparse structure of $\bar{K}$. Blue dots represent the elements of $\bar{K}$, with color intensity proportional to their magnitude. The gray squares represent the desired structure defined by $\mathcal{S}$.

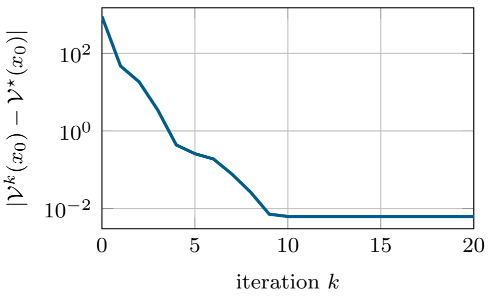


Figure 2.10: Infinite-horizon cost error of sparse policies $u = K^k x$ computed by Algorithm 3 with respect to optimal policy.

2.4 Structured Policy Design via Data-driven Lagrangian Methods

We consider discrete-time, linear and time-invariant systems in the form

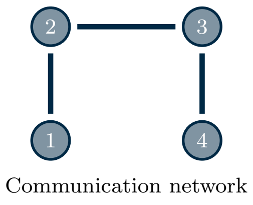
$$x_{t+1} = A_\star x_t + B_\star u_t, \quad x_0 \sim p_{\mathrm{x}_0} \tag{2.84}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the state and the input at time $t \in \mathbb{N}$, while $A_\star \in \mathbb{R}^{n\times n}$ and $B_\star \in \mathbb{R}^{n\times m}$ are the state and input matrices, respectively. The initial condition x_0 is assumed to be randomly distributed according to a known uniform probability distribution p_{x_0}. As formally stated in the following assumption, we assume the dynamics given in (2.84) to be unknown.

Assumption 2.6. *The system matrix $A_\star$ is unknown.* △

The control objective is to design, for the system (2.84), a static feedback policy in the form

$$u_t = K x_t \tag{2.85}$$



$$\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}$$

$\mathcal{S}$

Figure 2.11: Illustrative example of structured feedback matrix. In this case, the goal is to generate a gain matrix matching the communication structure of a network of agents. In this case, each agent $i = 1, \dots, 4$, has dynamics with two dimensional state and scalar input.

where the gain matrix $K \in \mathbb{R}^{m \times n}$ must

(i) stabilize the closed-loop system, i.e., $A_\star + B_\star K$ must have all the eigenvalues inside the open unit disk;

(ii) minimize a quadratic performance index;

(iii) match a given sparse structure, that is, having some prescribed entries equal to zero.

To enforce the (iii), let us introduce a matrix $\mathcal{S}^c \in \mathbb{R}^{m \times n}$ whose (i,j)-th entry $\mathcal{S}^c_{(i,j)}$ is

$$\mathcal{S}^c_{(i,j)} := \begin{cases} 1, & \text{if } K_{(i,j)} \text{ is free;} \\ 0, & \text{otherwise.} \end{cases}$$

Then, defining its logical complement

$$\mathcal{S} := \mathbf{1}_{m \times n} - \mathcal{S}^c, \tag{2.86}$$

the desired sparsity of the gain matrix K can be compactly imposed via the matrix constraint

$$K \circ \mathcal{S} = 0_{m \times n}.$$

Figure 2.11 schematically represents a distributed control scenario, where the feedback K is designed to match a graph structure representing the coupling among the agent dynamics.

Formally, the control requirements described so far can be posed in terms of an optimal control problem. Specifically, we want to design a feedback gain K, under Assumption 2.6, such that the state-input trajectory under the closed-loop policy (2.85) is a solution of the following *constrained* infinite-horizon LQR problem

$$\min_{\substack{x_1, x_2, \dots \\ u_0, u_1, \dots, K}} \quad \mathbb{E}\Big[\sum_{t=0}^{\infty} \Big(x_t^\top Q x_t + u_t^\top R u_t \Big) \Big] \tag{2.87a}$$

$$\text{subj.to } x_{t+1} = A_\star x_t + B_\star u_t, \qquad x_0 \sim p_{\mathrm{x}_0} \tag{2.87b}$$

$$u_t = K x_t \tag{2.87c}$$

$$K \circ \mathcal{S} = 0 \tag{2.87d}$$

where the linear constraint in (2.87d) is meant to enforce a structure on the gain K while $\mathbb{E}[\cdot]$ denotes the expectation operator over the random distribution p_{x_0}. The cost matrices are $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are such that $Q = Q^\top \geq 0$ and $R = R^\top > 0$. Moreover, the pair $(A_\star, B_\star)$ is controllable and the pair $(A_\star, Q^{1/2})$ is observable. To overcome the lack of knowledge imposed by Assumption 2.6, the controller must be synthesized based on data, i.e., system trajectories collected during experimental sessions.

Remark 2.4. Notice that, differently from the standard LQR framework, the problem formulation (2.87) explicitly considers the feedback matrix K as a decision variable in order to enforce the desired policy structure. △

Remark 2.5. Notice that, without the sparsity requirement imposed by (2.87d), the conditions $(A_\star, B_\star)$ controllable and $(A_\star, Q^{1/2})$ observable are sufficient to guarantee the existence of the (unique) optimal gain $K^\star$ (see, e.g., [29]). △

In the following, we assume that the desired sparse structure $\mathcal{S}$ ensures the existence of (at least one) stabilizing gain $K^\star$, solution of problem (2.87).

Remark 2.6. The design of a procedure to ensure the feasibility of (2.87), i.e., if there exists a stabilizing feedback K for dynamics (2.84) such that $K \circ \mathcal{S} =$, is it still under investigation. Also in the full knowledge case, i.e., when (A, B) are known, assessing the existence of a sparse static feedback K is an NP-hard problem, cf. [131]. △

In the considered data-driven scenario, we have access to T-long sequences of inputs, states and successor states

$$U := [u_0, u_1, \ldots, u_{T-1}] \in \mathbb{R}^{m\times T} \tag{2.88a}$$

$$X := [x_0, x_1, \ldots, x_{T-1}] \in \mathbb{R}^{n\times T} \tag{2.88b}$$

$$X^+ := [x_1, x_2, \ldots, x_T] \in \mathbb{R}^{n\times T}, \tag{2.88c}$$

with $T > 0$. Each tuple $(u_\tau, x_\tau, x_{\tau+1})$, with u_τ, x_τ being the τ-th column of U, X, respectively, and $x_{\tau+1}$, being the $(\tau+1)$-th column of X^+, satisfies the dynamics constraint (2.87b) for all $\tau = 0, \ldots, T-1$. Hence, it can be shown that (U, X, X^+) in (2.88) satisfy

$$X^+ = \begin{bmatrix} A_\star & B_\star \end{bmatrix} \begin{bmatrix} X \\ U \end{bmatrix}. \tag{2.89}$$

Then, we require an identifiability condition that data must meet for the learning objectives of this work. Formally, we consider the following assumption.

Assumption 2.7. *The state-input data are sufficiently informative in the sense that*

$$\operatorname{rank}\begin{bmatrix} X \\ U \end{bmatrix} = n + m, \tag{2.90}$$

namely, the matrix $[X^\top \; U^\top]^\top$ *is full-row rank.* $\triangle$

Remark 2.7. Condition (2.90) can be ensured by leveraging on Willems' lemma (cf. [238]), i.e., by choosing an input signal u_t persistently exciting of a sufficiently high order. $\triangle$

The subspace relations (2.89) and (2.90) can be exploited to parametrize the optimization problem (2.87) by data matrices (cf. [70]). Specifically, we are interested in a data-based representation of the *closed-loop* dynamics $(A_\star + B_\star K)$. Indeed, for any K, the rank condition (2.90) ensures the existence, by the Rouché-Capelli theorem, of a matrix $G \in \mathbb{R}^{T\times n}$ satisfying

$$\begin{bmatrix} I \\ K \end{bmatrix} = \begin{bmatrix} X \\ U \end{bmatrix} G. \tag{2.91}$$

Thus, defining the closed-loop matrix

$$A_{\text{cl}} := A_\star + B_\star K \tag{2.92}$$

and, leveranging on (2.89) and (2.91), we can compute an equivalent data-based representation of the closed-loop dynamics A_{cl} as

$$A_{\text{cl}} = X^+ G. \tag{2.93}$$

Such data-based parametrization can be then exploited to replace the closed-loop matrix $(A_\star + B_\star K)$ leading to a *direct formulation* of the LQR problem (see, e.g., [70]), i.e., by leveraging on the closed-loop matrix representation given in (2.93) the optimal solution of the unconstrained optimal control problem is computed exploiting the collected data.

Remark 2.8. Notice that for $T \geq n + m$, for any given K, the set $\mathcal{G}$ of parametrizing matrices G is given by

$$\mathcal{G} := \begin{bmatrix} X \\ U \end{bmatrix}^\dagger \begin{bmatrix} I \\ K \end{bmatrix} + \ker \begin{bmatrix} X \\ U \end{bmatrix}. \tag{2.94}$$

Namely, the nullspace of $[X^\top \; U^\top]^\top$ is nontrivial. $\triangle$

2.4.1 Data-driven Lagrangian Algorithm for Sparse Control Design

In this section, we formally introduce Data-driven AL Algorithm for Sparse Feedback Design, consisting of a bi-level optimization algorithm to design a sparse, stabilizing gain matrix K from data. We employ a data-driven approach to deal with the unknown dynamics and we use an augmented Lagrangian method to handle the sparsity constraints inspired by the approaches proposed in the continuous-time model-based framework in [149, 151].

First of all, we recall the *reduced cost* formulation of Problem (2.87) (see also Appendix B.1) with the additional constraint represented by the desired sparse structure to the gain K. The reduced problem reads as

$$\min_{K\in\mathcal{K}} \; J(K) \tag{2.95a}$$

$$\text{subj.to} \;\; K \circ \mathcal{S} = 0 \tag{2.95b}$$

with $\mathcal{S}$ as in (2.86) and $J(\cdot)$ defined as

$$J(K) := \text{Tr}\left[\sum_{t=0}^{\infty}(A_\star + B_\star K)^{t\top}(Q + K^\top R K)(A_\star + B_\star K)^t\right].$$

The leading idea is to address the constrained optimization problem (2.95) using the augmented Lagrangian (AL) approach. Specifically, the AL of (2.95) is given by

$$\mathcal{L}_c(K,\Lambda) = J(K) + \text{Tr}\left[\Lambda^\top K \circ \mathcal{S}\right] + \frac{c}{2}\|K \circ \mathcal{S}\|_F^2, \tag{2.96}$$

where $c > 0$ is a penalty parameter while $\Lambda \in \mathbb{R}^{m\times n}$ collects the Lagrange multipliers associated to the constraint in (2.95). The augmented Lagrangian method (see, e.g., [30, Section 5.2]) applied to (2.95) results in an iterative procedure in which at each iteration $\ell \in \mathbb{N}$ the following steps are performed

$$K^{\ell+1} = \underset{K}{\text{argmin}} \; \mathcal{L}_{c^\ell}(K, \Lambda^\ell) \tag{2.97a}$$

$$\Lambda^{\ell+1} = \Lambda^\ell + c^\ell(K^{\ell+1} \circ \mathcal{S}) \tag{2.97b}$$

$$c^{\ell+1} = \beta c^\ell \tag{2.97c}$$

with $c^\ell > 0$ being the penalty parameter, and $\beta > 0$ the scaling factor. Notice that a routine tailored to matrix variables, e.g., based on the gradient method, is needed to solve (2.97a).

In Algorithm 4, we present the proposed data-driven method based on the augmented Lagrangian approach solving problem (2.87). Starting from an initial possibly non sparse gain K_0 and multiplier Λ^0, at each iteration $\ell \in \mathbb{N}$ the following steps occur.

An updated gain matrix $K^{\ell+1}$, minimizer of (2.97a), is computed by executing a data-driven routine with steps indexed by $i \in \mathbb{N}$. Specifically, it consists of a *policy evaluation* step (2.98), followed by a gradient-based *policy improvement* step (2.99). Finally, the multiplier matrix $\Lambda^{\ell+1}$ is updated according to the ascent step (2.100a), and the penalty term $c^{\ell+1}$ is increased in (2.100b).

Algorithm 4 Data-driven AL Algorithm for Sparse Feedback Design

```
Require: Initial guess $K_0$, $\Lambda^0 = 0$ and system data $X^+, X, U$, $c_0$, $\beta$ a given threshold
  $\epsilon > 0$
  for $\ell = 0, 1, 2 \dots$ do
    Set $K_{\text{in}}^0 = K^\ell$
    for $i = 0, 1, 2 \dots$ do
      Policy Evaluation
      Compute $G^i$ such that
          $\begin{bmatrix} I \\ K_{\text{in}}^i \end{bmatrix} = \begin{bmatrix} X \\ U \end{bmatrix} G^i$          (2.98a)
      Compute $P^i$ solution of
          $G^{i,\top} X^{+\top} P X^+ G^i - P + Q + K_{\text{in}}^{i,\top} R K_{\text{in}}^i = 0$          (2.98b)
      Compute $W^i$ solution of
          $X^+ G^i W G^{i,\top} X^{+\top} - W + I = 0$          (2.98c)
      Policy Improvement
      Compute the descent direction $\Delta K^i$ as
          $\Delta K^i = -(R K_{\text{in}}^i + B_\star^\top P^i X^+ G^i) W^i - \Lambda^\ell - c^\ell (K_{\text{in}}^i \circ \mathcal{S})$          (2.99a)
      Update the gain matrix as
          $K_{\text{in}}^{i+1} = K_{\text{in}}^i + \gamma^i \Delta K^i$          (2.99b)
      Break if $\|\Delta K^i\| \leq \epsilon^\ell$
    Set $K^{\ell+1} = K_{\text{in}}^{i+1}$
    Update the multiplier and the penalty parameter as
          $\Lambda^{\ell+1} = \Lambda^\ell + c^\ell (K^{\ell+1} \circ \mathcal{S})$          (2.100a)
          $c^{\ell+1} = \beta c^\ell$          (2.100b)
```

At each $i \in \mathbb{N}$, the evaluation of K_{in}^i (2.99), consists in the resolution of two data-driven Lyapunov, equations (2.98b) and (2.98c). Namely, the closed-loop matrix is substituted with its data-driven representation X^+G^i obtained from (2.98a). The solutions, P^i and W^i are then used to compute the update direction, i.e., the gradient of (2.96) evaluated at $(K_{\text{in}}^i, \Lambda^\ell)$, for $c = c^\ell$. The gradient is then used as descent direction in the policy improvement step (2.99b). The current solution K_{in}^i is updated along the descent direction with step-size $\gamma^i > 0$, selected by Armijo back-tracking line-search. The step-size selection plays a pivotal role in ensuring the stability of the closed-loop system. This procedure is terminated as soon as the threshold $\epsilon^\ell \in \mathbb{R}$ for the descent direction is met and it outputs the updated $K^{\ell+1}$, approximate solution of (2.97a).

Before providing the convergence result for Algorithm 4, we make the following assumption about the step-size γ^i in (2.99b).

Assumption 2.8. *At every iteration i, the step-size $\gamma^i \in \mathbb{R}$, $\gamma^i > 0$ is selected using the Armijo selection rule [30].* $\triangle$

Assumption 2.9. *The threshold ϵ^ℓ is chosen such that, for all $\ell \in \mathbb{N}$, $\epsilon^\ell \geq 0$ and $\epsilon^\ell \to 0$ as $\ell \to \infty$.* $\triangle$

Finally, an assumption over the initial gain K^0 is needed.

Assumption 2.10. *The initial gain $K^0 \in \mathbb{R}^{m\times n}$ is stabilizing for the dynamics* (2.84)*, namely, the closed-loop system $A_\star + B_\star K^0$ has all the eigenvalues inside the open unit disk.* $\triangle$

Remark 2.9. Assumption 2.10 can be satisfied by employing non-structured data-driven gain design strategies utilizing the already available data that satisfies Assumption 2.7, as discussed in, e.g., [70]. $\triangle$

The next Theorem presents the first main result of this paper.

Theorem 2.3. *Let Assumptions 2.7 and 2.8 hold. Let $\{K^\ell, \Lambda^\ell\}_{\ell\geq 0}$ be the sequence of gain-multiplier matrices generated by Algorithm 4. Then, any limit point $(\bar{K}, \bar{\Lambda})$ of the sequence $\{K^\ell, \Lambda^\ell\}_{\ell\geq 0}$ satisfies the first order necessary conditions for optimality associated to the optimal control problem* (2.87)*. Moreover, at each ℓ, the matrix K^ℓ is such that $(A_\star + B_\star K^\ell)$ is Schur, i.e., K^ℓ is a stabilizing feedback for the closed-loop system.* $\triangle$

Corollary 2.1. *Let Assumptions 2.6, 2.7, 2.8, 2.9 and 2.10 hold. Then, for any limit point $\bar{K}$ of the sequence $\{K^\ell\}_{\ell\in\mathbb{N}}$ generated by Algorithm 4, there exists a sufficiently large $\bar{\ell} \in \mathbb{N}$ such that, for all converging subsequences $\{K^l\}_{l\in\mathbb{N}}$ and for all $l \geq \bar{\ell}$, the projection of K^l over the set of the gains feasible for the sparsity constraint* (2.87d) *is stabilizing for the dynamics* (2.84). $\triangle$

Proof. Let $\mathcal{K} \subset \mathbb{R}^{m\times n}$ be the set of stabilizing gains for dynamics (2.84), defined as

$$\mathcal{K} := K \in \mathbb{R}^{n\times m} \mid (A_\star + B_\star K) \text{ is Schur}, \tag{2.101}$$

and let $\mathcal{K}_{\mathcal{S}} \subset \mathbb{R}^{m\times n}$ be the set of sparse gains, i.e.,

$$\mathcal{K}_{\mathcal{S}} := K \in \mathbb{R}^{n\times m} \mid K \circ \mathcal{S} = 0. \tag{2.102}$$

Denote the projection onto $\mathcal{K}_{\mathcal{S}}$ as $\mathcal{P}_{\mathcal{S}} : \mathbb{R}^{m\times n} \to \mathcal{K}_{\mathcal{S}}$. By assumption, the intersection between (2.101) and (2.102) is non-empty. Moreover, since $\mathcal{K}$ is open (see, e.g., [48]) and $\mathcal{K}_{\mathcal{S}}$ is an affine subspace of $\mathbb{R}^{n\times m}$, their intersection is an open set. Let $\bar{K}$ be a limit point of the sequence of gain matrices generated by Algorithm 4. By Theorem 3.1, $\bar{K}$ is sparse and stabilizing for dynamics (2.84), i.e., $\bar{K} \in \mathcal{K} \cap \mathcal{K}_{\mathcal{S}}$. By the definition of a limit point, there exists a subsequence $K^l l \in \mathbb{N}$ such that

$$\lim_{l\to\infty} K^l = \bar{K}. \tag{2.103}$$

Namely, $K^l_{(i,j)} = \bar{K}_{(i,j)}$ for all $i = 1, \ldots, m$, $j = 1, \ldots, n$. Hence, for all $\delta > 0$, there exists $\bar{\ell} > 0$ such that $|K^l - \bar{K}| \le \delta$ for all $l \ge \bar{\ell}$. Note that, by Theorem 3.1, all $K^l \in \mathcal{K}$, but not necessarily $K^l \in \mathcal{K}_{\mathcal{S}}$. Now, for all $l \ge \bar{\ell}$, consider the projection $\mathcal{P}_{\mathcal{S}}(K^l) \in \mathcal{K}_{\mathcal{S}}$. It holds that $|\mathcal{P}_{\mathcal{S}}(K^l) - \bar{K}| \le |K^l - \bar{K}| \le \delta$, namely, $\mathcal{P}_{\mathcal{S}}(K^l) \in \mathcal{K} \cap \mathcal{K}_{\mathcal{S}}$ for all $l \ge \bar{\ell}$. The proof follows ■

2.4.2 Algorithm Analysis

Before providing the formal proof of Theorem 2.3, we leverage on some auxiliary lemmas to constructively show how Algorithm 4 implements an augmented Lagrangian approach to solve problem (2.95). Specifically, since the matrix $A_\star$ is not known we can neither compute the reduced cost $J(K)$ nor the augmented Lagrangian associated to problem (2.95) nor their gradient (cf. Appendix B.1.3). To overcome this issue, we assume to have access to a set of system data X, U, X^+ satisfying condition (2.90). Notice that, given a solution of the inner minimization problem in (2.97a), the step (2.97) does not require any knowledge about the system dynamics. Indeed, for a fixed multiplier Λ^ℓ, we seek a solution of the inner minimization problem

$$\min_{K_{\text{in}}} \ \mathcal{L}_{c^\ell}(K_{\text{in}}, \Lambda^\ell) \tag{2.104}$$

leveraging on a gradient descent method. Starting from an initial guess K^0_{in}, the solution is iteratively updated following the rule

$$K^{i+1}_{\text{in}} = K^i_{\text{in}} + \gamma^i \Delta K^i \tag{2.105}$$

for all inner-loop iterations $i \in \mathbb{N}$, where for a given K^i_{in}, the update direction ΔK^i is computed as in (2.99a), with G^i, P^i, W^i solutions of (2.98a)-(2.98c).

The update direction ΔK^i is designed such that the following holds true.

Lemma 2.6. *Let Assumption 2.7 hold. Assume K_{in}^i is stabilizing, i.e., such that the closed-loop matrix $(A + BK^i)$ is Schur. Consider the update direction ΔK^i defined as*

$$\Delta K^i = -(RK_{in}^i + B_\star^\top P^i X^+ G^i)W^i - \Lambda^\ell - c^\ell(K_{in}^i \circ \mathcal{S})$$

where G^i is such that

$$\begin{bmatrix} I \\ K_{in}^i \end{bmatrix} = \begin{bmatrix} X \\ U \end{bmatrix} G^i$$

holds and (P^i, W^i) are, respectively, the solutions of

$$G^{i,\top} X^{+\top} P X^+ G^i - P + Q + K_{in}^{i,\top} R K_{in}^i = 0, \qquad X^+ G^i W G^{i,\top} X^{+\top} - W + I = 0$$

Then, ΔK^i is a descent direction for the unconstrained optimization problem (2.104). △

Proof. The gradient of the augmented Lagrangian function (2.96) is

$$\nabla_1 \mathcal{L}_{c^\ell}(K_{\text{in}}^i, \Lambda^\ell) = \Big(\nabla J(K_{\text{in}}^i) + \Lambda^\ell + c^\ell(K_{\text{in}}^i \circ \mathcal{S})\Big) \tag{2.106}$$

where in (2.106) we leverage on the fact that

$$\text{Tr}\left[\Lambda^\top (K_{\text{in}} \circ \mathcal{S})\right] = \text{Tr}\left[(\Lambda \circ \mathcal{S})^\top K_{\text{in}}\right] \tag{2.107}$$

and on the properties of the derivative of the trace. For which it holds that, defining

$$g(K_{\text{in}}, \Lambda) := \text{Tr}\left[(\Lambda^\ell \circ \mathcal{S})^\top K_{\text{in}}\right],$$

the gradient of $g(\cdot,\cdot)$ with respect to K_{in}, is equal to

$$\nabla_1 g(K_{\text{in}}^i, \Lambda^\ell) = (\Lambda^\ell \circ \mathcal{S}) \overset{(a)}{=} \Lambda^\ell \tag{2.108}$$

where in (a) we exploit the fact that, by construction (cf. (2.100a)) at each iteration Λ^ℓ is such that $\Lambda^\ell \circ \mathcal{S}^c = \Lambda^\ell$. Notice that, the descent direction $\nabla_1 \mathcal{L}_{c^\ell}(K_{\text{in}}^i, \Lambda^\ell)$ requires the calculation of the gradient of the reduced cost function $J(K_{\text{in}})$ with respect to its argument. By Lemma B.1, we know that, $\nabla J(K_{\text{in}})$ evaluated at K^i can be computed as

$$\nabla J(K) = \Big(RK_{\text{in}}^i + B^\top P^i (A_\star + B_\star K_{\text{in}}^i)\Big) W^i \tag{2.109a}$$

where P^i, W^i are the solutions of

$$(A_\star + B_\star K_{\text{in}}^i) W (A_\star + B_\star K_{\text{in}}^i)^\top - W = -I, \tag{2.109b}$$

$$(A_\star + B_\star K_{\text{in}}^i)^\top P (A_\star + B_\star K_{\text{in}}^i) - P = -(Q + K^{i\top} R K^i). \tag{2.109c}$$

Notice that matrix $(A_\star + B_\star K_{\text{in}}^i)$ is not available since the system matrix $A_\star$ is unknown. We know that, see e.g. [70], under Assumption 2.7, it holds,

$$(A_\star + B_\star K_{\text{in}}^i) = X^+ G^i$$

where G^i is such that (2.98a) hold. Hence, replacing the highlighted terms in (2.109) with X^+G^i, the gradient of the augmented Lagrangian function (2.106) with respect to its argument K_{in} reads

$$\nabla_1 \mathcal{L}(K_{\text{in}}^i, \Lambda^\ell) = \Big(RK_{\text{in}}^i + B_\star^\top P^i X^+ G^i\Big) W^i + \Lambda^\ell + c^\ell (K_{\text{in}}^i \circ \mathcal{S}) \tag{2.110}$$

with P^i, W^i such that

$$G^{i,\top} {X^+}^\top P^i X^+ G^i - P^i + Q + K_{\text{in}}^{i,\top} R K_{\text{in}}^i = 0 \tag{2.111a}$$

$$X^+ G^i W^i G^{i,\top} {X^+}^\top - W^i + I = 0. \tag{2.111b}$$

The proof follows by setting $\Delta K^i = \nabla_1 \mathcal{L}(K_{\text{in}}^i, \Lambda^\ell)$. Moreover, iteration (2.105) implements a gradient method applied to (2.104). ■

Lemma 2.7. *Let Assumptions 2.7 and 2.8 hold and let $\{K_{in}^i\}_{i>0}$ be the sequence generated by the inner routine of Algorithm 4. Then, for any stabilizing initial condition K_{in}^0, every limit point $\bar{K_{in}}$ of $\{K_{in}^i\}_{i>0}$ is a stationary point of problem* (2.104), *i.e., is such that $\nabla_1 \mathcal{L}_{c^\ell}(\bar{K_{in}}, \Lambda^\ell) = 0$. Moreover, at each iteration $i > 0$, the current solution K_{in}^i is stabilizing for dynamics* (2.84). △

Proof. For a fixed (Λ^ℓ, c^ℓ), consider the sequence $\{K_{\text{in}}^i\}_{i>0}$ generated by the inner-loop steps (2.98) and (2.99). Leveraging on Theorem 2.6, we know that at each iteration i, the current solution is updated following a descent method where the descent direction ΔK^i is the gradient of the of the augmented Lagrangian $\mathcal{L}_{c^\ell}(K_{\text{in}}, \Lambda^\ell)$ evaluated at K_{in}^i. In light of Assumption 2.8, the inner loop implements a gradient method for the resolution of (2.104) with step-size γ^i chosen following the Armijo rule. Hence, each limit point $\bar{K_{\text{in}}}$ of the sequence $\{K_{\text{in}}^i\}_{i>0}$ is a stationary point of problem (2.104). The proof of the first statement follows.

We now prove the second statement. First of all, notice that the function $J(\cdot)$ increases to infinity as the gain approaches the boundary of the (compact) set of stabilizing gains. Now, by definition of the Armijo rule, it holds, for all $i > 0$,

$$\mathcal{L}_{c^\ell}(K_{\text{in}}^{i+1}, \Lambda^\ell) \leq \mathcal{L}_{c^\ell}(K_{\text{in}}^i, \Lambda^\ell). \tag{2.112}$$

i.e., the sequence $\{\mathcal{L}_{c^\ell}(K_{\text{in}}^i, \Lambda^\ell)\}_{i>0}$ is non-increasing and bounded from above by $\mathcal{L}_{c^\ell}(K_{\text{in}}^0, \Lambda^\ell) < +\infty$. This is sufficient to ensure the stability of the feedback gain K_{in}^i at each iteration i. ■

Proof of Theorem 2.3

We are now in the position to give a formal proof of Theorem 2.3. The proof strategy relies on showing that Algorithm 4 implements an augmented Lagrangian algorithm with inexact minimization, i.e., the minimum of the augmented Lagrangian for a fixed (Λ^ℓ, c^ℓ) is not found exactly. We leverage on the convergence result given in [30, Proposition 5.2.2], presented in Appendix A.2.

Let $\{K^\ell\}_{\ell\in\mathbb{N}}$, $\{\Lambda^\ell\}_{\ell\in\mathbb{N}}$, $\{c^\ell\}_{\ell\in\mathbb{N}}$, $\{\epsilon^\ell\}_{\ell\in\mathbb{N}}$ be the sequences generated by Algorithm 4. Notice that the cost function $J(K)$ is continuously differentiable over its domain, while the (linear) constraint $K \circ \mathcal{S}$ is of class C^∞. Firstly, for any fixed pair (Λ^ℓ, c^ℓ) the inner routine (cf. Theorem 2.6) implements a gradient method for the resolution of the augmented Lagrangian minimization. Hence, there always exists an inner loop iteration $i^\star$, such that $\nabla_1 \mathcal{L}_c^\ell(K_{\text{in}}^{i^\star}, \Lambda^\ell) \leq \varepsilon^\ell$. Since in Algorithm 4, we set, at each ℓ, $K^\ell = K_{\text{in}}^{i^\star}$, Assumption (i) in Theorem A.1 is satisfied. Secondly, the Lagrangian multiplier Λ^ℓ is updated such that the sequence $\{\Lambda^\ell\}$ is bounded (see [26, Section 2.5]). Hence, Assumption (ii) in Theorem A.1 is satisfied. Third, Choosing $\beta \in \mathbb{R}$, $\beta > 0$ ensures the sequence $\{c^\ell\}_{\ell\in\mathbb{N}}$ to be such that $\lim_{\ell\to\infty} c^\ell = \infty$, and $0 \leq c^\ell \leq c^{\ell+1}$ for all ℓ. Hence, Assumption (iii) in Theorem A.1 is satisfied. Finally, by Assumption 2.9, Assumption (iv) in Theorem A.1 is trivially satisfied. As a consequence, from Theorem A.1, we have that any limit point $(\bar{K}, \bar{\Lambda})$ of the sequences $\{(K^\ell, \Lambda^\ell)\}_{\ell\in\mathbb{N}}$ is such that the first order KKT conditions for optimality associated to problem (2.95), and hence of (2.87), are satisfied. The proof follows. $\triangle$

2.4.3 Regularized Approach for Noisy Data

Notice that, in Algorithm 4, when evaluating matrix G^i in (2.98a), we do not take advantage of the additional degree of freedom represented by the nontrivial nullspace of the data matrix $[X^\top \; U^\top]^\top$, cf. Remark 2.8. Indeed, it could be convenient to bias the selection of G towards matrices with favorable properties, e.g., noise rejection in case of noisy samples. Technically speaking, this could be achieved by leveraging on a regularized, LMI-based, policy evaluation step. In the following, we provide a robustified implementation of Data-driven AL Algorithm for Sparse Feedback Design, which we termed Regularized Data-driven AL for Sparse Feedback Design, leveraging an LMI-based regularized policy evaluation step. Before giving the formal implementation of the robustified strategy, we motivate the need for a regularized algorithm by considering the case in which the system evolution is also affected by an additive noise. Namely, data are collected from a discrete-time linear and time-invariant system in the form

$$x_{t+1} = A_\star x_t + B_\star u_t + d_t, \quad x_0 \sim \mathcal{D}^0 \tag{2.113}$$

where $d_t \in \mathbb{R}^n$ is a disturbance term. Hence, by setting

$$D := [d_0, d_1, \dots, d_{T-1}] \in \mathbb{R}^{n \times T}, \tag{2.114}$$

the following condition, in place of (2.89), is satisfied

$$(X_d^+ - D) = \begin{bmatrix} A_\star & B_\star \end{bmatrix} \begin{bmatrix} X_d \\ U \end{bmatrix}. \tag{2.115}$$

where X_d and X_d^+ collect the state trajectories of (2.113). Consider now a gain matrix $K \in \mathbb{R}^{m \times n}$ and introduce the data matrix $G \in \mathbb{R}^{T \times n}$ satisfying (2.91). In light of (2.115), we can parametrize the closed-loop matrix as

$$(A_\star + B_\star K) = (X_d^+ - D)G. \tag{2.116}$$

Notice that, matrix G is selected such that

$$\begin{bmatrix} I \\ K \end{bmatrix} = \begin{bmatrix} X_d \\ U \end{bmatrix} G. \tag{2.117}$$

Necessary and sufficient conditions for a gain matrix K to stabilize the noisy system (2.113), are that there exists a symmetric, positive-definite matrix P, such that

$$G^\top (X_d^+ - D)^\top P (X_d^+ - D) G - P + Q + K^\top R K \leq 0. \tag{2.118}$$

Notice that, measurements of the noise are typically not available, i.e., D is unknown. Hence, D should be disregarded and we employ directly the (noisy) measurements of the state trajectories X_d^+. Thus, we can only impose

$$G^\top {X_d^+}^\top P X_d^+ G - P + Q + K^\top R K \leq 0, \tag{2.119}$$

where G is chosen such that (2.93), rather than its noisy version (2.116), holds. A regularization term can hence be exploited to bias the choice of G towards solutions that (try to) also satisfy (2.118), while enforcing (2.119). The implication of the feasibility (2.118) from the feasibility of the corresponding (2.119) has been investigated in [71, 80].

Regularized data-driven algorithm

Moving from Algorithm 4, the policy evaluation steps (2.98a)-(2.98b), can be rewritten as an equivalent optimization problem, which gives room for the introduction of a regularizing term.

Consider a generic inner loop iteration $i \in \mathbb{N}$, with associated gain K_{in}^i. Then, a solution (P^i, G^i) of (2.98a) and (2.98b) can be equivalently computed as the solution of

$$\begin{aligned}
\min_{P,G} \quad & \operatorname{Tr}[P] && (2.120\text{a})\\
\text{subj.to} \quad & G^\top {X_d^+}^\top P X_d^+ G - P + Q + {K_{\text{in}}^i}^\top R K_{\text{in}}^i \leq 0 && (2.120\text{b})\\
& P > 0 && (2.120\text{c})\\
& \begin{bmatrix} I \\ K_{\text{in}}^i \end{bmatrix} = \begin{bmatrix} X_d \\ U \end{bmatrix} G. && (2.120\text{d})
\end{aligned}$$

Notice that, operatively, problem (2.120) is solved by resorting to an equivalent convex reformulation.

Let the function $r : \mathbb{R}^{T\times n} \to \mathbb{R}$ represent the regularizer. To obtain a regularized instance of problem (2.120), we can add to the original cost (2.120a) a regularization term to be applied on G, see, e.g., the discussion in [80]. The operative numerical strategy to be implemented in the regularized case is summarized in Algorithm 5. Differently from Algorithm 4, the policy evaluation steps (2.121)–(2.122) are performed concurrently at each iteration via a regularized version of problem (2.120). More in detail, we choose $r(GP^{-1}) = \lambda \|GP^{-1}\|_F^2$.

Notice that, the policy evaluation step (2.121), represents a tractable reformulation of the regularized version of problem (2.120). The equivalence is formally stated in the the following Theorem which is the second main result of the paper.

Theorem 2.4. *Let Assumption 2.7 hold. Consider a gain K_{in}^i stabilizing for dynamics* (2.84). *Let $(Y_{\text{sdp}}^i, L_{\text{sdp}}^i, Z_{\text{sdp}}^i)$ be a solution of the semi-definite program* (2.121), *where $Y_{\text{sdp}}^i \in \mathbb{R}^{n\times n}$, $L_{\text{sdp}}^i \in \mathbb{R}^{T\times n}$ and $Z_{\text{sdp}}^i \in \mathbb{R}^{T\times T}$. Then, the solution pair $(P_{\text{reg}}, G_{\text{reg}})$ computed as $P_{\text{reg}} = Y_{\text{sdp}}^{-1}$, $G_{\text{reg}} = L_{\text{sdp}} Y_{\text{sdp}}^{-1}$, represents a solution to a regularized version of problem* (2.120) *which reads as*

$$\begin{aligned}
\min_{P,G} \quad & \operatorname{Tr}[P] + \lambda \|GP^{-1}\|_F^2 && (2.123\text{a})\\
\text{subj.to} \quad & G^\top {X_d^+}^\top P X_d^+ G - P + Q + {K_{in}^i}^\top R K_{in}^i \leq 0 && (2.123\text{b})\\
& P > 0 && (2.123\text{c})\\
& \begin{bmatrix} I \\ K_{in}^i \end{bmatrix} = \begin{bmatrix} X_d \\ U \end{bmatrix} G. && (2.123\text{d})
\end{aligned}$$

Algorithm 5 Regularized Data-driven AL for Sparse Feedback Design

```
Require: The same of Algorithm 4
  for $\ell = 0, 1, 2 \dots$ do
    Set $K_{\text{in}}^{0} = K^{\ell}$
    for $i = 0, 1, 2 \dots$ do
      Policy Evaluation
      Find $(Y^i, L^i, Z^i)$ solution of
        $\min_{Y,L,Z} \ \mathrm{Tr}[Y^{-1}] + \lambda \, \mathrm{Tr}[Z]$    (2.121a)
        $\text{subj.to} \ \begin{bmatrix} Y & L^\top X_d^{+} & Y \\ X_d^{+} L & Y & 0 \\ Y & 0 & (Q + {K_{\text{in}}^{i}}^\top R K_{\text{in}}^{i})^{-1} \end{bmatrix} > 0$    (2.121b)
        $\begin{bmatrix} I \\ K_{\text{in}}^{i} \end{bmatrix} Y = \begin{bmatrix} X_d \\ U \end{bmatrix} L$    (2.121c)
        $\begin{bmatrix} Z & L \\ L^\top & I_n \end{bmatrix} > 0$    (2.121d)
      Set $P^i = (Y^i)^{-1}$, $G^i = L^i (Y^i)^{-1}$
      Compute $W^i$ solution of
        $X_d^{+} G^i W G^{i,\top} {X_d^{+}}^\top - W + I = 0$    (2.122)
      Policy Improvement
      Compute the descent direction $\Delta K^i$ as in (2.99a)
      Update the gain matrix $K_{\text{in}}^{i+1}$ as in (2.99b)
      Break if $\|\Delta K^i\| \leq \epsilon^{\ell}$
    Set $K^{\ell+1} = K_{\text{in}}^{i+1}$
    Update the multiplier $\Lambda^{\ell+1}$ and the penalty parameter $c^{\ell}$ as in (2.100)
```

with $\lambda\|GP^{-1}\|_F^2$ being the regularization term, where $\lambda \geq 0$ is a tuning parameter. Indeed, larger values λ favors solutions with GP^{-1} having small norm. △

Proof. The proof follows by showing the equivalence between (2.121) and (2.123). We start by manipulating the constraint (2.123b). Pre- and post-multiplying (2.120b) by P^{-1}, we obtain

$$\begin{aligned} P^{-1}G^\top X_d^{+\top} P X_d^+ G P^{-1} - P^{-1} \\ + P^{-1}QP^{-1} + P^{-1}K_{\text{in}}{}^\top R K_{\text{in}} P^{-1} \leq 0. \end{aligned} \tag{2.124}$$

Next, let us introduce the variables $Y \in \mathbb{R}^{n\times n}$ and $L \in \mathbb{R}^{T\times n}$ such that

$$(P, G) \longmapsto (Y, L) = (P^{-1}, G P^{-1}) \tag{2.125}$$

Hence, we can write (2.124) as

$$L^\top X^{+\top} Y^{-1} X^+ L - Y + YQY + YK_{\text{in}}{}^\top R K_{\text{in}} Y \leq 0 \tag{2.126}$$

with $Y > 0$. Inequality (2.126) can be further rewritten as

$$\begin{bmatrix} X^+ L \\ Y \end{bmatrix}^\top \begin{bmatrix} Y & 0 \\ 0 & (Q + K_{\text{in}}{}^\top R K_{\text{in}})^{-1} \end{bmatrix}^{-1} \begin{bmatrix} LX^+ \\ Y \end{bmatrix} - Y \leq 0$$

which, by applying the Schur complement lemma, is also equivalent to

$$\begin{aligned} &\begin{cases} \begin{bmatrix} Y & 0 \\ 0 & (Q + K_{\text{in}}{}^\top R K_{\text{in}})^{-1} \end{bmatrix} \geq 0 \\ Y - \begin{bmatrix} X^+ L \\ Y \end{bmatrix}^\top \begin{bmatrix} Y & 0 \\ 0 & (Q + K_{\text{in}}{}^\top R K_{\text{in}})^{-1} \end{bmatrix}^{-1} \begin{bmatrix} X^+ L \\ Y \end{bmatrix} \geq 0 \end{cases} \\ \iff &\begin{bmatrix} Y & L^\top X^{+\top} & Y \\ X^+ L & Y & 0 \\ Y & 0 & (Q + K_{\text{in}}{}^\top R K_{\text{in}})^{-1} \end{bmatrix} \geq 0. \end{aligned} \tag{2.127}$$

As for the constraint (2.120d), it can be post-multiplied by P^{-1} and reformulated as

$$\begin{aligned} 0 &= \begin{bmatrix} I \\ K_{\text{in}} \end{bmatrix} P^{-1} - \begin{bmatrix} X \\ U \end{bmatrix} G P^{-1} \\ &\overset{(a)}{=} \begin{bmatrix} I \\ K_{\text{in}} \end{bmatrix} Y - \begin{bmatrix} X \\ U \end{bmatrix} L \end{aligned}$$

where in (a) we introduce variables Y and L (cf. (2.125)). Finally, consider the regularizer

$\lambda\|GP^{-1}\|_F^2$. The equivalence between (2.123) and (2.121) is ensured by means of the cost (2.121a) and the constraint (2.121d). Indeed, one can observe that

$$\|GP^{-1}\|_F^2 = \mathrm{Tr}\left[G\,P^{-1}P^{-1}G^\top\right] \overset{(a)}{=} \mathrm{Tr}\left[LL^\top\right]. \tag{2.128}$$

where (a) holds by (2.125). We can now rewrite problem (2.123) as

$$\begin{aligned}
\min_{Y,L}\ & \mathrm{Tr}[Y^{-1}] + \lambda\,\mathrm{Tr}[LL^\top] \\
\text{subj.to}\ & \begin{bmatrix} Y & L^\top X_d^+ & Y \\ X_d^+ L & Y & 0 \\ Y & 0 & (Q + {K_{\text{in}}^i}^\top R K_{\text{in}}^i)^{-1} \end{bmatrix} > 0 \\
& \begin{bmatrix} I \\ K_{\text{in}}^i \end{bmatrix} Y = \begin{bmatrix} X_d \\ U \end{bmatrix} L
\end{aligned}$$

Introducing the variable $Z \in \mathbb{R}^{T\times T}$ and leveraging an epigraph reformulation, we further rewrite the problem as

$$\begin{aligned}
\min_{Y,L,Z}\ & \mathrm{Tr}[Y^{-1}] + \lambda\,\mathrm{Tr}[Z] \\
\text{subj.to}\ & \begin{bmatrix} Y & L^\top X_d^+ & Y \\ X_d^+ L & Y & 0 \\ Y & 0 & (Q + {K_{\text{in}}^i}^\top R K_{\text{in}}^i)^{-1} \end{bmatrix} > 0 \\
& \begin{bmatrix} I \\ K_{\text{in}}^i \end{bmatrix} Y = \begin{bmatrix} X_d \\ U \end{bmatrix} L \\
& Z - LL^\top > 0.
\end{aligned}$$

By applying the Schur complement lemma, it holds

$$\begin{cases} Z - L^\top I_n L \geq 0 \\ I_n \geq 0 \end{cases} \iff \begin{bmatrix} Z & L \\ L & I_n \end{bmatrix} \geq 0 \tag{2.129}$$

which admits a convex reformulation via the S-procedure. Hence, problem (2.120) can be written as (2.121), which can be effectively processed by any LMI solver. Calling Y_{sdp}^i, L_{sdp}^i the solution of (2.121), we can compute ${P_{\text{reg}}}^i$, ${G_{\text{reg}}}^i$ as

$${P_{\text{reg}}}^i = (Y_{\text{sdp}}^i)^{-1}, \qquad {G_{\text{reg}}}^i = L_{\text{sdp}}^i (Y_{\text{sdp}}^i)^{-1}.$$

This concludes the proof. ■

Remark 2.10. The choice of the regularizer $\lambda\|GP^{-1}\|$ in (2.123a) is motivated by the fact that, with (2.124) at hand, one could bias the term GP^{-1} to have small norm in order to try to satisfy (2.118), while enforcing (2.119), i.e., (2.124). $\triangle$

Remark 2.11. Notice that, in the noise free case, the semi-definite program (2.121), with $\lambda = 0$, solves problem (2.120). $\triangle$

2.4.4 Numerical Simulations

In this section we present numerical computations on a distributed control scenario. the coupling among the agents dynamics is modeled by an undirected graph $\mathcal{G} = (V, \mathcal{E})$, where $V = \{1, \dots, N\}$ is the set of agents and $\mathcal{E} \subseteq V \times V$ is the set of edges. An edge (i, j) belongs to $\mathcal{E}$ if and only if agents i and j are coupled with each other, in which case we have also $(j, i) \in \mathcal{E}$. We denote as $\mathcal{N}_i = \{j \in V : (i, j) \in \mathcal{E}\}$ the neighbors of each agent i. The graph is randomly generated according to an Erdős-Rényi model with edge probability $p = 0.05$. The unknown linear dynamics is defined, for all $i = 1, \dots, N$, as

$$x_{i,t+1} = A_{ii}x_{i,t} + \sum_{j \in \mathcal{N}_i} A_{ij}x_{j,t} + B_i u_{i,t}, \tag{2.130}$$

where $x_i \in \mathbb{R}^2$, $u_i \in \mathbb{R}^2$ for all $i = 1, \dots, N$. The state and input matrices are $A_{ii} = \left[\begin{smallmatrix} 1.01 & 0.1 \\ 0.1 & -1.01 \end{smallmatrix}\right]$, $A_{ij} = \left[\begin{smallmatrix} 0.5 & 0 \\ 0 & -0.5 \end{smallmatrix}\right]$, for all $j \in \mathcal{N}_i$, and $B_i = I_2$. Accordingly, $A_\star \in \mathbb{R}^{2N \times 2N}$ is a block matrix whose ij-th block $A_{\star,(i,j)} \in \mathbb{R}^{2\times 2}$ is defined as $A_{\star,(i,j)} = A_{ii}$, if $i = j$, $j \in \mathcal{N}_i$, $A_{\star,(i,j)} = A_{ij}$, if $i \neq j$, $j \in \mathcal{N}_i$, $A_{\star,(i,j)} = 0_{2\times 2}$, otherwise. Similarly, $B_\star \in \mathbb{R}^{2N \times 2N}$ is a block diagonal matrix whose ij-th block $B_{\star,(i,j)} \in \mathbb{R}^{2\times 2}$ is defined as $B_{\star,(i,j)} = B_i$, if $i = j$, else, $B_{\star,(i,j)} = 0_{2\times 2}$. Thus, $n = 2N$, $m = 2N$. We point out that the open-loop system is unstable, as the maximum eigenvalue of $A_\star$ is greater than 2. The quadratic cost is designed such that the matrix $Q = Q^\top \geq 0 \in \mathbb{R}^{2N \times 2N}$ matches the graph topology, i.e., Q is a block matrix whose ij-th block $Q_{(i,j)} \in \mathbb{R}^{2\times 2}$ is defined as $Q_{(i,j)} = 10 \cdot I_2$ if $i = j, j \in \mathcal{N}_i$, $Q_{(i,j)} = I_2$ if $i \neq j, j \in \mathcal{N}_i$, while $Q_{(i,j)} = 0_2$ otherwise. The matrix $R \in \mathbb{R}^{2N \times 2N}$ is chosen as $R = 5 \cdot I_{2N}$. The initial conditions are assumed to be normally distributed about the origin. We next consider two different settings: (i) a deterministic scenario, i.e., measurements of the states $x_{i,t}$ are noise free, to showcase the capabilities of Algorithm 4, and (ii) a setting where measurements are corrupted by random noise, to highlight the role of the regularized approach implemented by Algorithm 5.

Results in a deterministic setting

We evaluate the performances of Data-driven AL Algorithm for Sparse Feedback Design by running a Monte-Carlo simulation with $n_{\text{MC}} = 100$ different scenarios. In each scenario, we consider $N = 100$ agents communicating accordingly to a different, randomly generated, graph. The data-set X, U, X^+ is obtained by applying a sequence of randomly generated control inputs on the dynamics to ensure that Assumption 2.7 is satisfied.

Each control input is drawn from a normal distribution with mean value 1 and variance 1. Algorithm 4 is implemented with parameters $c^0 = 5$ and $\beta = 5$. To ensure numerical stability, the parameter c^ℓ is saturated at 10^3.

In Figure 2.12 (above), it is represented the evolution along the iterations of the average infinite horizon cost associated to the gain computed by Algorithm 4 and its 3-standard deviation band over n_{MC} Monte Carlo simulations. The average infinite horizon cost is defined, for all $\ell \in \mathbb{N}$ as

$$\hat{J}^\ell = \frac{1}{n_{\text{MC}}} \sum_{m=1}^{n_{\text{MC}}} \operatorname{Tr}\left[{A_{\text{cl}}}_m^{\ell}{}^\top (Q + {K_m^\ell}^\top R K_m^\ell) {A_{\text{cl}}}_m^\ell \right]$$

where ${A_{\text{cl}}}_m^\ell = (A_\star + B_\star K_m^\ell)$ for all $m = 1, \dots, n_{\text{MC}}$. As a benchmark, we depict also the average infinite horizon cost, denoted as J^{unc}, associated to the optimal *unconstrained* gain K_m^{unc}, solution of the unconstrained instance of Problem 2.87, for all $m = 1, \dots, n_{\text{MC}}$. Notice that, as the sparsity constraints have to be respected, the average infinite-horizon cost converges to a value which is greater than the unconstrained one. The average violation of the sparsity constraint computed as $\frac{1}{n_{\text{MC}}} \sum_{m=1}^{n_{\text{MC}}} \|K_m^\ell \circ S_c\|_F$ and its 3-standard deviation band over n_{MC} Monte Carlo simulations is depicted in Figure 2.12 (below). Notice that the violation is progressively decreased along iterations. Finally, in Figure 2.13 it is possible to observe the structure of the feedback gain across the iterations. Notice that the sparsity degree of matrix K^ℓ is progressively refined until the desired structure is obtained.

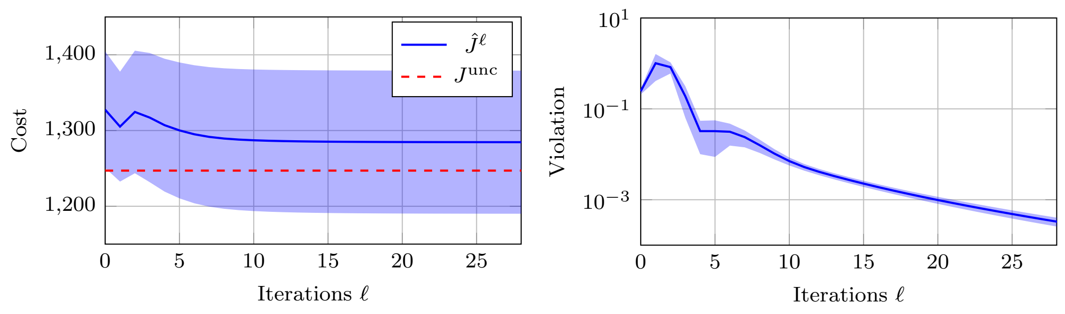


Figure 2.12: (Left) Evolution along iterations $\ell \in \mathbb{N}$ of the average infinite horizon cost associated to to the gain computed by Data-driven AL Algorithm for Sparse Feedback Design $\hat{J}^\ell$ (blue) and the average infinite horizon cost associated to the optimal unconstrained gain J^{unc} (dashed red) computed over 100 Monte Carlo simulations. In light blue it is depicted the 3-standard deviation band associated to $\hat{J}^\ell$. (Right) Evolution along iterations $\ell \in \mathbb{N}$ of the average constraint violation associated to to the gain computed by Data-driven AL Algorithm for Sparse Feedback Design and the 3-standard deviation band over 100 Monte Carlo simulations.

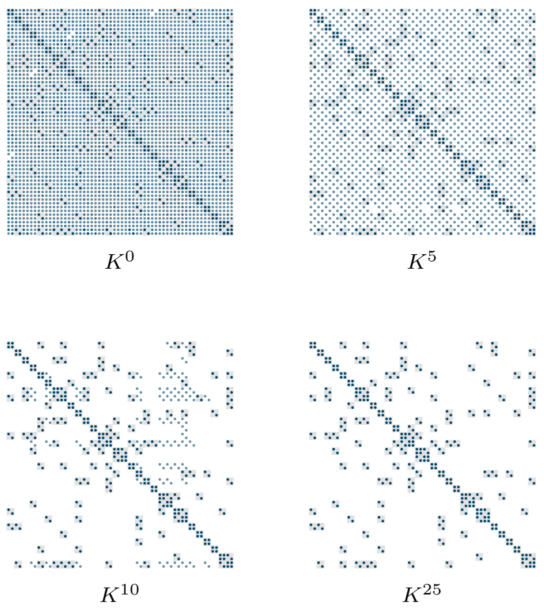


Figure 2.13: Representation of the sparse structure of the first 60×60 components of K^ℓ at iterations $\ell = 0, 5, 10, 25$ associated to a single Monte Carlo simulation. Blue dots represent the elements of K^ℓ, with color intensity proportional to their magnitude. The gray squares represent the desired structure defined by $\mathcal{S}$.

Results in a noisy setting

We now consider the original dynamics, corrupted by additive noise, namely for all $i = 1, \dots, N$,

$$x_{i,t+1} = A_{ii}x_{i,t} + \sum_{j \in \mathcal{N}_i} A_{ij}x_{j,t} + B_i u_{i,t} + d_{i,t} \tag{2.131}$$

where A_{ii}, A_{ij} and B_i are the same as above, for all $i, j = 1, \dots, N$, and $d_{i,t}$ is a random noise sampled from a normal distribution with mean value 0.01, and variance 1. We start considering a scenario with $N = 10$ agents and communication graphs with edge probability $p = 0.05$. The noisy dataset X_d, U, X_d^+ is collected, as before, by applying a sequence of randomly generated inputs until Assumption 2.7 is satisfied. We first run Regularized Data-driven AL for Sparse Feedback Design with regularization parameter $\lambda = 10$. The resulting violation of the sparsity constraint is depicted in Figure 2.14 as the norm of the constraint (2.87d) across iterations.

The importance of the regularization process is highlighted by running Algorithm 5 over $n_{\text{test}} = 20$ different scenarios for different regularization values λ in two settings with different edge probability values for the communication graph, namely $p = 0.05$ and $p = 0.2$. Each scenario corresponds to a different, randomly generated, graph and noisy dataset. In Figure 2.15 the percentage of controllers generated, from noisy data, by Algorithm 5 that are stabilizing for the original dynamics (2.131). Notice that the

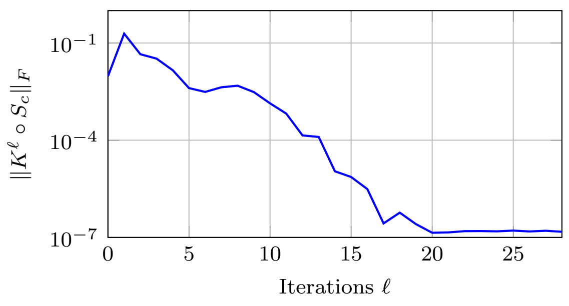


Figure 2.14: Evolution along iterations $\ell \in \mathbb{N}$ of the constraint violation associated to to the gain computed by Algorithm 5.

percentage of stabilizing controllers is proportional to λ.

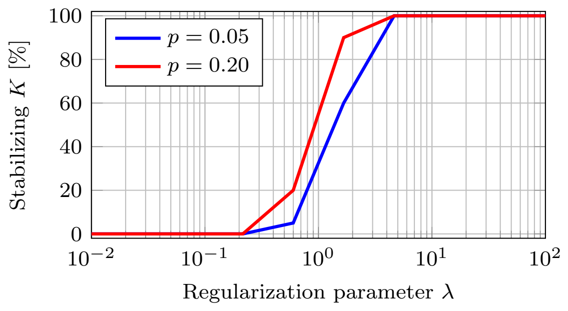


Figure 2.15: Percentage of controllers stabilizing for the original dynamics (2.131) generated by Algorithm 5 using noisy data for different values of λ.

Chapter 3

Feedback Embedding Paradigm for Numerical Optimal Control of Large-scale, Multi-agent and Uncertain Systems

In this chapter, we introduce a novel class of numerical optimal control algorithms tailored for solving nonlinear optimal control problems. These algorithms are applicable in various contexts, from large-scale scenarios to multi-agent settings.

First-order methods are well recognized effective tools for the resolution of optimization problems with large number of decision variables, e.g., neural network training. Drawing inspiration from their success, we introduce a framework of *first-order* numerical strategies developed for the resolution of optimal control problems involving large-scale dynamics and multi-agent systems. Central to our approach is the *feedback embedding* paradigm relying on the introduction of a feedback policy within the problem formulation. This technique not only improves the numerical stability of the developed algorithms but also enables us to recast the optimal control problem into a reduced optimization problem, amenable to state-of-art numerical optimization strategies. This feature has been instrumental in extending our methodology to the *distributed optimal control* framework, where tracking-like approaches can be deployed for the reconstruction of global quantities. Furthermore, it paved the way for the development of novel learning-based methodologies that integrate *model-based* and *data-driven* strategies for nonlinear optimal control. Finally, the inherent flexibility of the proposed framework enabled the extension of the method to *stochastic settings*, leveraging a Stochastic Gradient Descent (SGD) scheme.

Section 3.2 introduces the general feedback-embedding framework for solving nonlinear discrete-time optimal control problems in a large-scale setting. This framework, which we termed GoPRONTO, short for Generalized first-Order PROjectioN operator method for Trajectory Optimization[1], incorporates the original dynamics into a closed-loop system. By utilizing this feedback-based methodology, we can reinterpret the optimal control problem as a cost function minimization task, paving the way for advanced first-order numerical optimal control strategies. In Section 3.3, we expand the feedback-embedding paradigm to address distributed optimal control problems within networks of cooperative multi-agent systems. The resultant algorithm is a distributed scheme that iteratively updates solutions through a distributed tracking mechanism. This mechanism leverages inter-agent communication to effectively reconstruct global quantities. In Section 3.4 we combine model-based and data-driven optimal control strategies by assuming the ability to actuate control input sequences onto a real system while only an inaccurate description of the dynamics is available for the control design. This approach integrates trajectory optimization with Gaussian process regression, iteratively refining the model and performing optimization steps. Finally, Section 3.5 details how our first-order framework is adapted for a stochastic optimal control setting. The proposed scheme, implementing a SGD algorithm, shows asymptotic convergence in expectation. The results of this chapter are based on [208, 209, 212].

3.1 Literature Review

A vast number of engineering applications in Automation and Robotics require the resolution of an optimal control problem involving a nonlinear system. Next we review some numerical methods for nonlinear optimal control. First of all, we consider the classical centralized framework, i.e., all the calculations are performed by a central unit. Then, we review the most recent contributions to the distributed and multi-agent optimal control framework, together with the latest works on learning-driven and stochastic optimal control.

Literature on Numerical Optimal Control

Existing centralized numerical methods for the resolution of optimal control problems are typically classified as *indirect* methods (see the recent works [177, 203]) and *direct* ones (see [77] for an overview). While *indirect methods* aim to satisfy the necessary conditions of optimality and typically solve a (two-point) boundary-value problem arising from calculus of variations (see, e.g., [134, 201]) or from Pontryagin's Maximum Principle (see, e.g., [181]), *direct methods* rely on the parameterization of the control, the discretization of the states by an appropriate integration scheme, and then the solution of the resulting nonlinear program (NLP). In the overview paper [77], direct methods

[1]This acronym is chosen as a tribute to professor Hauser's PRONTO, see [113]

are subclassified into two different categories: *simultaneous* and *sequential*. Simultaneous approaches commonly take into account the constraints within the optimization, so that all the original variables, i.e., the controls and the states, are treated as decision variables. In general, the NLP formulation of the original optimal control problem is obtained via collocation methods [34, 227] as well as multiple shooting methods [38]. In these approaches, the optimality conditions of the original optimal control problem are related to the Karush-Kuhn-Tucker (KKT) optimality conditions of the NLP [32]. The NLP is then addressed solving directly the KKT conditions of the problem by Newton's type optimization algorithms due to their fast convergence rate [169]. The two major families of Newton type optimization methods are Sequential Quadratic Programming (SQP) and Interior Point optimization (IP). The literature on SQP is quite vast and we refer the interested reader to [77, 169] for detailed overviews. SQP methods for the resolution of optimal control problems have been employed in various applications. For example, in [248], an algorithm based on SQP is proposed to solve a vehicle coordination optimal control problem. For IP methods, instead, we refer to [43, 240]. Widely adopted implementations of nonlinear IP methods are represented by the toolboxes IPOPT [35] and the more recent FORCES [246]. The major drawback of these approaches is that they do not enjoy a "dynamic feasibility". That is, the state-input curves computed at each iteration do not satisfy the dynamics in general. However this feature can be extremely important in real-time control schemes (as, e.g., in Model Predictive Control) since it may allow for suboptimal schemes stopping after few iterations. Approaches to deal with feasibility of the dynamic constraints in SQP methods have been presented in [222] and in [17]. In contrast to simultaneous methods, sequential approaches tackle the NLP in the reduced space of control variables only. Indeed, the state trajectory associated to any input sequence can be recovered by forward simulation of the dynamics with the advantage that a system trajectory is available at each iteration. A first-order sequential approach for the resolution of nonlinear optimal control problems is presented, e.g., in [27, Section 1.9], where an adjoint equation is used to compute the descent direction.

Dynamic Programming (DP) gives raise to other numerical approaches solving the optimal control problem by relying on the well-known principle of optimality, see, e.g., [22, 31, 46]. Since it produces the optimal control policy for any state of the system, feasibility of the trajectory is always granted. The extension of DP to the nonlinear framework is represented by Differential Dynamic Programming (DDP). DDP proceeds iteratively by quadratic approximations about the current state-input trajectory of the cost and the dynamics. It is worth observing that DDP can be seen as a sequential, direct method where a Newton's step is adopted [82]. Remarkably, DDP must evaluate second-order derivatives of the dynamics when computing the feedback controller. Moreover, DDP exhibits only local convergence to the optimum. Other algorithms based on the resolution of quadratic approximations of the nonlinear optimal control problem about a state-input trajectory are presented in [69, 82, 147, 174]. In [144] Iterative-LQR, an ad-hoc version of the DDP approach, is presented. In [113], the *PRojection Operator Newton Method for Trajectory Optimization (PRONTO)* is proposed. Here, through the use of a control feedback, the dynamically constrained optimization problem is converted into an unconstrained one to which a Newton's method is applied. Extensions of PRONTO have been proposed for constrained optimization [114] and optimal control on Lie groups [199]. PRONTO has been applied to several contexts as motion planning of single and multiple vehicles, see, e.g., [3] and references therein. In [93] an iterative numerical method based on PRONTO is developed, where the optimal control problem is tackled via a constrained-gradient descent. A discrete-time counterpart of PRONTO is presented in [17] with a projected SQP reformulation.

A literature intimately connected with optimal control is the one associated with Model Predictive Control (MPC). In this approach, at each time instant a finite-horizon optimal control problem is solved and the first (optimal) control input is applied. A detailed overview is provided, e.g., in [187]. Economic MPC represents the extension of MPC to generic cost functions, see, e.g., [90] for a detailed overview. MPC controllers are applied in a great variety of problem fields. Recent applications include, e.g., vehicle coordination [119], supply chain management [139] and autonomous racing [129]. In the MPC domain, computational tractability of the optimal control algorithms is an open issue. For example, in embedded MPC (where only limited computational time and power is available) the deployment of methods based only on first-order derivatives is often considered, see e.g., in [123, 141]. Decomposition techniques to reduce the computational complexity are proposed instead in [213], while a suboptimal strategy exploiting fixed sensitivities is presented in [247].

Literature on Multi-agent and Distributed Optimal Control

Another current area of research is represented by the extension to a distributed computation framework of state-of-art centralized strategies. In [95] distributed optimal control problems are addressed in the context of multiscale dynamical systems, where microscopic control laws at the agent level are derived from the optimal macroscopic description of the system. Distributed control schemes based on an Alternating Direction Method of Multipliers (ADMM) approach are proposed in [111, 190, 202, 214, 232] with applications in multi-robot systems. An extension of DDP to a multi-agent setting is proposed in [132, 215], where some game theoretic DDP-based methods are presented. Distributed optimal control algorithms represent a building block for *co-operative distributed* Model Predictive Control (MPC) to plan feasible trajectories. The survey paper [164] provides an overview of distributed (economic) MPC approaches, however in several works a distributed algorithm for the solution of the optimal control problem is assumed. From an algorithmic perspective, in [101] an accelerated gradient method is proposed for distributed cooperative MPC, while in [118] an augmented Lagrangian method is proposed in the context of convex MPC. Augmented Lagrangian and ADMM-based distributed optimization and optimal control algorithms gained traction in both cost- and constraint-coupled scenarios [16, 85]. For constraint coupled problems in [51] a distributed MPC scheme, GRAMPC, is developed based on the ADMM approach. A distributed augmented Lagrangian algorithm is proposed in [83], which may be extended to optimal control applications. Notice that, a coordination step is required to ensure consensus among the dual variables. In [219], a decentralized Sequential Quadratic Programming approach, based on ADMM methods, tailored for nonlinear (constraint-coupled) problems. The implementation of real-time distributed MPC schemes on robotic platforms is studied in [220], for constraint-coupled optimal control problems.

Noticeably, the problem structure and the way the decisions variables are coupled play a pivotal role in the algorithm development. Indeed, the proposed solution addresses multi-agent optimal control problems where the problem structure reflects the aggregative optimization framework. This optimization setting takes inspiration from the recently emerged distributed aggregative optimization framework introduced in [146] for a static, unconstrained case. In this context, differently from the aggregative games [20, 55, 84, 99, 175], the network agents cooperate (rather than compete with each other) to minimize the sum of local functions that depend both on a local decision variable and an aggregation of them. It is worth noticing that the aggregative optimization setting is a general distributed framework in which agents do not necessarily need to do consensus (as in cost-coupled problems). This makes the aggregative setting very versatile, especially for robotic applications. The recent survey [223] offers an overview of popular tasks arising in cooperative robotics (e.g., surveillance and resource allocation) that can be cast in this distributed aggregative optimization framework. Moreover, applications such as smart grids management, economic market analysis, electric vehicles charging, and network congestion control are modeled in the literature as aggregative games [21, 122, 176], i.e., a network of players that compete with each other to minimize individual costs depending on both local quantities and an aggregation of all of them. Therefore, problem (3.43) allows for reformulating these tasks in cooperative scenarios. We also remark that, differently from problem (3.43), the mentioned references [21, 122, 176, 223] deal with dynamic systems (e.g., robots or energy generators) without optimizing or considering their low-level control. An online version of the problem is addressed in [54, 145], while in [57] a learning-based distributed algorithm is proposed to deal with partially unknown local objective functions. In [56, 60], the aggregative framework is investigated in a continuous-time setup. Distributed algorithms based on a Franke-Wolfe update [237] and on ADMM [105] are proposed for a static, constrained setup. In [62] the setup with finite bits for communication among the agents is addressed. We remark that, differently from the above works, our solution extends the aggregative optimization framework to the optimal control scenario, hence addressing the task of generating a trajectory for a nonlinear dynamical system while optimizing a cost functional involving an aggregate variable.

Literature on Learning-driven Optimal Control

Classic system identification methods adopt parametric models and exploit observation to tune their parameters to achieve model accuracy [63, 153]. In view of the recent success in the field of machine learning, data-driven control techniques have gained growing interest in the control system community [117, 196]. Different data-driven approaches have been considered, e.g., reinforcement learning [106], model fitting [12, 75] and stochastic nonparametric estimation [152]. Gaussian Processes (GP) and their associated nonparametric regression techniques are well recognized in the field of controls for its flexibility in modelling complex unknown dynamics [138]. GPs have also been recently exploited in the quantification of model uncertainty [245]. In the field of optimal control, GPs have been exploited as a valid alternative to the prominent approach of modeling uncertainty as a stochastic disturbance. In [76] GPs are used in a dynamic programming framework. In [229], a scenario-based optimal control strategy is proposed based on a GP approximation of the dynamics. In [121] a combined Bayesian approach and GPs is proposed to select the most informative data for optimally updating a nominal model. Many interesting applications can be found also in the field of adaptive control [18, 24, 230]. GPs have been also successfully applied in robotics [167, 168], aircraft control [65] and power demand management [166]. Finally, also Model Predictive Control (MPC) schemes based on GP have been investigated [36, 94, 194]. Among the other works, we refer to [116] where a MPC approach that integrates a nominal system with an additive unknown dynamics modeled as a GP is exploited. The strategy is then extended to autonomous racing driving in [129]. While GPs have been successfully applied, most approaches lack formal guarantees for GP models. Recently, some control approaches with formal guarantees have been developed in [19, 86]. Bounds on the estimation error, in particular, are deeply analyzed in the field of Bayesian optimization [140, 218]. A theoretical analysis of the GP-based MPC controllers is presented in [157].

Literature on Stochastic Optimal Control

The literature on solving stochastic optimal control problems encompasses various approaches, each tailored to specific problem settings. Dynamic Programming (DP) serves as one of the main approaches to solving such problems [25]. DP, however, offers tractable solutions only in cases characterized by finite state and control spaces, linear dynamics, and convex quadratic cost functions. Consequently, several strategies aim to reformulate optimal control problems to fit within these frameworks. In the work [224], stochastic differential dynamic programming is introduced, employing DP principles along with stochastic dynamics featuring multiplicative noise. The approach involves second-order expansions and discretization around a given trajectory, facilitating the calculation of the expected value due to the presence of zero-mean multiplicative Gaussian noise. In [225] an iterative Linear Quadratic Gaussian (LQG) methods is presented. By linearizing dynamics and quadratizing the cost around a trajectory, an affine cost-to-go function is obtained. The concept of addressing stochastic control with affine dynamics is extended in [15] , encompassing finite switching modes, stochastic disturbances, and random switching. The synthesis of data-based controllers in the presence of stochastic disturbances, is described in [92]. The control policies are parametrized as a feedback gain with a feedforward term. The approach employs stochastic gradient descent and iteratively improves the system model for control synthesis.

3.2 GoPRONTO: Generalized first-Order PROjectioN operator method for Trajectory Optimization

In this section, we introduce the GoPRONTO optimization framework for general nonlinear optimal control problems.

We start by considering a nonlinear, discrete-time systems described by

$$x_{t+1} = f(x_t, u_t) \qquad t \in \mathbb{N} \tag{3.1}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the state and the input of the system at time t, respectively. The map $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the vector field describing the nonlinear dynamics. The initial condition of the system is a fixed state $x_{\text{init}} \in \mathbb{R}^n$. For notational convenience, we use $\boldsymbol{x} \in \mathbb{R}^{nT}$ and $\boldsymbol{u} \in \mathbb{R}^{mT}$ to denote, respectively, the stack of the states x_t for all $t \in [1, T]$ and the inputs u_t for all $t \in [0, T-1]$, that is $\boldsymbol{x} := \text{col}(x_1, \dots, x_T)$ and $\boldsymbol{u} := \text{col}(u_0, \dots, u_{T-1})$.

As already introduced in Section 1.2, our objective is to generate a trajectory $(\boldsymbol{x}, \boldsymbol{u})$ for system (3.1) such that an user defined cost functional is minimized. Before explicitly stating the problem setup, we need to formalize the concept of trajectory.

Definition 3.1 (Trajectory). *A pair $(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ is called a* trajectory *of the system described by* (3.1) *if its components satisfy the constraint represented by the dynamics* (3.1) *for all $t \in [0, T-1]$. In particular, $\boldsymbol{x}$ is the state trajectory, while $\boldsymbol{u}$ is the input trajectory.*

More formally, we define the set $\mathcal{T}_{x_{\text{init}}} \subseteq \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ such that

$$\mathcal{T}_{x_{\text{init}}} := \left\{ (\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT} \mid x_0 = x_{\text{init}}, \;\; x_{t+1} = f(x_t, u_t) \qquad \forall t \in [0, T] \right\} \tag{3.2}$$

as the space of trajectories of system (3.1) *with initial condition $x_{\text{init}} \in \mathbb{R}^n$.* △

Conversely, we refer to a generic pair $(\boldsymbol{\alpha}, \boldsymbol{\mu}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ with $\boldsymbol{\alpha} := \text{col}(\alpha_1, \ldots, \alpha_T)$ and $\boldsymbol{\mu} := \text{col}(\mu_0, \ldots, \mu_{T-1})$ as a state-input *curve*. Notice that a curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ is not necessarily a trajectory, i.e., it does not necessarily satisfy the dynamics (3.1). It can be shown that the tangent space to the space of trajectories (3.2) at a given trajectory (point), denoted as $T_{(\boldsymbol{x},\boldsymbol{u})}\mathcal{T}$, is represented by the set of trajectories satisfying the linearization of the nonlinear dynamics $f(\cdot, \cdot)$ about the trajectory $(\boldsymbol{x}, \boldsymbol{u})$.

Formally, our goal is to compute a state-input sequence $(\boldsymbol{x}, \boldsymbol{u})$ (i) satisfying Definition 3.1, (ii) solution of the the optimal control problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^{nT}, \boldsymbol{u} \in \mathbb{R}^{mT}} \; \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{3.3a}$$

$$\text{subj.to } \; x_{t+1} = f(x_t, u_t), \quad t \in [0, T-1] \tag{3.3b}$$

with initial condition $x_0 = x_{\text{init}} \in \mathbb{R}^n$, stage cost $\ell_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and terminal cost $\ell_T : \mathbb{R}^n \to \mathbb{R}$.

Assumption 3.1. *All functions $\ell_t(\cdot, \cdot)$, $\ell_T(\cdot)$ and $f(\cdot, \cdot)$ are twice continuously differentiable, i.e., they are of class $\mathcal{C}^2$ with respect to their arguments.* △

Problem (3.3) can be compactly reformulated by rewriting the nonlinear dynamics (3.3b) as an implicit equality constraint $h : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \to \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ given by

$$h(\boldsymbol{x}, \boldsymbol{u}) := \begin{bmatrix} f(x_0, u_0) - x_1 \\ \vdots \\ f(x_{T-1}, u_{T-1}) - x_T \end{bmatrix}. \tag{3.4}$$

and by compactly defining the cost function (3.3a) as

$$\ell(\boldsymbol{x}, \boldsymbol{u}) := \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T). \tag{3.5}$$

Therefore, problem (3.3) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^{nT}, \boldsymbol{u} \in \mathbb{R}^{mT}} \;& \ell(\boldsymbol{x}, \boldsymbol{u}) \\ \text{subj.to } \;& h(\boldsymbol{x}, \boldsymbol{u}) = 0 \end{aligned} \qquad \text{or} \qquad \min_{(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{T}} \; \ell(\boldsymbol{x}, \boldsymbol{u})$$

It is worth noting that, in light of the nonlinear equality constraint $h(\boldsymbol{x}, \boldsymbol{u}) = 0$ of the nonlinear dynamics, problem (3.3) is a nonconvex program. Figure 3.1 provides a graphical representation of the optimal control problem as a nonlinear (nonconvex) program.

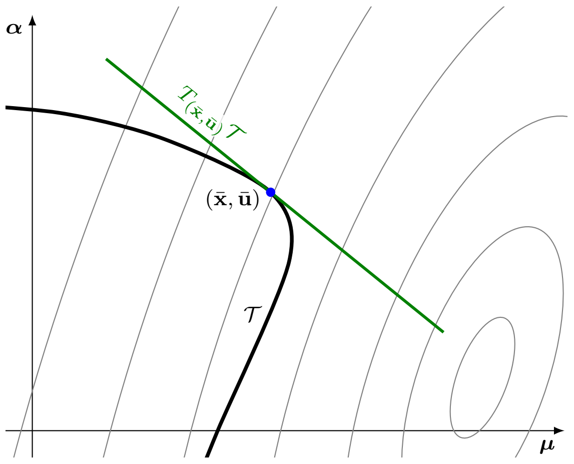

$\boldsymbol{\alpha}$
$T_{(\bar{\mathbf{x}},\bar{\mathbf{u}})}\mathcal{T}$
$(\bar{\mathbf{x}},\bar{\mathbf{u}})$
$\mathcal{T}$
$\boldsymbol{\mu}$

Figure 3.1: Two dimensional representation of the optimal control problem: in gray the level curves of the cost function $\ell(\cdot,\cdot)$, in black the nonlinear constraint representing the trajectory manifold $\mathcal{T}$, in green its tangent space $T_{(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}})}\mathcal{T}$ about $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$.

The Feedback Embedding Paradigm

The Feedback Embedding approach draws inspiration by the *Projection Operator* method, firstly introduced in continuous time by [113]. The underlying idea is to design a feedback policy mapping generic elements $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ of the space $\mathbb{R}^{nT} \times \mathbb{R}^{mT}$, the so-called *curves*, into the space of trajectories $\mathcal{T}_{x_{\text{init}}}$ feasible for the dynamics (3.3b). Among the different possibilities, the feedback policy is implemented, for all $t \in \{1, \ldots, T-1\}$, via the nonlinear tracking system

$$\begin{aligned} u_t &= \mu_t + K_t(\alpha_t - x_t) \\ x_{t+1} &= f(x_t, u_t), \end{aligned} \tag{3.6}$$

where $K_t \in \mathbb{R}^n \times \mathbb{R}^m$ is a suitably chosen feedback which is stabilizing for the dynamics $f(\cdot,\cdot)$, e.g., a linear quadratic regulator involving the system linearization about the current trajectory $(\boldsymbol{x}, \boldsymbol{u})$. As it become clearer later, the feedback term $K_t(\alpha_t - x_t)$ is fundamental to guarantee numerical stability of the algorithm.

Remark 3.1. The feedback gain K_t in (3.6) should ensure local stability about the current state-input trajectory. Among different alternatives, a viable strategy could be leveraging a linear quadratic regulator involving the system linearization about the current trajectory. Also, one can use more advanced design approaches, e.g., linear parameter varying controllers [216]. $\triangle$

Notice that, the feedback system (3.6) implements a nonlinear projection operator $\mathcal{P} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \to \mathcal{T}_{x_{\text{init}}}$ which maps state-input curves into trajectories of the nonlinear system. More formally, $\mathcal{P}(\boldsymbol{\alpha}, \boldsymbol{\mu})$ is designed such that, for any $(\boldsymbol{\alpha}, \boldsymbol{\mu}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ and $(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{T}_{x_{\text{init}}}$ it holds

$$\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\mu} \end{bmatrix} \longmapsto \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{bmatrix} := \mathcal{P}(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \begin{bmatrix} \phi(\boldsymbol{\alpha}, \boldsymbol{\mu}) \\ \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}) \end{bmatrix}, \tag{3.7}$$

where $\phi(\boldsymbol{\alpha}, \boldsymbol{\mu})$ and $\psi(\boldsymbol{\alpha}, \boldsymbol{\mu})$ are the state and input components of $\mathcal{P}(\boldsymbol{\alpha}, \boldsymbol{\mu})$. For all $t \in [0, T-1]$, we can also define the maps

$$x_t = \phi_t(\boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{3.8a}$$

$$u_t = \psi_t(\boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{3.8b}$$

The closed-loop policy (3.6) can be embedded as a redundant constraint in problem (3.3), thus resulting into the following optimal control problem

$$\min_{\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}} \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{3.9a}$$

$$\begin{aligned} \text{subj.to } & x_{t+1} = f(x_t, u_t) \qquad t = 0, \ldots, T-1 \\ & u_t = \mu_t + K_t(\alpha_t - x_t) \\ & x_0 = x_{\text{init}}. \end{aligned} \tag{3.9b}$$

Remark 3.2. The independent decision variables of problem (3.9) are the curves $(\boldsymbol{\alpha}, \boldsymbol{\mu}_i$. Indeed, once the curves $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ are computed, the variables $(\boldsymbol{x}, \boldsymbol{u})$ are uniquely determined by means of the (projection) policy (3.6). $\triangle$

We can recast problem (3.9) in its reduced form by expressing both the state x_t and the input u_t as functions of a state-input curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ via (3.8). Namely, we obtain a reduced instance of problem (3.3) given by

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\mu}} \sum_{t=0}^{T-1} \ell_t(\phi_t(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi_t(\boldsymbol{\alpha}, \boldsymbol{\mu})) + \ell_T(\phi_T(\boldsymbol{\alpha}, \boldsymbol{\mu})) = \min_{\boldsymbol{\alpha}, \boldsymbol{\mu}} J(\boldsymbol{\alpha}, \boldsymbol{\mu}). \tag{3.10}$$

Importantly, problem (3.10) is an unconstrained optimization problem in $(\boldsymbol{\alpha}, \boldsymbol{\mu})$.

3.2.1 Feedback Embedding Paradigm for Nonlinear Optimal Control: GoPRONTO Algorithm Description and Analysis

We are ready to present the general set of first-order approaches, called GoPRONTO, for numerical optimal control. We start by describing a pure gradient (or steepest) descent implementation which we call Gradient GoPRONTO.

The founding idea of GoPRONTO is to formulate and solve the reduced embedded feedback optimization problem via a first-order method, while, at the same time, taking also advantage from the beneficial effects of the embedded feedback control policy. Our embedded feedback approach is inspired by the first- and second-order approaches detailed in Appendix B, and presented in [27, 113].

The proposed procedure is summarized in Algorithm 6, where we use the shorthand notation

$$a_t^k := \nabla_1 \ell_t(x_t^k, u_t^k), \qquad b_t^k := \nabla_2 \ell_t(x_t^k, u_t^k), \tag{3.11a}$$

$$A_t^k := \nabla_1 f(x_t^k, u_t^k)^\top, \qquad B_t^k := \nabla_2 f(x_t^k, u_t^k)^\top. \tag{3.11b}$$

We also assume that, for all k, the state-input trajectory is initialized at $x_0^k = x_{\text{init}}$.

Algorithm 6 Gradient GoPRONTO

```
for $k = 0, 1, 2 \dots$ do
    set $\lambda_T^k = \nabla \ell_T(x_T^k)$
    for $t = T-1, \dots, 0$ do
        Step 1: compute descent direction
            $\lambda_t^k = (A_t^k - B_t^k K_t)^\top \lambda_{t+1}^k + a_t^k - K_t^\top b_t^k$        (3.12a)
            $\Delta\mu_t^k = -B_t^{k\top} \lambda_{t+1}^k - b_t^k$        (3.12b)
            $\Delta\alpha_t^k = K_t^\top \Delta\mu_t^k$        (3.12c)
    for $t = 0, \dots, T-1$ do
        Step 2: update (unfeasible) curve
            $\alpha_t^{k+1} = \alpha_t^k + \gamma^k \Delta\alpha_t^k$
            $\mu_t^{k+1} = \mu_t^k + \gamma^k \Delta\mu_t^k$        (3.13)
        Step 3: compute new (feasible) trajectory
            $u_t^{k+1} = \mu_t^{k+1} + K_t(\alpha_t^{k+1} - x_t^{k+1})$
            $x_{t+1}^{k+1} = f(x_t^{k+1}, u_t^{k+1})$        (3.14)
```

Algorithm 6 implements a numerically robust gradient descent method to solve problem (3.9) in the space of the curves by exploiting the beneficial effect of the feedback operator. This algorithm averages a gradient method to solve the reduced problem formulation (3.10) in the space of curves $(\boldsymbol{\alpha}, \boldsymbol{\mu})$. Specifically, at each iteration $k \in \mathbb{N}$, a descent direction is provided by computing $\Delta\alpha_t^k$ and $\Delta\mu_t^k$ through a backward integration of the closed-loop dynamics (3.12). This adjoint update can be seen as a closed-loop (robustified) version of the equation appearing in adjoint sensitivity methods. Then, in (3.13), such a direction is used to update the current curve. Finally, the new (updated) trajectory is obtained by applying the (feedback) policy onto the updated curve (3.14). Notice that, although GoPRONTO involves a forward-backward sweep typical of many Dynamic Programming-based approaches, it differs in the sense that (i) it does not seek to compute a Value Function and (ii) it retains the flexibility of the feedback gain design in (3.6). Moreover, the problem reformulation in (3.9) allows also for a perturbation of the state sequence $\boldsymbol{\alpha}$ to further improve the cost (cf. (3.13)).

A visual representation of this optimization problem is provided in Figure 3.2.

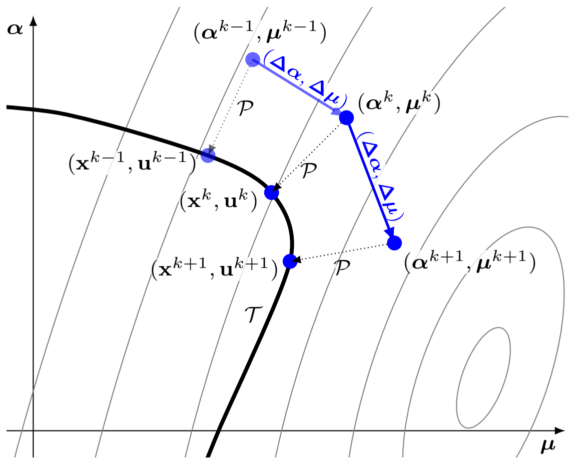

$\boldsymbol{\alpha}$
$(\boldsymbol{\alpha}^{k-1}, \boldsymbol{\mu}^{k-1})$
$(\Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\mu})$
$(\boldsymbol{\alpha}^{k}, \boldsymbol{\mu}^{k})$
$\mathcal{P}$
$(\mathbf{x}^{k-1}, \mathbf{u}^{k-1})$
$\mathcal{P}$
$(\Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\mu})$
$(\mathbf{x}^{k}, \mathbf{u}^{k})$
$(\mathbf{x}^{k+1}, \mathbf{u}^{k+1})$
$\mathcal{P}$
$(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})$
$\mathcal{T}$
$\boldsymbol{\mu}$

Figure 3.2: Representation of GoPRONTO approach: in gray the level curves of the reduced cost $J(\cdot,\cdot)$, in black the trajectory manifold $\mathcal{T}$, in blue the descent directions. At each iteration k, the current curve $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ is updated along the (generic) descent direction defined by the gradient of the reduced cost $J(\cdot,\cdot)$. The updated curve $(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})$ is, then, projected onto the trajectory manifold $\mathcal{T}$ by the projection operator $\mathcal{P}$ (dotted line).

Algorithm analysis

In the following, we show how our strategy reads as a gradient descent method applied on (3.10) We start by noticing that the reduced embedded feedback optimization problem (3.10) is an unconstrained optimization problem with nonconvex, twice continuously differentiable cost function $J(\cdot,\cdot)$ (obtained as the composition of $\mathcal{C}^2$ functions). Therefore, we apply the gradient method in which the tentative solution $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ is iteratively refined as

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k - \gamma^k \nabla_1 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k), \qquad \boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k - \gamma^k \nabla_2 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k) \tag{3.15}$$

where $k > 0$ is the iteration index while γ^k is the step-size. We can see that the descent direction is searched in the entire space of curves $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ Moreover, the update-direction search is not restricted to any tangent space.

The update (3.15) can be expressed also in a component-wise fashion as

$$\alpha_t^{k+1} = \alpha_t^k - \gamma^k \underbrace{\left[\nabla_1 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t}_{-\Delta\alpha_t^k} \tag{3.16a}$$

$$\mu_t^{k+1} = \mu_t^k - \gamma^k \underbrace{\left[\nabla_2 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t}_{-\Delta\mu_t^k} \tag{3.16b}$$

for all $t \in [0, T-1]$, in which each pair $(\Delta\alpha_t^k, \Delta\mu_t^k) \in \mathbb{R}^n \times \mathbb{R}^m$ represents the descent direction in (3.12). As it can be seen in Figure 3.2, each (updated) state-input curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ is then projected by the projection operator $\mathcal{P}$ onto the trajectory manifold $\mathcal{T}$ as per (3.14).

Before providing the convergence result for Algorithm 6, let us make an assumption on the step-size.

Assumption 3.2. *Let the step-size $\gamma^k \in \mathbb{R}$, $\gamma^k > 0$ be chosen via Armijo backtracking line search.* $\triangle$

The following theorem holds true.

Theorem 3.1. *Let Assumptions 3.1 and 3.2 hold. Let $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}_{k\geq 0}$ be the sequence generated by Algorithm 6. Every limit point $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ of the sequence $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}_{k\geq 0}$ satisfies $\nabla J(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star) = 0$. Moreover, let $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ be the trajectory associated to state-input curve $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ and $\boldsymbol{\lambda}^*$ the associated costate trajectory generated by Algorithm 6 in correspondence of $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$. Then, $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ represents a trajectory satisfying the first order necessary conditions for optimality in correspondence of costate trajectory $\boldsymbol{\lambda}^*$.* $\triangle$

Proof. The proof is arranged in two main parts. In the first part, we prove that any limit point $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ of the sequence $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}_{k\geq 0}$ generated by Algorithm 6 is a stationary point of the unconstrained problem (3.10), i.e., it satisfies $\nabla J(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star) = 0$. Specifically, we show that Algorithm 6 represents a gradient descent method applied to problem (3.10).

Let us prove that the descent direction computed in (3.12) is the gradient of $J(\cdot, \cdot)$ evaluated at the point $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$. To this end let us express the nonlinear dynamics in (3.9) as an implicit equality constraint $\tilde{h} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{nT} \times \mathbb{R}^{mT} \to \mathbb{R}^{nT+mT}$ defined as

$$\tilde{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}) := \begin{bmatrix} f(x_0, u_0) - x_1 \\ \vdots \\ f(x_{T-1}, u_{T-1}) - x_T \\ \mu_0 + K_0(\alpha_0 - x_0) - u_0 \\ \vdots \\ \mu_{T-1} + K_{T-1}(\alpha_{T-1} - x_{T-1}) - u_{T-1} \end{bmatrix}. \tag{3.17}$$

Therefore, by means of (3.5), we can compactly recast problem (3.9) as

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}} \ & \ell(\boldsymbol{x}, \boldsymbol{u}) \\ \text{subj.to} \ & \tilde{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = 0. \end{aligned} \tag{3.18}$$

Then we can introduce an auxiliary function[2] associated to problem (3.18), say $\mathcal{L} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times R^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{nT} \to \mathbb{R}$, defined as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda}) := \ell(\boldsymbol{x}, \boldsymbol{u}) + \tilde{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu})^\top \boldsymbol{\lambda} \tag{3.19}$$

where the (multiplier) vector $\boldsymbol{\lambda} \in \mathbb{R}^{nT+mT}$ is arranged as

$$\boldsymbol{\lambda} := \text{col}(\lambda_1, \ldots, \lambda_T, \tilde{\lambda}_1, \ldots, \tilde{\lambda}_T)$$

with each $\lambda_t \in \mathbb{R}^n$ and $\tilde{\lambda}_t \in \mathbb{R}^m$. By defining $\phi(\cdot)$ and $\psi(\cdot)$ as the vertical stack of the maps $\phi_t(\cdot)$ and $\psi_t(\cdot)$ (Cf. (3.8)), we can see that, by construction, for all $(\boldsymbol{\alpha}, \boldsymbol{\mu}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ it holds

$$\tilde{h}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}) = 0. \tag{3.20}$$

Since $J(\boldsymbol{\alpha}, \boldsymbol{\mu}) \equiv \ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}.\boldsymbol{\mu}))$ (Cf. (3.10) and (3.5)), the auxiliary function (3.19) enjoys the following property

$$\mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = J(\boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{3.21}$$

for all $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ and *for all* $\boldsymbol{\lambda} \in \mathbb{R}^{nT+mT}$. Therefore, in this formulation we can think about $\boldsymbol{\lambda}$ as a parameter or a degree of freedom. As a consequence of (3.21), it also results

$$\nabla \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \nabla J(\boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{3.22}$$

[2] It is evidently the Lagrangian function of problem (3.18). However, since we do not pursue a Lagrangian approach, we prefer not to use such nomenclature.

for all $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ and, again, *for all* $\boldsymbol{\lambda}$, where the gradient of $\mathcal{L}(\cdot)$ is meant to be calculated only with respect to $(\boldsymbol{\alpha}, \boldsymbol{\mu})$.

In the following, we exploit (3.22) together with the degree of freedom represented by $\boldsymbol{\lambda}$ in order to efficiently compute $\nabla J(\cdot, \cdot)$. In fact, we can write the two components of the gradient of $J(\cdot, \cdot)$ as

$$\begin{aligned}\nabla_1 J(\boldsymbol{\alpha}, \boldsymbol{\mu}) &= \nabla_1 \phi(\boldsymbol{\alpha}, \boldsymbol{\mu}) \underline{\nabla_1 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda})} \\ &\quad + \nabla_1 \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}) \underline{\nabla_2 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda})} \\ &\quad + \nabla_3 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda})\end{aligned}$$

and

$$\begin{aligned}\nabla_2 J(\boldsymbol{\alpha}, \boldsymbol{\mu}) &= \nabla_2 \phi(\boldsymbol{\alpha}, \boldsymbol{\mu}) \underline{\nabla_1 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda})} \\ &\quad + \nabla_2 \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}) \underline{\nabla_2 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda})} \\ &\quad + \nabla_4 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda}).\end{aligned}$$

Both these expressions involve the calculation of $\nabla \phi(\cdot)$ and $\nabla \psi(\cdot)$ which may be difficult to compute. However, since (3.22) holds for any $\boldsymbol{\lambda}$, we set this degree of freedom to greatly simplify the previous formulas. In fact, the underlined terms $\nabla_{\boldsymbol{x}} \mathcal{L}(\cdot)$ and $\nabla_{\boldsymbol{u}} \mathcal{L}(\cdot)$ have the following peculiar structure

$$\nabla_1 \mathcal{L}(\cdot) = \nabla_1 \ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu})) + \nabla_1 \tilde{h}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu})^\top \boldsymbol{\lambda} \tag{3.24a}$$

and

$$\nabla_2 \mathcal{L}(\cdot) = \nabla_2 \ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu})) + \nabla_2 \tilde{h}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu})^\top \boldsymbol{\lambda}. \tag{3.24b}$$

Therefore, with a proper choice of $\boldsymbol{\lambda}$ we can annihilate (3.24). In fact, by choosing $\boldsymbol{\lambda} = \bar{\boldsymbol{\lambda}}$ such that

$$\nabla_1 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{\lambda}}) = 0, \qquad \nabla_2 \mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{\lambda}}) = 0$$

i.e., by setting

$$\nabla_1 \ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu})) + \nabla_1 \tilde{h}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu})^\top \bar{\boldsymbol{\lambda}} = 0 \tag{3.25a}$$

$$\nabla_2 \ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu})) + \nabla_2 \tilde{h}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu})^\top \bar{\boldsymbol{\lambda}} = 0, \tag{3.25b}$$

both the terms involving $\nabla \phi(\cdot)$ and $\nabla \psi(\cdot)$ cancel out. Hence, the gradient components of $J(\cdot,\cdot)$ reduces to

$$\begin{aligned}\nabla_1 J(\boldsymbol{\alpha},\boldsymbol{\mu}) &= \nabla_3 \mathcal{L}(\phi(\boldsymbol{\alpha},\boldsymbol{\mu}),\psi(\boldsymbol{\alpha},\boldsymbol{\mu}),\boldsymbol{\alpha},\boldsymbol{\mu},\bar{\boldsymbol{\lambda}})\\ \nabla_2 J(\boldsymbol{\alpha},\boldsymbol{\mu}) &= \nabla_4 \mathcal{L}(\phi(\boldsymbol{\alpha},\boldsymbol{\mu}),\psi(\boldsymbol{\alpha},\boldsymbol{\mu}),\boldsymbol{\alpha},\boldsymbol{\mu},\bar{\boldsymbol{\lambda}}).\end{aligned}$$

By using again the definition of $\mathcal{L}(\cdot)$, the latter terms can be written as

$$\begin{aligned}\nabla_1 J(\boldsymbol{\alpha},\boldsymbol{\mu}) &= \nabla_3 \tilde{h}(\phi(\boldsymbol{\alpha},\boldsymbol{\mu}),\psi(\boldsymbol{\alpha},\boldsymbol{\mu}),\boldsymbol{\alpha},\boldsymbol{\mu})^\top \bar{\boldsymbol{\lambda}}\\ \nabla_2 J(\boldsymbol{\alpha},\boldsymbol{\mu}) &= \nabla_4 \tilde{h}(\phi(\boldsymbol{\alpha},\boldsymbol{\mu}),\psi(\boldsymbol{\alpha},\boldsymbol{\mu}),\boldsymbol{\alpha},\boldsymbol{\mu})^\top \bar{\boldsymbol{\lambda}}.\end{aligned} \tag{3.26}$$

With this derivation at reach, let us now focus on the k-th iteration of Algorithm 6. In correspondence of the current state-input curve $(\boldsymbol{\alpha}^k,\boldsymbol{\mu}^k)$, which represents a tentative solution of problem (3.10), we can compute the vector

$$\boldsymbol{\lambda}^k := \mathrm{col}(\lambda_1^k,\ldots,\lambda_T^k,\tilde{\lambda}_1^k,\ldots,\tilde{\lambda}_T^k)$$

such that (3.25) holds with $(\boldsymbol{\alpha},\boldsymbol{\mu}) = (\boldsymbol{\alpha}^k,\boldsymbol{\mu}^k)$ and $\bar{\boldsymbol{\lambda}} = \boldsymbol{\lambda}^k$. Therefore, by recalling the definitions of $\ell(\cdot)$ and $\tilde{h}(\cdot)$ in (3.5) and (3.17) and since the functions $f(\cdot),\ell_t(\cdot),\ell_T(\cdot)$ are differentiable by Assumption 3.1, the components λ_t^k of $\boldsymbol{\lambda}^k$ need to satisfy

$$\nabla \ell_T(\phi_T(\boldsymbol{\alpha}^k,\boldsymbol{\mu}^k)) - \lambda_T^k = 0$$

and, for all $t \in [0, T-1]$,

$$a_t^k + A_t^{k\top}\lambda_{t+1}^k - \lambda_t^k - K_t^\top \tilde{\lambda}_t^k = 0$$

which descends from (3.25a). As for the components $\tilde{\lambda}_t^k$ of $\boldsymbol{\lambda}^k$, they needs to be such that for all $t \in [0, T-1]$

$$b_t^k + B_t^{k\top}\lambda_{t+1}^k - \tilde{\lambda}_t^k = 0$$

which comes from (3.25b). More compactly, a vector $\boldsymbol{\lambda}^k \in \mathbb{R}^{nT+mT}$ such that for $\bar{\boldsymbol{\lambda}} = \boldsymbol{\lambda}^k$ (3.25) is satisfied for a given $(\boldsymbol{\alpha}^k,\boldsymbol{\mu}^k)$, can be obtained by backward simulation of the adjoint system dynamics

$$\lambda_t^k = (A_t^k - B_t^k K_t)^\top \lambda_{t+1}^k + a_t^k - K_t^\top b_t^k \qquad \tilde{\lambda}_t^k = b_t^k + B_t^{k\top}\lambda_{t+1}^k \tag{3.28}$$

with terminal condition $\lambda_T^k = \nabla \ell_T(\phi_T(\boldsymbol{\alpha}^k,\boldsymbol{\mu}^k))$. With a suitable $\boldsymbol{\lambda}^k$ at hand, we can now compute the gradient of $J(\cdot,\cdot)$ as in (3.26), with $(\boldsymbol{\alpha},\boldsymbol{\mu}) = (\boldsymbol{\alpha}^k,\boldsymbol{\mu}^k)$ and $\bar{\boldsymbol{\lambda}} = \boldsymbol{\lambda}^k$. Considering a generic time instant t and recalling the structure of $\tilde{h}(\cdot)$ in (3.17), we have

$$\left[\nabla_1 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t = K_t^\top \tilde{\lambda}_t^k = K_t^\top \left(b_t^k + B_t^{k\top} \lambda_{t+1}^k\right) \tag{3.29a}$$

and

$$\left[\nabla_2 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t = \tilde{\lambda}_t^k = b_t^k + B_t^{k\top} \lambda_{t+1}^k \tag{3.29b}$$

for all $t \in [0, T-1]$. Comparing (3.12) in Algorithm 6 with (3.29), we can see that $\Delta\alpha_t^k, \Delta\mu_t^k$ in (3.12) must satisfy

$$\Delta\alpha_t^k := -\left[\nabla_1 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t, \qquad \Delta\mu_t^k := -\left[\nabla_2 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t.$$

Therefore, we proved that Algorithm 6 tackles problem (3.10) via a gradient descent method. In light of Assumption 3.2 the step-size γ^k in (3.13) is selected according to the Armijo rule on the cost function $J(\boldsymbol{\alpha}, \boldsymbol{\mu})$. Therefore, we can conclude that every limit point $(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*)$ of $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}_{k\geq 0}$ is a stationary point of $J(\boldsymbol{\alpha}, \boldsymbol{\mu})$, i.e., $\nabla J(\boldsymbol{\alpha}^*, \boldsymbol{\mu}^*) = 0$. This completes the first part of the proof.

In the second part, we prove that the trajectory $(\boldsymbol{x}^\star, \boldsymbol{u}^\star) = (\phi(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star), \psi(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star))$ together with the costate vectors $\boldsymbol{\lambda}^* \in \mathbb{R}^{nT}$ generated by Algorithm 6 in correspondence of $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$, satisfies the first order necessary optimality conditions for the optimal control problem (3.3). To this end, let us introduce the Hamiltonian function of problem (3.3) given by

$$H_t(x_t, u_t, \lambda_{t+1}) := \ell_t(x_t, u_t) + f(x_t, u_t)^\top \lambda_{t+1}$$

and next we show that $\nabla_2 H_t(x_t^*, u_t^*, \lambda_{t+1}^*) = 0$

$$\nabla_2 H_t(x_t^*, u_t^*, \lambda_{t+1}^*) = 0$$

and

$$\lambda_t^* = \nabla_1 H_t(x_t^*, u_t^*, \lambda_{t+1}^*) \tag{3.30}$$

with terminal condition $\lambda_T^* = \nabla \ell_T(x_T^*)$.

In light of the projection-operator step (3.14), the point $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ satisfies the dynamics (3.3b) by construction, i.e., it is a trajectory.

Let us define the shorthand for the linearization of the cost and the dynamics about the trajectory $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$

$$a_t^* := \nabla_1 \ell_t(x_t^*, u_t^*), \qquad b_t^* := \nabla_2 \ell_t(x_t^*, u_t^*), \tag{3.31a}$$

$$A_t^* := \nabla_1 f(x_t^*, u_t^*)^\top, \qquad B_t^* := \nabla_2 f(x_t^*, u_t^*)^\top. \tag{3.31b}$$

Then, we can define $\boldsymbol{\lambda}^*$ as the stack of the costate vectors $\lambda_t^* \in \mathbb{R}^n$, obtained from the adjoint equation (3.12a) evaluated at $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$, i.e., for all $t \in [T-1, 0]$

$$\lambda_t^* = \left(A_t^* - B_t^* K_t^*\right)^\top \lambda_{t+1}^* + a_t^* - K_t^\top b_t^* \tag{3.32}$$

with terminal condition $\lambda_T^* = \nabla \ell_T(x_T^*)$. Equation (3.32) corresponds to the gradient with respect to x_t of the Hamiltonian evaluated along the trajectory $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$, i.e., the first order necessary condition for optimality (3.30) holds by construction.

Finally, with $\boldsymbol{\lambda}^*$ at hand, we can see that condition

$$\nabla_2 H_t(x_t^*, u_t^*, \lambda_{t+1}^*) = 0$$

can be written as

$$\nabla_2 H_t(x_t^*, u_t^*, \lambda_{t+1}^*) = b_t^* + B_t^{*\top} \lambda_{t+1}^* \tag{3.33}$$

which corresponds to $\Delta\mu_t^k$, the gradient of $J(\cdot,\cdot)$ in (3.29b) evaluated at $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$. In light of the first part of the proof, this term is equal to zero. Therefore, the first order necessary conditions for optimality are satisfied by the trajectory $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$, thus concluding the proof. ■

Remark 3.3. We point out that Theorem 3.1 can be extended with suitable assumptions to different step-size selection rules other than Armijo backtracking line-search, e.g., constant step-size and diminishing step-size. △

Comparison with the gradient method presented in [27]: GoPRONTO and the gradient method for optimal control (cf. Section B.2.1 in Appendix B.1) share the same idea of the resolution of the optimal control problem via a gradient method in which the derivatives are computed through a costate dynamics. However, the introduction of the projection operator implies two fundamental improvements for GoPRONTO. First, we highlight that GoPRONTO enjoys numerical stability thanks to the different structure of the costate dynamics (3.12a). In fact, while the dynamical system represented by (B.25a) is governed by the matrix A_t^k, in our algorithm the adjoint system is governed by the matrix $A_t^k - B_t^k K_t$, which for a proper choice of the gain matrices K_t represents a stabilized, time-varying system as the horizon length goes to infinity. Moreover, thanks to the projection operator, in GoPRONTO the input trajectory $\boldsymbol{u}^{k+1}$ implements a nonlinear tracking controller of the (updated) state-input curve $(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})$. Therefore, the trajectory update (3.14) is performed under a closed-loop strategy rather than in open loop as in (B.26), so that dynamical systems subject to instability issues can be taken into account.

Comparison with the PRONTO method presented in [113]: Both GoPRONTO and PRONTO (cf.Section B.2.2 in Appendix B) iteratively refine a state-input curve which is remapped, using the feedback embedding approach, into a state-input trajectory. An important difference relies on how these curves are calculated at each iteration. In PRONTO, see (B.32), the next state-input curve is obtained by perturbing the current trajectory with a descent direction obtained via an LQ problem. Therefore the direction is sought on the tangent space of the trajectory manifold at the current iterate. In GoPRONTO, instead, we proceed from curve to curve following the descent direction defined by the gradient of the reduced cost (cf. (3.16)) obtained through the adjoint system (3.12). Since no constraints are imposed on the descent direction and no LQ problems are solved to find the descent direction, a lower computation cost is in general required. As a final remark, we point out that our algorithmic framework GoPRONTO can be exploited as a globalization technique for Newton's type optimization methods, see [27, Section 1.4] for a discussion.

3.2.2 Enhanced Versions of GoPRONTO

In this section we show how the embedded feedback framework (thanks to the efficient computation of the gradient of the cost function in (3.10)) can be combined with accelerated gradient-based optimization techniques available in the literature. This produces accelerated versions of Algorithm 6 as described next.

Conjugate GoPRONTO: In the seminal paper [115], the Conjugate Gradient (CG) method is presented as an approach to solve symmetric, positive-definite linear systems. Nevertheless, many strategies were developed in the nonlinear system framework, as detailed in [110]. Details on its implementation are given in Appendix A.

The Conjugate GoPRONTO optimal control method is obtained by applying CG to problem (3.10). Let, for all $k > 0$ and for all $t \in [0, T-1]$,

$$\alpha_t^{k+1} = \alpha_t^k + \gamma^k \Delta\tilde{\alpha}_t^k$$
$$\mu_t^{k+1} = \mu_t^k + \gamma^k \Delta\tilde{\mu}_t^k$$

where γ^k is chosen via Armijo backtracking line search, and the descent directions $\Delta\tilde{\alpha}_t^k$ and $\Delta\tilde{\mu}_t^k$ are obtained according to the GC algorithm as

$$\Delta\tilde{\alpha}_t^k := \Delta\alpha_t^k + \rho_{\alpha_t}^k \Delta\tilde{\alpha}_t^{k-1} \tag{3.34a}$$
$$\Delta\tilde{\mu}_t^k := \Delta\mu_t^k + \rho_{\mu_t}^k \Delta\tilde{\mu}_t^{k-1} \tag{3.34b}$$

with $\rho^k_{\alpha_t}$ and $\rho^k_{\mu_t}$ defined as

$$\rho^k_{\alpha_t} := \frac{\Delta\alpha_t^{k\top}(\Delta\alpha_t^k - \Delta\alpha_t^{k-1})}{\|\Delta\alpha_t^{k-1}\|^2}, \qquad \rho^k_{\mu_t} := \frac{\Delta\mu_t^{k\top}(\Delta\mu_t^k - \Delta\mu_t^{k-1})}{\|\Delta\mu_t^{k-1}\|^2}. \tag{3.35a}$$

Recalling that

$$\Delta\alpha_t^k = -\big[\nabla_1 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\big]_t, \qquad \Delta\mu_t^k = -\big[\nabla_2 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\big]_t \tag{3.36a}$$

can be computed by means of (3.12), the procedure in Algorithm 7 is obtained.

Algorithm 7 Conjugate GoPRONTO

```
for $k = 0, 1, 2 \dots$ do
    for $t = T-1, \dots, 0$ do
        Step 1: compute descent direction $\Delta\alpha_t^k, \Delta\mu_t^k$ as in (3.12)
    for $t = 0, \dots, T-1$ do
        compute CG update parameters $\rho^k_{\alpha_t}, \rho^k_{\mu_t}$ as in (3.35)
        compute CG update direction:
            $\Delta\tilde{\alpha}_t^k = \Delta\alpha_t^k + \rho^k_{\alpha_t}\Delta\tilde{\alpha}_t^{k-1}$
            $\Delta\tilde{\mu}_t^k = \Delta\mu_t^k + \rho^k_{\mu_t}\Delta\tilde{\mu}_t^{k-1}$
        Step 2: update (unfeasible) curve
            $\alpha_t^{k+1} = \alpha_t^k + \gamma^k\Delta\tilde{\alpha}_t^k$
            $\mu_t^{k+1} = \mu_t^k + \gamma^k\Delta\tilde{\mu}_t^k$
        Step 3: compute new (feasible) trajectory via (3.14)
```

As expected, when implemented with the necessary cautions, e.g., restarting policies and conjugacy tests, this method exhibits a faster convergence rate with respect to its plain gradient counterpart, see Section 3.2.3 for implementative details.

Heavy-Ball GoPRONTO: The Heavy-Ball method is a two-step procedure for the resolution of unconstrained optimization problems. It improves the convergence rate with respect to the plain gradient descent.

The Heavy-Ball GoPRONTO optimal control method is obtained by applying the

Heavy-ball iteration to problem (3.10), i.e., for all $k > 0$ and for all $t \in [0, T-1]$, we have

$$\alpha_t^{k+1} = \alpha_t^k - \gamma^k \underbrace{\left[\nabla_1 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t}_{-\Delta\alpha_t^k} + \gamma_{\text{hb}}(\alpha_t^k - \alpha_t^{k-1})$$

$$\mu_t^{k+1} = \mu_t^k - \gamma^k \underbrace{\left[\nabla_2 J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\right]_t}_{-\Delta\mu_t^k} + \gamma_{\text{hb}}(\mu_t^k - \mu_t^{k-1}),$$

where $\gamma^k > 0$ and $\gamma_{\text{hb}} > 0$ are suitable step-sizes. The descent directions $\Delta\alpha_t^k, \Delta\mu_t^k$ are computed by means of the costate equation (3.12). The resulting procedure is recapped in Algorithm 8.

Algorithm 8 Heavy-Ball GoPRONTO

```
for $k = 0, 1, 2 \dots$ do
    for $t = T-1, \dots, 0$ do
        Step 1: compute descent direction $\Delta\alpha_t^k, \Delta\mu_t^k$ as in (3.12)
    for $t = 0, \dots, T-1$ do
        Step 2: update (unfeasible) curve
            $\alpha_t^{k+1} = \alpha_t^k + \gamma^k \Delta\alpha_t^k + \gamma_{\text{hb}}(\alpha_t^k - \alpha_t^{k-1})$
            $\mu_t^{k+1} = \mu_t^k + \gamma^k \Delta\mu_t^k + \gamma_{\text{hb}}(\mu_t^k - \mu_t^{k-1})$
        Step 3: compute new (feasible) trajectory via (3.14)
```

We point out that, although a faster convergence rate of the Heavy-Ball method (with respect to the plain gradient descent) is rigorously proved for convex problems only, the practical implementation of this approach within our methodology confirmed these expectations (see Section 3.2.3).

Nesterov's GoPRONTO: Nesterov's accelerated gradient represents an alternative momentum method for the resolution of unconstrained optimization problems and has a faster convergence rate than the plain gradient methods.

The Nesterov's GoPRONTO optimal control algorithm is obtained by applying Nesterov's iteration (cf. (A.3)) to problem (3.10). Let, for all $k > 0$ and for all $t \in [0, T-1]$,

$$\alpha_t^{k+1} = \tilde{\alpha}_t^k - \gamma^k \underbrace{\left[\nabla_1 J(\tilde{\boldsymbol{\alpha}}^k, \tilde{\boldsymbol{\mu}}^k)\right]_t}_{-\Delta\tilde{\alpha}_t^k} \qquad \mu_t^{k+1} = \tilde{\mu}_t^k - \gamma^k \underbrace{\left[\nabla_2 J(\tilde{\boldsymbol{\alpha}}^k, \tilde{\boldsymbol{\mu}}^k)\right]_t}_{-\Delta\tilde{\mu}_t^k}, \tag{3.37a}$$

where γ^k is the step-size, while $\tilde{\boldsymbol{\alpha}}^k$ and $\tilde{\boldsymbol{\mu}}^k$ are the stacks of $\tilde{\alpha}_t^k$ and $\tilde{\mu}_t^k$, which are respectively defined as

$$\tilde{\alpha}_t^k = \alpha_t^k + \tfrac{k}{k+3}(\alpha_t^k - \alpha_t^{k-1}) \qquad \tilde{\mu}_t^k = \mu_t^k + \tfrac{k}{k+3}(\mu_t^k - \mu_t^{k-1}). \tag{3.37b}$$

The procedure is summarized in Algorithm 9.

Algorithm 9 Nesterov's GoPRONTO

```
for $k = 0, 1, 2 \dots$ do
    for $t = T-1, \dots, 0$ do
        Step 1: compute descent direction
            $$\begin{aligned} \lambda_t^k &= \left(\tilde{A}_t^k - \tilde{B}_t^k K_t\right)^\top \lambda_{t+1}^k + \tilde{a}_t^k - K_t^\top \tilde{b}_t^k \\ \Delta\tilde{\mu}_t^k &= -\tilde{B}_t^{k\top} \lambda_{t+1}^k - \tilde{b}_t^k \\ \Delta\tilde{\alpha}_t^k &= K_t^\top \Delta\tilde{\mu}_t^k. \end{aligned} \tag{3.38}$$
    for $t = 0, \dots, T-1$ do
        compute $\tilde{\alpha}_t^k, \tilde{\mu}_t^k$ as in (3.37b)
        Step 2: update (unfeasible) curve
            $$\begin{aligned} \alpha_t^{k+1} &= \tilde{\alpha}_t^k + \gamma^k \Delta\tilde{\alpha}_t^k \\ \mu_t^{k+1} &= \tilde{\mu}_t^k + \gamma^k \Delta\tilde{\mu}_t^k \end{aligned}$$
        Step 3: compute new (feasible) trajectory via (3.14)
```

We point out that the descent direction $(\Delta\tilde{\alpha}_t^k, \Delta\tilde{\mu}_t^k)$ in Algorithm 9 is computed at the current auxiliary curve $(\tilde{\boldsymbol{\alpha}}^k, \tilde{\boldsymbol{\mu}}^k)$ rather than $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ (cf. (A.3b)). The immediate consequence is that the linearization considered when evaluating the adjoint equations is computed about the system trajectory associated to $(\tilde{\boldsymbol{\alpha}}^k, \tilde{\boldsymbol{\mu}}^k)$. In (3.38), in fact, the matrices $\tilde{A}_t^k, \tilde{B}_t^k, \tilde{a}_t^k, \tilde{b}_t^k$ are defined as

$$\begin{aligned} \tilde{a}_t^k &:= \nabla_1 \ell_t(\tilde{x}_t^k, \tilde{u}_t^k), & \tilde{b}_t^k &:= \nabla_2 \ell_t(\tilde{x}_t^k, \tilde{u}_t^k), \\ \tilde{A}_t^k &:= \nabla_1 f(\tilde{x}_t^k, \tilde{u}_t^k)^\top, & \tilde{B}_t^k &:= \nabla_2 f(\tilde{x}_t^k, \tilde{u}_t^k)^\top, \end{aligned}$$

with $(\tilde{x}_t^k, \tilde{u}_t^k) = (\phi_t(\tilde{\boldsymbol{\alpha}}^k, \tilde{\boldsymbol{\mu}}^k), \psi_t(\tilde{\boldsymbol{\alpha}}^k, \tilde{\boldsymbol{\mu}}^k))$ for all k and t.

3.2.3 Numerical Simulations

In this section, we give extensive, explanatory simulation of the algorithms proposed in the previous sections. First of all, we show the basic implementation of the feedback embedding framework, detailed in Algorithm 6 and, then, we propose comparisons

Table 3.1: Parameters of each pendulum-on-cart system.

Length of Pendulum	l	1.0 [m]
Mass of Pendulum	M_p	0.2 [kg]
Mass of Cart	M_c	6.0 [kg]
Damping of Pendulum	f_p	0.01 [$\frac{\mathrm{Nms}}{\mathrm{rad}}$]
Damping of Cart	f_c	10.0 [$\frac{\mathrm{Ns}}{\mathrm{m}}$]
Spring Constant	κ_s	0.5 [$\frac{\mathrm{N}}{\mathrm{m}}$]
Gravitational Acceleration	g	9.81 [$\frac{\mathrm{m}}{\mathrm{s}^2}$]

with the enhanced versions. We implement a trajectory generation task, i.e., we aim at defining the optimal trajectory while tracking a given reference curve. We consider a challenging, large-scale system made by a train of N inverted pendulums on carts as shown in Figure 3.3. Each cart is connected with its preceding and its subsequent by means of a spring, with the only exception of the extremal carts.

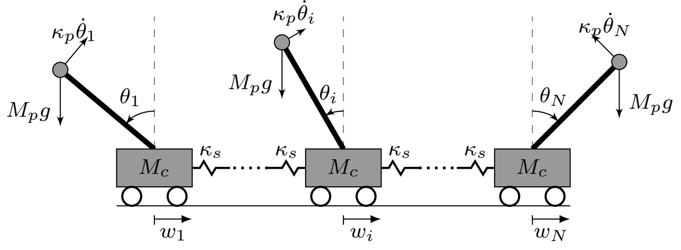


Figure 3.3: Scheme of the train of inverted pendulum-on-cart systems.

For each system $i \in \{1, \dots, 50\}$, the nonlinear dynamics is given by

$$\begin{aligned} M_p l^2 \ddot{\theta}_i + \mathrm{f}_p \dot{\theta}_i - M_p l \sin(\theta_i)\ddot{w} - M_p l g \sin(\theta_i) = 0 \\ (M_c + M_p)\ddot{w}_i + \mathrm{f}_c \dot{w}_i - \frac{1}{2} M_p l \cos(\theta)\ddot{\theta} + \\ + \frac{1}{2} M_p l \sin(\theta)\dot{\theta}^2 - \kappa_s w_{i+1} + \kappa_s w_{i-1} = u_i, \end{aligned}$$

where θ_i is the angle measured from the vertical upward position and w_i is the the lateral position of the cart. Each system is controlled through a force u_i applied to the cart. The parameters are identical for all carts and their values are reported in Table 3.1. The state space representation of the dynamics has states $x_i = (\theta_i, \dot{\theta}_i, w_i, \dot{w}_i)^\top \in \mathbb{R}^4$ with input $u_i \in \mathbb{R}$ for all i. Thus, the full state of the system is $x := \mathrm{col}(x_1, \dots, x_N) \in \mathbb{R}^{4\times N}$ and the full input is $u := \mathrm{col}(u_1, \dots, u_N) \in \mathbb{R}^N$. Then, we use a multiple step Runge-Kutta integrator of order 4 to obtain a discretized version of the plant given by $x_{t+1} = f(x_t, u_t)$ with sampling period $\delta = 0.05$ seconds. The sensitivities are computed by Algorithmic Differentiation. For the sake of compactness the discrete time state-space equations are omitted. This curve tracking problem has a quadratic cost function, where $\ell_t(x_t, u_t) = \|x_t - x_{\mathrm{ref},t}\|_Q^2 + \|u_t - u_{\mathrm{ref},t}\|_R^2$ and $\ell_T(x_T) = \|x_T - x_{\mathrm{ref},T}\|_{Q_f}^2$ with symmetric, positive-definite matrices $Q := \mathrm{diag}(Q_1, \dots, Q_N) \in \mathbb{R}^{4N\times 4N}$ and $R := \mathrm{diag}(R_1, \dots, R_N)$ where, for all $i = 1, \ldots, N$,

$$Q_i = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \qquad R_i = 10^{-1}.$$

The terminal cost matrix Q_f is, instead, defined as the solution of the (discrete-time) algebraic Riccati equation evaluated at the linearization of the system about the equilibrium. We aim at performing a swing manoeuver between $+30°$ and $-30°$ along a smooth curve which represents the reference signal for each θ_i. The desired angular velocity is determined differentiating the smooth curve for θ_i. Finally, the reference positions, velocities and inputs are set to zero. The feedback gain K_t in (3.6) is selected solving a LQR problem associated to the linearization of the nonlinear system about the trajectory $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ available at the current iteration with quadratic cost matrices defined as $Q_{\text{reg}} = Q$, $R_{\text{reg}} = I$ and $Q_{f,\text{reg}} = Q_f$.

The reference curve is defined, for each cart $i = 1, \ldots, N$, as

$$\theta_{\text{i,ref}}(t) = \frac{\theta_{\text{amp}}^{\text{rad}} \tanh(t - T/2)(1 - \tanh^2(t - T/2))}{\max\limits_{t \in [0,T]} \theta_{\text{amp}}^{\text{rad}} \tanh(t - T/2)(1 - \tanh^2(t - T/2))}$$

where $\theta_{\text{amp}}^{\text{rad}}$ represent the desired amplitude in radians. The desired angular velocity is determined by differentiating the smooth curve $\theta_{\text{i,ref}}(t)$. The other reference signals are zero.

We start considering $N = 50$ and performing a swing manoeuver between $\pm\theta_{\text{amp}}$, $\theta_{\text{amp}} = 30°$ along the smooth reference curve $\theta_{\text{i,ref}}(t)$ representing the angular reference signal for each θ_i. The state-input optimal trajectory together with some intermediate trajectories for the first cart-pole system are presented in Figure 3.4 while the reference signals are depicted in dashed green.The optimal trajectory for the angular position of some of the 50 carts is represented in Figure 3.5.

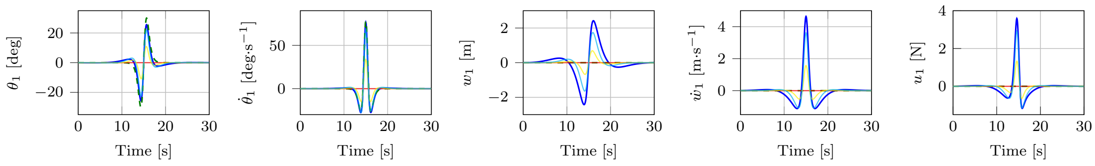


Figure 3.4: Optimal trajectory obtained one cart out of 50 via GoPRONTO. In blue the optimal trajectory, in dashed green the reference signals. In red, yellow and cyan the trajectory at iteration $k = 0, 2, 4$, respectively.

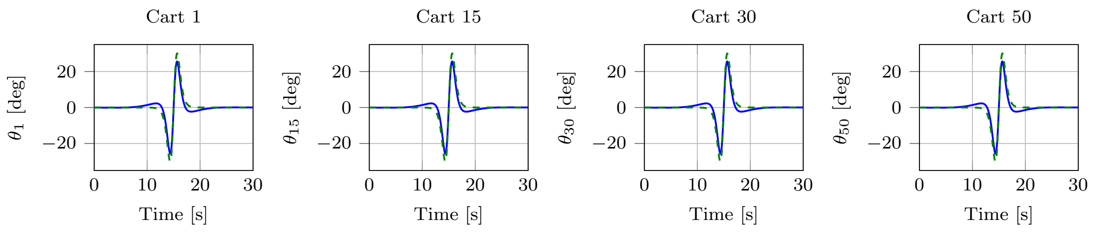


Figure 3.5: Optimal trajectory for state θ_i for carts $1, 15, 30, 50$ obtained via GoPRONTO. In blue the optimal trajectory, in dashed green the reference signal.

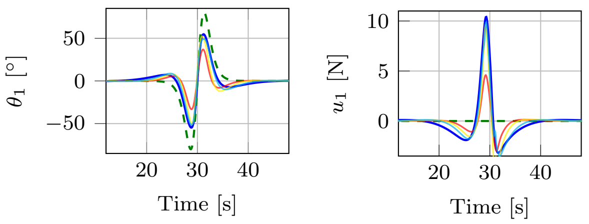


Figure 3.6: Optimal angle and input trajectory obtained for the first (of 100) pendulum-on-cart generated via GoPRONTO with $\theta_{\text{amp}} = 80°$. In blue the optimal trajectory, in dashed green the reference signals. In red, yellow and cyan the trajectories at iteration $k = 0, 2, 4$, respectively.

Comparison with existing numerical methods In this subsection, we compare GoPRONTO and the SQP-based algorithms for optimal control based on qpOASES, qpDUNES, HPIPM and the first-order OSQP, available in, e.g., acados [234]. For all algorithms, we consider the same numerical formulation of the optimal control problem and discretization scheme described above for $N = 3, 50, 100$ systems. Now, we aim at performing a swing maneuver between $\pm\theta_{\text{amp}}$, $\theta_{\text{amp}} \in \{60°, 80°\}$ along the smooth reference curve $\theta_{\text{i,ref}}(t)$. This scenario represents a challenging setting for numerical optimal control algorithms due to the large dimension of the decisions variables for the NLP. Moreover, for $\theta_{\text{amp}} = 80°$, the problem is even more challenging since it represents an asymptotically ill-posed problem. We chose this specific setting to provide an insight about the possibilities offered by GoPRONTO, which aims at representing a valid alterative in particular scenarios (e.g., quasi ill-posed problems and large-scale systems) where other approaches may face challenge. We remark the fact that GoPRONTO is implemented as in Algorithm 6 without pre-conditioning nor code optimization. The stepsize is selected via Armijo-line search and the shooting nodes coincide with the discretization time steps. The initial trajectory is $\theta_{i,t} \equiv 0, \dot{\theta}_{i,t} \equiv 0, w_{i,t} \equiv 0, \dot{w}_{i,t} \equiv 0$, for all t and i. The reference signals and the optimal trajectories obtained via GoPRONTO are shown in Figure 3.6 for the first of 100 pendulum-on-carts.

The full condensing versions of qpOASES and HPIPM fail to provide a solution for all N for both references. This can be due to the fact that full condensing approaches eliminate state variables via the unstable dynamics. The partial condensing version of HPIPM, in which only few state variables are eliminated, successfully solves the problem for $N = 3, 50$ for both reference signals. Being HPIPM a second-order solver a faster convergence rate than GoPRONTO is achieved. A comparable convergence rate is achieved only with $\theta_{\text{amp}} = 80°$ and $N = 50$. This can be due to the significant first-order acceleration required by the ill-posedness of the problem. For $N = 100$ HPIPM gives segmentation fault. As for partial condensing qpDUNES, it never reaches convergence. Finally, we compare GoPRONTO with the first-order solver OSQP. QSQP has a slower convergence rate for $N = 3$ and $\theta_{\text{amp}} = 80°$, while it has a faster convergence rate for $\theta_{\text{amp}} = 60°$. In the other cases it fails to provide a solution. Table 3.2 summarizes the performances achieved by the solvers that succeeded in at least one task.

Table 3.2: Iterations up to convergence of Gradient GoPRONTO, partial condensing HPIPM and OSQP ($\times$ means failure).

N	SQP solvers				GoPRONTO	
	Partial HPIPM		First-order OSQP			
θ_{amp}	60°	80°	60°	80°	60°	80°
3	8	9	14	50	30	34
50	14	43	$\times$	$\times$	35	39
100	$\times$	$\times$	$\times$	$\times$	40	42

Comparison with inspiring methods: Next, Gradient GoPRONTO is implemented on the previously presented setup with $N = 2$. In this case and in the following simulations, the swing maneuver is performed between $+30°$ and $-30°$. Algorithm 6 is compared with the Gradient Method (cf. Sec. B.2.1) and the first-order version of PRONTO (cf. Sec. B.2.2). The step-size γ^k is selected by Armijo line search rule. The evolution of the norm of the gradient $\nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, is presented in Figure 3.7. Notice that the Gradient Method diverges after very few iterations, while the first-order version of PRONTO exhibits a slower convergence rate compared with Algorithm 6.

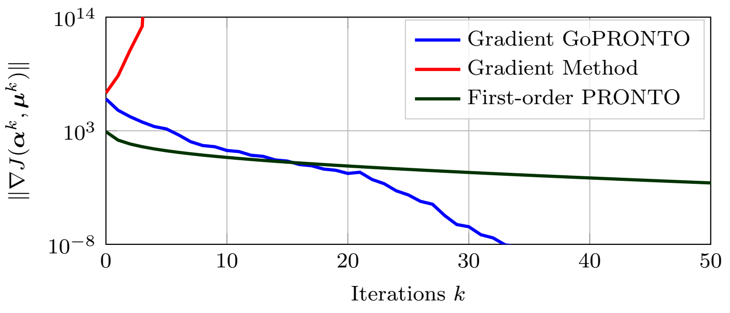


Figure 3.7: Evolution of the norm of the gradient $\|\nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\|$ in Gradient GoPRONTO, the Gradient Method and PRONTO.

Table 3.3 compares GoPRONTO and PRONTO in terms of the computation time required to compute the descent direction and the total computation time with $N = 2, 3, 5, 10$. Since PRONTO computes the descent direction by solving an LQ problem, as the state-input dimensions increases, larger computation time per iteration is required with respect to GoPRONTO. In GoPRONTO, the total computation time includes the time required to compute the projection gain, which is recomputed only when it looses its stabilizing property, (2-3 times per simulation). While the total computation time is higher with low state dimension, GoPRONTO shows a faster convergence with high state dimensions. Notice that, in PRONTO, one could use the feedback gain obtained by solving the LQ problem at the price of loosing the additional degree of freedom represented by the projection gain.

Table 3.3: Computation times per iteration in [s].

	Computation time per iteration [s]				
Number of carts N	2	5	10	50	100
PRONTO	0.12	0.24	0.79	40.86	260.29
GoPRONTO	0.05	0.08	0.18	6.73	33.13
	Total computation time [s]				
PRONTO	0.84	1.92	7.9	449.6	3123.48
GoPRONTO	1.76	3.36	9.28	328.63	1807.30

In the following, we compare Gradient GoPRONTO with its enhancements.

Comparison with Conjugate GoPRONTO: Here, $N = 2$ and the step-size γ^k is chosen via Armijo line search as required by the Conjugate Gradient method. Since the CG method is applied to a nonquadratic function, we need to deal with the resulting loss of conjugacy. The implemented method operates in cycles of conjugate direction steps, with the first step of each cycle being a basic gradient step. We choose to restart the policy when the conjugacy test fails, i.e. as soon as $|\nabla J(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})^\top \nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)| > 0.7\|\nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\|^2$. The evolution of the descent direction, i.e., the norm of the gradient $\nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, is presented in Figure 3.8. We can see that the descent direction decreases with a faster rate when the CG-enhanced version is adopted.

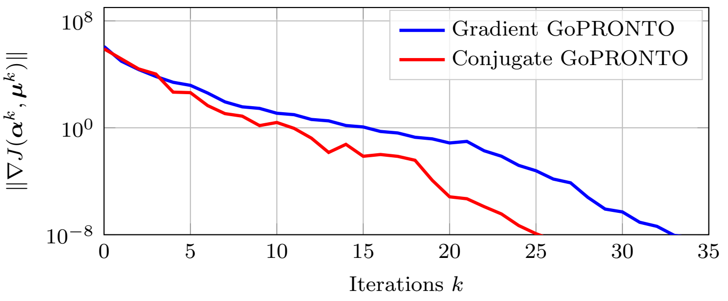


Figure 3.8: Evolution of $\|\nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\|$ in Gradient GoPRONTO and Conjugate GoPRONTO for $N = 2$. γ^k is chosen via Armijo line-search.

Comparison with Heavy-Ball GoPRONTO and Nesterov's GoPRONTO: Finally, we consider $N = 50$ resulting in $x_t \in \mathbb{R}^{200}$ and $u_t \in \mathbb{R}^{50}$. The step-size γ^k is fixed with

$\gamma^k \equiv \gamma = 10^{-3}$ while the Heavy-ball step $\gamma_{\text{hb}} = 0.5$. The evolution of the norm of the gradient $\nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, is presented in Figure 3.9. Notice that the enhanced versions of GoPRONTO present a faster convergence rate than its basic implementation.

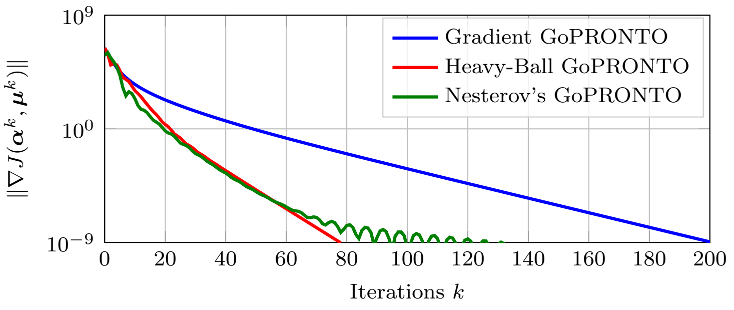


Figure 3.9: Evolution of $\|\nabla J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\|$ in Gradient GoPRONTO, Nesterov's GoPRONTO and Heavy-Ball GoPRONTO for $N = 50$. The stepsize is constant with $\gamma^k \equiv 10^{-3}$.

Constrained optimal control: In Figure 3.10 we present an example of optimal trajectory where input constraints are enforced. The problem setup is the same as above with $\theta_{\text{amp}} = 80°$. For all carts, i.e., for all $i = 1, \dots, N$, the maximum input is saturated at $u_{\max} = 5$ [N] $g(x_{i,t}) = \|u_{i,t}/u_{\max}\| - 1 \geq 0$, for all $t = [0, T]$ added to the optimal control problem (3.3). This constraint is enforced via the barrier function approach proposed in [114].

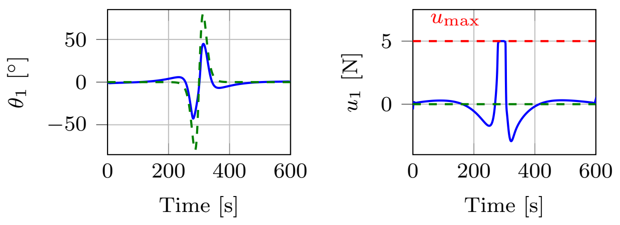


Figure 3.10: Constrained optimal angle and cart position trajectory for the first pendulum-on-cart. In blue the optimal trajectory, in dashed green the reference trajectory, in dashed red the bound on the control action.

3.3 Distributed Multi-agent Aggregative Optimal Control

In this section, we leverage on the Embedded Feedback Optimization framework, introduced in Section 3.2, to develop a distributed optimal control algorithm, tailored for multi-agent aggregative optimal control problems.

We start by formalizing the problem setup introduced in Section 1.2.1. We consider a network of N (possibly) heterogeneous agents evolving according to the nonlinear dynamics

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \tag{3.39}$$

for all $i \in \{1, \dots, N\}$, where $f_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^{m_i}$, $x_{i,t} \in \mathbb{R}^{n_i}$ is the state of agent i at time t, and $u_{i,t} \in \mathbb{R}^{m_i}$ is the control input applied to agent i at time t. The stacks of the states and inputs of all the agents at time t are denoted as

$$x_t := \text{col}(x_{1,t}, \dots, x_{N,t}) \in \mathbb{R}^n \quad u_t := \text{col}(u_{1,t}, \dots, u_{N,t}) \in \mathbb{R}^m,$$

where $n = \sum_{i=1}^N n_i$ and $m = \sum_{i=1}^N m_i$. Moreover, in parallel with the notation adopted in Section 3.2, considering a time horizon of length $T \in \mathbb{N}$, for each agent $i \in \{1, \dots, N\}$, we define as $\boldsymbol{x}_i \in \mathbb{R}^{n_i T}$ and $\boldsymbol{u}_i \in \mathbb{R}^{m_i T}$ the stack of the states $x_{i,1}, \dots, x_{i,T}$ and inputs $u_{i,0}, \dots, u_{i,T-1}$, respectively. For notational convenience, we also denote the stack of all $\boldsymbol{x}_i$, $\boldsymbol{u}_i$ as

$$\boldsymbol{x} := \text{col}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N) \in \mathbb{R}^{nT}, \qquad \boldsymbol{u} := \text{col}(\boldsymbol{u}_1, \dots, \boldsymbol{u}_N) \in \mathbb{R}^{mT}.$$

Then, we introduce the *aggregative* variables $\sigma_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$, for all $t \in [0, T-1]$, and $\sigma_T : \mathbb{R}^n \to \mathbb{R}^q$ that couple the states and inputs of all the agents according to

$$\sigma_t(x_t, u_t) = \frac{1}{N} \sum_{i=1}^N \varphi_{i,t}(x_{i,t}, u_{i,t}), \qquad \sigma_T(x_T) = \frac{1}{N} \sum_{i=1}^N \varphi_{i,T}(x_{i,T}), \tag{3.40}$$

where, for all and $i \in \{1, \dots, N\}$, the functions $\varphi_{i,t} : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^q$, for all $t \in [0, T-1]$, and $\varphi_{i,T} : \mathbb{R}^{n_i} \to \mathbb{R}^q$ represent the i-th contribution to the aggregative variable. We will use the symbol $\boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{q(T+1)}$ to denote the stack of all $\sigma_t(\cdot, \cdot)$, for all $t \in [0, T-1]$, and $\sigma_T(\cdot)$, i.e., we define

$$\boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{u}) := \begin{bmatrix} \sigma_0(x_0, u_0) \\ \vdots \\ \sigma_{T-1}(x_{T-1}, u_{T-1}) \\ \sigma_T(x_T) \end{bmatrix} = \frac{1}{N} \sum_{i=1}^N \varphi_i(\boldsymbol{x}_i, \boldsymbol{u}_i),$$

where $\varphi_i(\boldsymbol{x}_i, \boldsymbol{u}_i) := \text{col}(\varphi_{i,0}(x_{i,0}, u_{i,0}), \dots, \varphi_{i,T}(x_{i,T}))$. The agents of the network aim to cooperatively minimize the overall cost $\ell : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{q(T+1)} \to \mathbb{R}$ given by

$$\ell(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{u})) := \sum_{i=1}^N \ell_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{u})), \tag{3.41}$$

in which, for all $i \in \{1, \dots, N\}$, the local function $\ell_i : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{q(T+1)} \to \mathbb{R}$ of agent

i depends on both the local state and input and the aggregative variable $\boldsymbol{\sigma}$ coupling the states and the inputs of all the agents. Specifically, ℓ_i reads as

$$\ell_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{u})) := \sum_{t=0}^{T-1} \ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t, u_t)) + \ell_{i,T}(x_{i,T}, \sigma_T(x_T)), \tag{3.42}$$

where $\ell_{i,t} : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \times \mathbb{R}^q \to \mathbb{R}$ represents the so-called stage cost, while $\ell_{i,T} : \mathbb{R}^{n_i} \times \mathbb{R}^q \to \mathbb{R}$ is the terminal cost. Overall, our goal is to design a system trajectory $(\boldsymbol{x}, \boldsymbol{u})$ such that the global performance index (3.41) is minimized. More formally, we aim at solving the optimal control problem

$$\min_{\substack{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N \\ \boldsymbol{u}_1,\dots,\boldsymbol{u}_N}} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t, u_t)) + \ell_{i,T}(x_{i,T}, \sigma_T(x_T)) \tag{3.43a}$$

$$\text{subj.to} \quad x_{i,t+1} = f_i(x_{i,t}, u_{i,t}) \quad t = 0, \dots, T-1 \tag{3.43b}$$

$$x_{i,0} = x_{i,\text{init}} \quad i = 1, \dots, N, \tag{3.43c}$$

where $x_{i,\text{init}} \in \mathbb{R}^{n_i}$ represents the initial condition of agent i, for all $i \in \{1, \dots, N\}$.

We consider an instance of problem (3.43) in which the dynamics, the cost function and the aggregation functions satisfy the following assumptions.

Assumption 3.3 (Dynamics and cost regularity). *For all $i \in \{1, \dots, N\}$, the dynamics function $f_i(x_{i,t}, u_{i,t})$ and the cost functions $\ell_{i,t}(\cdot, \cdot, \cdot)$, $\ell_{i,T}(\cdot, \cdot)$ and $\varphi_i(\cdot, \cdot)$ are $\mathcal{C}^2$. Moreover, the cost functions $\ell(\cdot, \cdot, \boldsymbol{\sigma}(\cdot, \cdot))$ are radially unbounded.* $\triangle$

Although a solution to (3.43) could be retrieved via a centralized optimal control solver, due to the network dimension (and thus the problem one) and to robustness issues, "centralized" approaches where a single unity knows all data and controls all the agents are often undesirable or even not applicable. In the following, we develop a distributed method meant as an algorithm involving cooperation via local communication (over a network) among the N control nodes to solve (3.43), while taking advantage of the computational power available to each system.

The communication model, fundamental for the optimal control algorithm development, is formalized resorting to graph theory.

More formally, we assume the agents exchange information among each other according to a directed graph $\mathcal{G} = (\{1, \dots, N\}, \mathcal{E})$, where $\mathcal{E} \subset \{1, \dots, N\} \times \{1, \dots, N\}$ is the so-called edge set. Agent i receives information from agent j only if the edge $(j, i) \in \mathcal{E}$. The set of in-neighbors of i is $N_i := \{j \in \{1, \dots, N\} \mid (j, i) \in \mathcal{E}\}$. We associate to the graph $\mathcal{G}$ a weighted adjacency matrix $\mathcal{W} \in \mathbb{R}^{N \times N}$ whose entries match the graph, i.e., $w_{ij} > 0$ whenever $(j, i) \in \mathcal{E}$ and $w_{ij} = 0$ otherwise. Specifically, we consider graphs and adjacency matrices matching the following definitions.

Definition 3.2 (Connectivity [50]). *A directed graph $\mathcal{G}$ is said to be* strongly connected *if for every pair of nodes (i, j) there exists a path of directed edges that goes from i to j. If $\mathcal{G}$ is undirected, we say that $\mathcal{G}$ is* connected. △

Definition 3.3 (Stochastic matrices [50]). *The matrix $A \in \mathbb{R}^{N \times N}$ is said to be* row stochastic *if it holds $A\mathbf{1}_N = \mathbf{1}_N$. Analogously, A is said to be* column stochastic *if it holds $\mathbf{1}_N^\top A = \mathbf{1}_N^\top$- If A is both row and column stochastic, then it is said to be* doubly stochastic.△

In the following assumption we formalized the considered time-invariant network topology.

Assumption 3.4 (Network). *The directed graph $\mathcal{G}$ is strongly connected and the matrix $\mathcal{W}$ is doubly stochastic.* △

Inspired by the general (single-agent) case detailed in Section 3.2, we design, for each agent i, a (local) feedback policy mapping (local) state-input curves into trajectories for the i-th system. Operatively, such a policy is implemented via the nonlinear tracking system

$$u_{i,t} = \mu_{i,t} + K_{i,t}(\alpha_{i,t} - x_{i,t}), \qquad x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \tag{3.44}$$

where the matrix $K_{i,t} \in \mathbb{R}^{m_i \times n_i}$ is chosen to ensure local stability about a given state-input trajectory for the local nonlinear dynamics (3.39). The introduction of the feedback operator in (3.44) leads to the closed-loop formulation of (3.45) described by

$$\min_{\substack{\boldsymbol{x}_i,\boldsymbol{u}_i,\boldsymbol{\alpha}_i,\boldsymbol{\mu}_i \\ i=1,\ldots,N}} \sum_{i=1}^{N}\sum_{t=0}^{T-1} \ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t, u_t)) + \ell_{i,T}(x_{i,T}, \sigma_T(x_T)) \tag{3.45a}$$

$$\begin{aligned} \text{subj.to}\ & x_{i,t+1} = f_i(x_{i,t}, u_{i,t}) \quad t = 0, \ldots, T-1 && (3.45\text{b})\\ & u_{i,t} = \mu_{i,t} + K_{i,t}(\alpha_{i,t} - x_{i,t}) && (3.45\text{c})\\ & x_{i,0} = x_{i,\text{init}} \quad i = 1, \ldots, N. && (3.45\text{d}) \end{aligned}$$

Notice that the cost functions of problem (3.45) have an aggregative structure, while the constraints are local to each agent.

3.3.1 Tracking-based Distributed GoPRONTO for Aggregative Optimal Control

In this section, we detail the distributed optimization algorithm to solve (3.45) cooperatively via a protocol in which each agent employs only local data and exchanges information only with its neighbors. Let $k \in \mathbb{N}$ be the iteration index. At each iteration k, each agent i updates its local estimate $(\boldsymbol{x}_i^k, \boldsymbol{u}_i^k, \boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k)$ of the i-th component of an optimal solution to problem (3.45). Specifically, for each agent i, we first update the independent variables $(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k)$, i.e., the curve, and, then, via the feedback operator (3.44) projecting curves into trajectories, we recover the corresponding trajectory $(\boldsymbol{x}_i^k, \boldsymbol{u}_i^k)$.

The rationale behind the update mechanism of the curve $(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k)$ (cf. (3.47)) is to approximate the centralized (gradient-related) dynamics by replacing the global information (not locally available) with local proxies $s_{i,t}$ and $y_{i,t}$ for all $t \in [0,T]$. That is, for each $t \in [0,T]$, agent i updates two auxiliary variables $s_{i,t}^k \in \mathbb{R}^q$ and $y_{i,t}^k \in \mathbb{R}^q$ in order to reconstruct the aggregative variable $\sigma_t(x_t^k, u_t^k)$ and the related gradient $\sum_{i=1}^N \nabla_3 \ell_i(x_{i,t}^k, u_{i,t}^k, \sigma_t(x^k, u^k))$, respectively. Specifically, for all $i \in \{1, \dots, N\}$ and $t \in \{0, \dots, T\}$, each variable $s_{i,t}$ is tracking the quantity $\sigma_t(x_t^k, u_t^k)$ that varies along the iterations k. To this end, we update each $s_{i,t}$ by combining a consensus term $\sum_j w_{ij} s_{j,t}^k$ with the correction one $\varphi_{i,t}(x_{i,t}^{k+1}, u_{i,t}^{k+1}) - \varphi_{i,t}(x_{i,t}^k, u_{i,t}^k)$ giving rise to the so-called perturbed consensus dynamics widely adopted in the context of distributed average tracking, see, e.g., [133, 252]. The same arguments apply also for the set of trackers $y_{i,t}$. Informally, the desired asymptotic behavior of $s_{i,t}^k$ and $y_{i,t}^k$ is

$$s_{i,t}^k \xrightarrow{\text{tracks}} \sigma_t(x_t^k, u_t^k) = \frac{1}{N} \sum_{j=1}^N \varphi_{j,t}(x_{j,t}^k, u_{j,t}^k)$$

$$y_{i,t}^k \xrightarrow{\text{tracks}} \frac{1}{N} \sum_{j=1}^N \nabla_3 \ell_{j,t}(x_{j,t}^k, u_{j,t}^k, \sigma_t(x_t^k, u_t^k)),$$

for all $t \in [0, T-1]$. Similar behaviors are desired for $s_{i,T}^k$ and $y_{i,T}^k$ too.

The overall strategy is presented in Algorithm 10 from the perspective of agent i, and by adopting the shorthands

$$A_{i,t}^k := \nabla_1 f_i(x_{i,t}^k, u_{i,t}^k)^\top, \qquad B_{i,t}^k := \nabla_2 f_i(x_{i,t}^k, u_{i,t}^k)^\top$$

for all $i \in \{1, \dots, N\}$ and $t \in [0, T-1]$.

Notice that, the computational burden for each agent is proportional to the *local* state, input and aggregate variable dimensions and time length only. As for the communication network usage, steps (3.49)–(3.50) require agents to exchange T pairs of vectors in $\mathbb{R}^q$. Thereby, the data exchanged between agents consists of $2q(T+1)$ floats.

Algorithm Convergence Result

In this section, we provide the convergence result characterizing Algorithm 10.

For the sake of readability, we define the stacks $\mathbf{s}_i^k$, $\mathbf{y}_i^k$, and $G_{i,3}(\boldsymbol{x}_i^k, \boldsymbol{u}_i^k, \boldsymbol{\sigma}(\boldsymbol{x}^k, \boldsymbol{u}^k))$ as

$$\mathbf{s}_i^k := \text{col}(s_{i,0}^k, \dots, s_{i,T}^k) \in \mathbb{R}^{q(T+1)}, \qquad \mathbf{y}_i^k := \text{col}(y_{i,0}^k, \dots, y_{i,T}^k) \in \mathbb{R}^{q(T+1)}$$

$G_{i,3}(\boldsymbol{x}_i^k, \boldsymbol{u}_i^k, \mathbf{s}_i^k) :=$

Algorithm 10 Tracking-based Distributed GoPRONTO Algorithm – Agent i

```
for $k = 0, 1, 2 \dots$ do
    set $\lambda^k_{i,T} = \nabla_1 \ell_{i,T}(x^k_{i,T}, s^k_{i,T}) + \nabla \varphi^k_{i,T}(x^k_{i,T}) y^k_{i,T}$
    for $t = T-1, \dots, 0$ do
        Compute proxy variables based on $s^k_{i,t}$ and $y^k_{i,t}$
            $$a_{i,t} = \nabla_1 \ell_{i,t}(x^k_{i,t}, u^k_{i,t}, s^k_{i,t}) + \nabla_1 \varphi(x^k_{i,t}, u^k_{i,t})\, y^k_{i,t}$$
            $$b_{i,t} = \nabla_2 \ell_{i,t}(x^k_{i,t}, u^k_{i,t}, s^k_{i,t}) + \nabla_2 \varphi(x^k_{i,t}, u^k_{i,t})\, y^k_{i,t} \qquad (3.46a)$$
        Compute local proxy descent direction
            $$\lambda_{i,t} = (A^k_{i,t} - B^k_{i,t} K^k_{i,t})^\top \lambda_{i,t+1} + a_{i,t} \qquad (3.46b)$$
            $$\Delta\mu^k_{i,t} = -b_{i,t} - B^{k\top}_{i,t} \lambda_{i,t+1} \qquad (3.46c)$$
            $$\Delta\alpha^k_{i,t} = K^\top_{i,t} \Delta\mu^k_{i,t} \qquad (3.46d)$$
    Update the local solution estimates and trackers
    for $t = 0, \dots, T-1$ do
        Update curve
            $$\alpha^{k+1}_{i,t} = \alpha^k_{i,t} + \Delta\alpha^k_{i,t}$$
            $$\mu^{k+1}_{i,t} = \mu^k_{i,t} + \Delta\mu^k_{i,t} \qquad (3.47)$$
        Compute new (feasible) trajectory with $x^{k+1}_{i,0} = x_{i,\text{init}}$
            $$u^{k+1}_{i,t} = \mu^{k+1}_{i,t} + K_{i,t}(\alpha^{k+1}_{i,t} - x^{k+1}_{i,t}) \qquad (3.48a)$$
            $$x^{k+1}_{i,t+1} = f_i(x^{k+1}_{i,t}, u^{k+1}_{i,t}) \qquad (3.48b)$$
        Update trackers
            $$s^{k+1}_{i,t} = \sum_{j \in \mathcal{N}_i} w_{ij} s^k_{j,t} + \varphi_{i,t}(x^{k+1}_{i,t}, u^{k+1}_{i,t}) - \varphi_{i,t}(x^k_{i,t}, u^k_{i,t})$$
            $$y^{k+1}_{i,t} = \sum_{j \in \mathcal{N}_i} w_{ij} y^k_{j,t} + \nabla_3 \ell_{i,t}(x^{k+1}_{i,t}, u^{k+1}_{i,t}, s^{k+1}_{i,t}) - \nabla_3 \ell_{i,t}(x^k_{i,t}, u^k_{i,t}, s^k_{i,t}) \qquad (3.49)$$
        $$s^{k+1}_{i,T} = \sum_{j \in \mathcal{N}_i} w_{ij} s^k_{j,T} + \varphi_{i,T}(x^{k+1}_{i,T}) - \varphi_{i,T}(x^k_{i,T})$$
        $$y^{k+1}_{i,T} = \sum_{j \in \mathcal{N}_i} w_{ij} y^k_{j,T} + \nabla_2 \ell_{i,T}(x^{k+1}_{i,T}, s^{k+1}_{i,T}) - \nabla_2 \ell_{i,T}(x^k_{i,T}, s^k_{i,T}) \qquad (3.50)$$
```

$$\mathrm{col}\big(\nabla_3 \ell_{i,0}(x_{i,0}^k, u_{i,0}^k, s_{i,0}^k), \dots, \nabla_3 \ell_{i,T-1}(x_{i,T-1}^k, u_{i,T-1}^k, s_{i,T-1}^k), \nabla_2 \ell_{i,T}(x_{i,T}^k, s_{i,T}^k)\big) \in \mathbb{R}^{q(T+1)}.$$

The next theorem establishes the convergence guarantees of Algorithm 10.

Theorem 3.2. *Consider the distributed optimal control Algorithm 10. Let Assumptions 3.3 and 3.4 hold. For any initial condition* $(\boldsymbol{x}_i^0, \boldsymbol{u}_i^0, \mathbf{s}_i^0, \mathbf{y}_i^0) \in \mathbb{R}^{n_i T} \times \mathbb{R}^{m_i T} \times \mathbb{R}^{q(T+1)} \times \mathbb{R}^{q(T+1)}$ *such that* $(\boldsymbol{x}_i^0, \boldsymbol{u}_i^0)$ *is a trajectory and*

$$\begin{aligned} \mathbf{s}_i^0 &= \varphi_i(\boldsymbol{x}_i^0, \boldsymbol{u}_i^0), && \forall i \in \{1, \dots, N\} \\ \mathbf{y}_i^0 &= G_{i,3}(\boldsymbol{x}_i^0, \boldsymbol{u}_i^0, \mathbf{s}_i^0) && \forall i \in \{1, \dots, N\}, \end{aligned} \tag{3.51}$$

there exists $\bar{\gamma} > 0$ *such that, for any* $\gamma \in (0, \bar{\gamma})$, *it holds*

$$\lim_{k \to \infty} w\left((\boldsymbol{x}^k, \boldsymbol{u}^k), \Xi^\star\right) = 0.$$

where the symbol $\Xi^\star$ *denotes the set of trajectories* $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ *satisfying the first-order necessary conditions for optimality of the original problem* (3.43). $\triangle$

The complete proof of Theorem 3.2 is provided in Section 3.3.2. We underline that the result of Theorem 3.2 is semi-global restricted to the set defined by conditions (3.51). Notice that, as customary in nonconvex optimization (cf. Assumption 3.3), Theorem 3.2 ensures that Algorithm 10 asymptotically converges to a point satisfying the first-order necessary conditions of optimality for problem (3.43). Indeed, since the costs are nonconvex and the dynamics nonlinear, the solution of problem (3.43) is, in general, not unique. However, as detailed extensively in Remark 3.4, in a neighborhood of an (isolated) local minimum of (3.43) satisfying second-order sufficient conditions of optimality, Algorithm 10 shows a linear rate of convergence.

3.3.2 Algorithm Analysis

The analysis of the Distributed Aggregative GoPRONTO algorithm relies on the following cornerstone steps. First, in Section 3.3.2, the local feedback-based controllers (3.48a) allow us to reformulate (3.45) as a reduced (unconstrained) optimization problem. Second, in Section 3.3.2, we show that our strategy reads as an approximation of the gradient descent method (cf. Lemma 3.1) interlaced with the dynamics of the proxies for the unavailable global quantities (cf. Lemma 3.2). Third, in Section 3.3.2, we rewrite Algorithm 10 via a change of variables (cf. Lemma 3.3) that (i) highlights the approximation error due to the proxies and (ii) removes the influence of their average components on the algorithm dynamics. Finally, in Section 3.3.2, we close the proof of Theorem 3.2 via LaSalle's based arguments.

Reduced Reformulation of the Optimal Control Problem: Given an initial condition $x_{i,0} \in \mathbb{R}^{n_i}$, we define the space of trajectories of agent i as $\mathcal{T}_{i,x_{i,\text{init}}} \subseteq \mathbb{R}^{n_i T} \times \mathbb{R}^{m_i T}$ defined as for all $i \in \{1, \dots, N\}$

$$\mathcal{T}_{i,x_{i,\text{init}}} := \Big\{(\boldsymbol{x}_i, \boldsymbol{u}_i) \in \mathbb{R}^{n_i T} \times \mathbb{R}^{m_i T} \mid x_{i,0} = x_{i,\text{init}},\ x_{i,t+1} = f_i(x_{i,t}, u_{i,t}) \quad \forall t \in [0, T]\Big\}.$$

Specifically, for each agent $i \in \{1, \dots, N\}$, the local policy (3.44) implements a (local) feedback-projection operator $\mathcal{P}_i : \mathbb{R}^{n_i T} \times \mathbb{R}^{m_i T} \to \mathcal{T}_{i,x_{i,\text{init}}}$, which maps (local) state-input curves into trajectories for the i-th system. More formally, $\mathcal{P}_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)$ is designed such that for any $(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \in \mathbb{R}^{n_i T} \times \mathbb{R}^{m_i T}$ and $(\boldsymbol{x}_i, \boldsymbol{u}_i) \in \mathcal{T}_{i,x_{i,\text{init}}}$, it holds

$$\begin{bmatrix} \boldsymbol{\alpha}_i \\ \boldsymbol{\mu}_i \end{bmatrix} \longmapsto \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{u}_i \end{bmatrix} := \mathcal{P}_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) := \begin{bmatrix} \phi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \\ \psi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \end{bmatrix}, \tag{3.52}$$

where $\phi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)$ and $\psi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)$ are the state and input components of $\mathcal{P}_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)$. Notice that it holds $\mathcal{P}_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) \in \mathcal{T}_{i,x_{i,\text{init}}}$ and it satisfies the projection property $\mathcal{P}_i(\boldsymbol{x}_i, \boldsymbol{u}_i) = (\boldsymbol{x}_i, \boldsymbol{u}_i)$. For all $t \in [0, T-1]$ and $i \in \{1, \dots, N\}$, we can also define the maps

$$x_{i,t} = \phi_{i,t}(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \qquad u_{i,t} = \psi_{i,t}(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \tag{3.53}$$

where $\phi_{i,t} : \mathbb{R}^{n_i T} \times \mathbb{R}^{m_i T} \to \mathbb{R}^{n_i}$ and $\psi_{i,t} : \mathbb{R}^{n_i T} \times \mathbb{R}^{m_i T} \to \mathbb{R}^{m_i}$. We denote as $\phi_i(\cdot,\cdot)$ and $\psi_i(\cdot,\cdot)$ the stack of all $\phi_{i,t}(\cdot,\cdot)$ and $\psi_{i,t}(\cdot,\cdot)$ for all $t = 0, \dots, T-1$ and, similarly, as $\phi(\cdot,\cdot)$, $\psi(\cdot,\cdot)$ the stack of the maps $\phi_i(\cdot,\cdot)$, $\psi_i(\cdot,\cdot)$ for all $i \in \{1, \dots, N\}$. The maps (3.53) are compositions of continuous functions and thus continuous. Therefore, we can reformulate problem (3.45) as

$$\min_{\substack{\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_N \\ \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N}} \underbrace{\sum_{i=1}^{N} J_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i, \boldsymbol{\varsigma}(\boldsymbol{\alpha}, \boldsymbol{\mu}))}_{J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\varsigma}(\boldsymbol{\alpha}, \boldsymbol{\mu}))}, \tag{3.54}$$

where

$$J_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i, \boldsymbol{\varsigma}(\boldsymbol{\alpha}, \boldsymbol{\mu})) := \ell_i(\phi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \psi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \boldsymbol{\sigma}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}))),$$

in which $\boldsymbol{\varsigma}(\boldsymbol{\alpha}, \boldsymbol{\mu})$ is denoted as

$$\boldsymbol{\varsigma}(\boldsymbol{\alpha}, \boldsymbol{\mu}) := \frac{1}{N} \sum_{i=1}^{N} \bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i),$$

with $\bar{\boldsymbol{\varphi}}_i : \mathbb{R}^{n_iT} \times \mathbb{R}^{m_iT} \to \mathbb{R}^{q(T+1)}$ is defined as

$$\bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}, \boldsymbol{\mu}) := \operatorname{col}(\varphi_{i,0}(\phi_{i,t}(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \psi_{i,t}(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)), \dots, \varphi_{i,T}(\phi_{i,T}(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \psi_{i,T}(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i))).$$

Problem (3.54) is an unconstrained optimization problem in $(\boldsymbol{\alpha}, \boldsymbol{\mu})$. Thus, it can be addressed by a distributed implementation of the gradient descent method. In this context, each agent i iteratively updates its local estimate $(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k) \in \mathbb{R}^{n_iT} \times \mathbb{R}^{m_iT}$ about the i-th component of a solution of problem (3.54), with $k \in \mathbb{N}$ denoting the iteration index. The gradient method update applied to problem (3.54) reads as

$$\boldsymbol{\alpha}_i^{k+1} = \boldsymbol{\alpha}_i^k - \gamma \Gamma_{\boldsymbol{\alpha}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \varsigma(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)) \tag{3.55a}$$

$$\boldsymbol{\mu}_i^{k+1} = \boldsymbol{\mu}_i^k - \gamma \Gamma_{\boldsymbol{\mu}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \varsigma(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)), \tag{3.55b}$$

for all $i \in \{1, \dots, N\}$, $\gamma > 0$ is the so-called step-size, and we introduced the operators $\Gamma_{\boldsymbol{\alpha}_i} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{q(T+1)} \to \mathbb{R}^{n_iT}$ and $\Gamma_{\boldsymbol{\mu}_i} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{q(T+1)} \to \mathbb{R}^{m_iT}$ defined as

$$\Gamma_{\boldsymbol{\alpha}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \varsigma(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)) := \left.\frac{\partial J(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i^k, \varsigma(\boldsymbol{\alpha}, \boldsymbol{\mu}^k))}{\partial \boldsymbol{\alpha}_i}\right|_{\boldsymbol{\alpha}=\boldsymbol{\alpha}_i^k}$$

$$\Gamma_{\boldsymbol{\mu}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \varsigma(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)) := \left.\frac{\partial J(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i, \varsigma(\boldsymbol{\alpha}^k, \boldsymbol{\mu}))}{\partial \boldsymbol{\mu}_i}\right|_{\boldsymbol{\mu}_i=\boldsymbol{\mu}_i^k}.$$

In order to implement (3.55), we need to overcome two main challenges. Firstly, being the cost function in (3.54) a composition of ℓ_i with maps (3.53), we need to calculate $\nabla\phi_i$ and $\nabla\psi_i$, which, in general is non trivial. Secondly, the cost function (3.54) also exhibits a composition with the aggregative variable ς that couples all the variables $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ of the network. Consequently, the update (3.55) requires, for each agent in the network, the knowledge of unavailable global information. This issue is tackled by properly exploiting a consensus-based strategy. The next section is devoted to providing insights into these two aspects.

Descent Direction and Local Proxies: In this section, we formally define a useful numerical routine for the computation of the gradient of the reduced cost function $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \varsigma(\boldsymbol{\alpha}, \boldsymbol{\mu}))$. Then, we show how to compensate for the unavailable global information required by this routine.

Lemma 3.1. *For all $i = 1, \dots, N$, consider the integration backward in time from $t = T-1, \dots, 0$ of the closed-loop costate equation*

$$\lambda_{i,t} = (A_{i,t} - K_{i,t}B_{i,t})^\top \lambda_{i,t+1} + a_{i,t},$$

where $\lambda_{i,t} \in \mathbb{R}^{n_i}$*,*

$$a_{i,t} = \nabla_2 \ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t, u_t)) + \nabla_2 \varphi_{i,t}(x_{i,t}, u_{i,t}) \sum_{j=1}^{N} \nabla_3 \ell_{j,t}(x_{j,t}, u_{j,t}, \sigma_t(x_t, u_t)), \tag{3.56a}$$

and the initial condition $\lambda_{i,T} \in \mathbb{R}^{n_i}$*, is set as*

$$\lambda_{i,T} = \nabla \ell_{i,T}(x_{i,T}, \sigma_T(x_T)) + \nabla \varphi_{i,T}(x_{i,T}) \sum_{j=1}^{N} \nabla_2 \ell_{j,T}(x_{j,T}, \sigma_t(x_T)). \tag{3.56b}$$

Then, for all $i = 1, \dots, N$*, the gradient of the reduced cost function* (3.54) *with respect to* $\mu_{i,t}$ *and* $\alpha_{i,t}$ *can be calculated respectively as*

$$\Gamma_{\boldsymbol{\alpha}_i}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{\varsigma}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)) = -b_{i,t} - B_{i,t}\lambda_{i,t+1} \tag{3.57a}$$

$$\Gamma_{\boldsymbol{\mu}_i}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{\varsigma}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)) = K_{i,t}^\top \Delta\mu_{i,t}, \tag{3.57b}$$

where

$$b_{i,t} = \nabla_2 \ell_{i,t}(x_{i,t}, u_{i,t}) + \nabla_2 \varphi_{i,t}(x_{i,t}, u_{i,t}) \sum_{j=1}^{N} \nabla_3 \ell_{j,t}(x_{j,t}, u_{j,t}, \sigma_t(x_t, u_t)). \tag{3.58}$$

for all $t = 0, \dots, T$ $\triangle$

Proof. The proof extends the arguments we leverage in the proof of Theorem 3.1 to the multi-agent case. First of all, we rewrite the equality constraints represented by the nonlinear dynamics in (3.45b) and the closed-loop structure of the control input (3.45c) in its implicit form $h : \mathbb{R}^{n_iT} \times \mathbb{R}^{m_iT} \times \mathbb{R}^{n_iT} \times \mathbb{R}^{m_iT} \to \mathbb{R}^{n_iT}$ defined as

$$h_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) := \begin{bmatrix} f_i(x_{i,0}, u_{i,0}) - x_{i,1} \\ \vdots \\ f_i(x_{i,T-1}, u_{i,T-1}) - x_{i,T} \\ \mu_{i,0} + K_t(\alpha_{i,0} - x_{i,0}) - u_{i,0} \\ \vdots \\ \mu_{i,T-1} + K_{i,t}(\alpha_{i,T-1} - x_{i,T-1}) - u_{i,T-1} \end{bmatrix}. \tag{3.59}$$

We can now introduce an auxiliary function $\mathcal{L} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{nT+m} \to \mathbb{R}$, defined as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda}) := \sum_{i=1}^{N} \Big(\ell_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{u})) + h_i(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\alpha}_i, \boldsymbol{\mu}_i)^\top \boldsymbol{\lambda}_i \Big), \tag{3.60}$$

where, for all $i \in \{1, \dots, N\}$, the (multiplier) vector $\boldsymbol{\lambda}_i \in \mathbb{R}^{n_i T + m_i T}$ is arranged as

$$\boldsymbol{\lambda}_i := \operatorname{col}(\lambda_{i,1}, \dots, \lambda_{i,T}, \tilde{\lambda}_{i,1}, \dots, \tilde{\lambda}_{i,T}),$$

with $\lambda_{i,t} \in \mathbb{R}^n$ and $\tilde{\lambda}_{i,t} \in \mathbb{R}^m$, for all $t \in [0, T]$. By construction, for any $(\boldsymbol{\alpha}, \boldsymbol{\mu}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$, it holds

$$h_i(\phi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \psi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \boldsymbol{\alpha}_i, \boldsymbol{\mu}_i) = 0. \tag{3.61}$$

Hence, the auxiliary function (3.60) enjoys the property

$$\begin{aligned}\mathcal{L}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\lambda}) &= \sum_{i=1}^{N} \ell_i(\phi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i), \psi_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i, \boldsymbol{\sigma}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}))) \\ &= \sum_{i=1}^{N} J_i(\boldsymbol{\alpha}_i, \boldsymbol{\mu}_i, \varsigma(\boldsymbol{\alpha}, \boldsymbol{\mu})), \qquad (3.62)\end{aligned}$$

for *any* $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ and $\boldsymbol{\lambda} \in \mathbb{R}^{nT+mT}$. Therefore, in this formulation, we can interpret $\boldsymbol{\lambda}$ as a parameter or a degree of freedom. Indeed, building on (3.62), one can simplify the computation of the gradient of J with a proper choice of $\boldsymbol{\lambda}$ (see [212] for further details). Such a choice gives rise to the decoupled computation

$$\lambda_{i,t} = (A_{i,t} - K_{i,t} B_{i,t})^\top \lambda_{i,t+1} + a_{i,t},$$

which, in turn, leads to

$$\Gamma_{\boldsymbol{\alpha}_i}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \varsigma(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)) = -b_{i,t} - B_{i,t} \lambda_{i,t+1} \tag{3.63a}$$

$$\Gamma_{\boldsymbol{\mu}_i} J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \varsigma(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)) = K_{i,t}^\top \Delta \mu_{i,t}, \tag{3.63b}$$

where

$$a_{i,t} = \nabla_1 \ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t, u_t)) + \nabla_1 \varphi_{i,t}(x_{i,t}, u_{i,t}) \sum_{j=1}^{N} \nabla_3 \ell_{j,t}(x_{j,t}, u_{j,t}, \sigma_t(x_t, u_t)) \tag{3.64a}$$

$$b_{i,t} = \nabla_2 \ell_{i,t}(x_{i,t}, u_{i,t}) + \nabla_2 \varphi_{i,t}(x_{i,t}, u_{i,t}) \sum_{j=1}^{N} \nabla_3 \ell_{j,t}(x_{j,t}, u_{j,t}, \sigma_t(x_t, u_t)). \tag{3.64b}$$

Equations (3.63) should be calculated backward in time from $t = T-1, \dots, 0$, with initial condition $\lambda_{i,T} = a_T$, where

$$a_{i,T} = \nabla_1 \ell_{i,T}(x_{i,T}, \sigma_T(x_T)) + \nabla \varphi_{i,T}(x_{i,T}) \sum_{j=1}^{N} \nabla_2 \ell_{j,T}(x_{j,T}, \sigma_t(x_T)).$$

The proof follows. ■

Notice that, the terms in (3.56) and (3.58) require the local knowledge of the global quantities $\sigma_t(\cdot,\cdot)$, $\sum_{j=1}^{N} \nabla_3 \ell_{j,t}(x_{j,t}, u_{j,t}, \sigma_t(\cdot,\cdot))$, $\sigma_T(\cdot)$, and $\sum_{j=1}^{N} \nabla_2 \ell_{j,T}(x_{j,T}, \sigma_T(\cdot))$. For this reason, in the computation of each $a_{i,t}$, $b_{i,t}$, and $a_{i,T}$, these global quantities are replaced by the trackers $s_{i,t}$ and $y_{i,t}$. Concurrently, these trackers are updated according to perturbed consensus dynamics exploiting inter-agent communication (cf. (3.49) and (3.50)). Hence, we formally show the relationship between the directions $\boldsymbol{\Delta\alpha}_i^k$ and $\boldsymbol{\Delta\mu}_i^k$ computed via Algorithm 10 and the derivative of the reduced costs J_i with respect to $\boldsymbol{\alpha}_i$ and $\boldsymbol{\mu}_i$.

Lemma 3.2 (Local proxy of the descent direction)**.** *Consider Algorithm 10. Then, it holds*

$$\begin{aligned}
\boldsymbol{\Delta\alpha}_i^k &= -\Gamma_{\boldsymbol{\alpha}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \mathbf{s}_i^k) - \nabla_1 \bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k)\, \mathbf{y}_i^k \\
\boldsymbol{\Delta\mu}_i^k &= -\Gamma_{\boldsymbol{\mu}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \mathbf{s}_i^k) - \nabla_2 \bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k)\, \mathbf{y}_i^k,
\end{aligned}$$

for all $i \in \{1, \ldots, N\}$. △

Proof. At each iteration k, consider the quantities $\mathbf{s}_i^k$ and $\mathbf{y}_i^k$ as the numerical proxies of the centralized quantities $\sigma_t(x_t, u_t)$ and $\sum_{i=1}^{N} \nabla_3 \ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t, u_t))$, respectively. Considering equations (3.56)-(3.57) of Lemma 3.1, one can see that the backward sweep (3.46) computes the approximated gradient of the reduced cost J at the current iteration. ■

Algorithm reformulation: average and perpendicular dynamics of the trackers in error coordinates: Next, we provide a reformulation of Algorithm 10 used to prove Theorem 3.2. The initial condition $(\boldsymbol{x}_i^0, \boldsymbol{u}_i^0)$ is a trajectory, $(\mathbf{s}_i^0, \mathbf{y}_i^0)$ is such that (3.51) holds, and Assumptions 3.3 and 3.4 hold. By exploiting Lemma 3.2, the local update describing Algorithm 10 can be summarized as

$$\boldsymbol{\alpha}_i^{k+1} = \boldsymbol{\alpha}_i^k - \gamma\big(\Gamma_{\boldsymbol{\alpha}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \mathbf{s}_i^k) + \nabla_1 \bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \mathbf{s}_i^k)\mathbf{y}_i^k\big) \tag{3.65a}$$

$$\boldsymbol{\mu}_i^{k+1} = \boldsymbol{\mu}_i^k - \gamma\big(\Gamma_{\boldsymbol{\mu}_i}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \mathbf{s}_i^k) + \nabla_2 \bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \mathbf{s}_i^k)\mathbf{y}_i^k\big) \tag{3.65b}$$

$$\mathbf{s}_i^{k+1} = \sum_{j \in N_i} w_{ij} \mathbf{s}_j^k + \bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}_i^{k+1}, \boldsymbol{\mu}_i^{k+1}) - \bar{\boldsymbol{\varphi}}_i(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k) \tag{3.65c}$$

$$\mathbf{y}_i^{k+1} = \sum_{j \in N_i} w_{ij} \mathbf{y}_j^k + \nabla_3 J_i(\boldsymbol{\alpha}_i^{k+1}, \boldsymbol{\mu}_i^{k+1}, \mathbf{s}_i^{k+1}) - \nabla_3 J_i(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k, \mathbf{s}_i^k). \tag{3.65d}$$

Now, we set $W := \mathcal{W} \otimes I_{q(T+1)}$ and use (3.65) to compactly describe the update of all the variables of the network as

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k - \gamma(\Gamma_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \mathbf{s}^k) + \nabla_1 \bar{\boldsymbol{\varphi}}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\mathbf{y}^k) \tag{3.66a}$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k - \gamma(\Gamma_{\boldsymbol{\mu}}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \mathbf{s}^k) + \nabla_2\bar{\varphi}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\mathbf{y}^k) \tag{3.66b}$$

$$\mathbf{s}^{k+1} = W\mathbf{s}^k + \bar{\varphi}(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1}) - \bar{\varphi}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k) \tag{3.66c}$$

$$\mathbf{y}^{k+1} = W\mathbf{y}^k + \Gamma_{\varsigma}(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1}, \mathbf{s}^{k+1}) - \Gamma_{\varsigma}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \mathbf{s}^k), \tag{3.66d}$$

where we introduced the operators $\Gamma_{\boldsymbol{\alpha}} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{q(T+1)N} \to \mathbb{R}^{nT}$, $\Gamma_{\boldsymbol{\mu}} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{q(T+1)N} \to \mathbb{R}^{mT}$, and $\Gamma_{\varsigma} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{q(T+1)N} \to \mathbb{R}^{q(T+1)N}$ defined as

$$\Gamma_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{s}) := \operatorname{col}(\Gamma_{\boldsymbol{\alpha}_1}(\boldsymbol{\alpha}_1, \boldsymbol{\mu}_1, \mathbf{s}_1), \dots, \Gamma_{\boldsymbol{\alpha}_N}(\boldsymbol{\alpha}_N, \boldsymbol{\mu}_N, \mathbf{s}_N)) \tag{3.67a}$$

$$\Gamma_{\boldsymbol{\mu}}(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{s}) := \operatorname{col}(\Gamma_{\boldsymbol{\mu}_1}(\boldsymbol{\alpha}_1, \boldsymbol{\mu}_1, \mathbf{s}_1), \dots, \Gamma_{\boldsymbol{\mu}_N}(\boldsymbol{\mu}_N, \boldsymbol{\mu}_N, \mathbf{s}_N)) \tag{3.67b}$$

$$\Gamma_{\varsigma}(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{s}) := \operatorname{col}(\nabla_3 J_1(\boldsymbol{\alpha}_1, \boldsymbol{\mu}_1, \mathbf{s}_1), \dots, \nabla_3 J_N(\boldsymbol{\alpha}_N, \boldsymbol{\mu}_N, \mathbf{s}_N)), \tag{3.67c}$$

with $\boldsymbol{\alpha} := \operatorname{col}(\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_N)$, $\boldsymbol{\mu} := \operatorname{col}(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N)$, and $\mathbf{s} := \operatorname{col}(\mathbf{s}_1, \dots, \mathbf{s}_N)$, with $\boldsymbol{\alpha}_i \in \mathbb{R}^{n_iT}$, $\boldsymbol{\mu}_i \in \mathbb{R}^{m_iT}$, and $\mathbf{s}_i \in \mathbb{R}^{q(T+1)}$ for all $i \in \{1, \dots, N\}$. The next lemma introduces a useful change of coordinates which simplifies the analysis of the algorithm. Before providing its formal statement, let us introduce the compact notation $\mathbf{1} := 1_N \otimes I_{q(T+1)}$ and the operators $\Gamma_{\boldsymbol{\xi}} : \mathbb{R}^{(n+m)T} \times \mathbb{R}^{2q(T+1)N} \to \mathbb{R}^{(n+m)T}$, $u_1 : \mathbb{R}^{(n+m)T} \times \mathbb{R}^{2q(T+1)N} \to q(T+1)N$, and $u_2 : \mathbb{R}^{(n+m)T} \to 2q(T+1)N$ defined as

$$\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}, \mathbf{s}) := \begin{bmatrix} \Gamma_{\boldsymbol{\alpha}}(\boldsymbol{\xi}, \mathbf{s}) \\ \Gamma_{\boldsymbol{\mu}}(\boldsymbol{\xi}, \mathbf{s}) \end{bmatrix} + \nabla\bar{\varphi}(\boldsymbol{\xi})\Gamma_{\varsigma}(\boldsymbol{\xi}, \mathbf{s}) \tag{3.68a}$$

$$u_1(\boldsymbol{\xi}, \boldsymbol{\vartheta}) := \Gamma_{\varsigma}(\boldsymbol{\xi}, \mathbf{1}\varsigma(\boldsymbol{\xi}) + \begin{bmatrix} R & 0 \end{bmatrix} \boldsymbol{\vartheta}) - \Gamma_{\varsigma}(\boldsymbol{\xi}, \mathbf{1}\varsigma(\boldsymbol{\xi})) \tag{3.68b}$$

$$u_2(\boldsymbol{\xi}, \boldsymbol{\xi}') := \begin{bmatrix} \bar{\varphi}(\boldsymbol{\xi}) - \bar{\varphi}(\boldsymbol{\xi}') \\ \Gamma_{\varsigma}(\boldsymbol{\xi}, \mathbf{1}\varsigma(\boldsymbol{\xi})) - \Gamma_{\varsigma}(\boldsymbol{\xi}', \mathbf{1}\varsigma(\boldsymbol{\xi}')) \end{bmatrix}, \tag{3.68c}$$

where, with a slight abuse of notation, we joined the first two arguments of the operators ς, $\bar{\varphi}$, $\Gamma_{\boldsymbol{\alpha}}$, $\Gamma_{\boldsymbol{\mu}}$, and Γ_{ς} to adapt them for this notation using $\boldsymbol{\xi}$ instead of $(\boldsymbol{\alpha}, \boldsymbol{\mu})$.

Lemma 3.3 (Dynamics reformulation). *Consider* (3.66). *Then, there exists a set of coordinates* $(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \in \mathbb{R}^{(n+m)T} \times \mathbb{R}^{2(N-1)q(T+1)}$ *such that* (3.66) *is equivalent to*

$$\boldsymbol{\xi}^{k+1} = \boldsymbol{\xi}^k - \gamma\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k) + \begin{bmatrix} R & 0 \end{bmatrix} \boldsymbol{\vartheta}^k) - \gamma \begin{bmatrix} 0 & \nabla\bar{\varphi}(\boldsymbol{\xi}^k)R \end{bmatrix} \boldsymbol{\vartheta}^k \tag{3.69a}$$

$$\boldsymbol{\vartheta}^{k+1} = \mathcal{A}\boldsymbol{\vartheta}^k + \mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) + \mathcal{B}_2 u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1}), \tag{3.69b}$$

where $R \in \mathbb{R}^{N(q(T+1)) \times (N-1)(q(T+1))}$ *is such that* $R^\top R = I$, $R^\top \mathbf{1} = 0$ *and* $\mathcal{A}$, $\mathcal{B}_1$, $\mathcal{B}_2$ *are given matrices defined as*

$$\mathcal{A} := \begin{bmatrix} R^\top W R & 0 \\ 0 & R^\top W R \end{bmatrix}, \quad \mathcal{B}_1 := \begin{bmatrix} 0 \\ R^\top (W - I) \end{bmatrix}, \quad \mathcal{B}_2 := \begin{bmatrix} R^\top & 0 \\ 0 & R^\top \end{bmatrix}. \tag{3.70}$$

Proof. Let $\boldsymbol{\xi}_i^k := \operatorname{col}(\boldsymbol{\alpha}_i^k, \boldsymbol{\mu}_i^k)$ for all $i \in \{1, \dots, N\}$, and $\boldsymbol{\xi}^k := \operatorname{col}(\boldsymbol{\xi}_1^k, \dots, \boldsymbol{\xi}_N^k)$. Then, by using the notation adopted in (3.67), we rewrite (3.66) as

$$\boldsymbol{\xi}^{k+1} = \boldsymbol{\xi}^k - \gamma \begin{bmatrix} \Gamma_{\boldsymbol{\alpha}}(\boldsymbol{\xi}^k, \mathbf{s}^k) \\ \Gamma_{\boldsymbol{\mu}}(\boldsymbol{\xi}^k, \mathbf{s}^k) \end{bmatrix} + \nabla \bar{\varphi}(\boldsymbol{\xi}^k) \begin{bmatrix} \mathbf{s}^k \\ \mathbf{y}^k \end{bmatrix} \tag{3.71a}$$

$$\mathbf{s}^{k+1} = W\mathbf{s}^k + \bar{\varphi}(\boldsymbol{\xi}^{k+1}) - \bar{\varphi}(\boldsymbol{\xi}^k) \tag{3.71b}$$

$$\mathbf{y}^{k+1} = W\mathbf{y}^k + \Gamma_{\varsigma}(\boldsymbol{\xi}^{k+1}, \mathbf{s}^{k+1}) - \Gamma_{\varsigma}(\boldsymbol{\xi}^k, \mathbf{s}^k), \tag{3.71c}$$

where, with a slight abuse of notation, we joined the first two arguments of the operators $\Gamma_{\boldsymbol{\alpha}}$ and $\Gamma_{\boldsymbol{\mu}}$. Let us introduce the new coordinates $\boldsymbol{w}^k, \mathbf{z}^k \in \mathbb{R}^{Nq(T+1)}$ defined as

$$\boldsymbol{w}^k := \mathbf{s}^k - \bar{\varphi}(\mathrm{s}^k), \quad \mathbf{z}^k := \mathbf{y}^k - \Gamma_{\varsigma}(\boldsymbol{\xi}^k, \mathbf{s}^k),$$

which allow us to rewrite (3.71) as

$$\boldsymbol{\xi}^{k+1} = \boldsymbol{\xi}^k - \gamma\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}^k, \bar{\varphi}(\boldsymbol{\xi}^k) + \boldsymbol{w}^k) - \gamma\nabla\bar{\varphi}(\boldsymbol{\xi}^k)\mathbf{z}^k \tag{3.72a}$$

$$\boldsymbol{w}^{k+1} = W\boldsymbol{w}^k + (W - I)\bar{\varphi}(\boldsymbol{\xi}^k) \tag{3.72b}$$

$$\mathbf{z}^{k+1} = W\mathbf{z}^k + (W - I)\Gamma_{\varsigma}(\boldsymbol{\xi}^k, \bar{\varphi}(\boldsymbol{\xi}^k) + \boldsymbol{w}^k), \tag{3.72c}$$

where the operator $\Gamma_{\boldsymbol{\xi}}$ is given in (3.67). Note that the initialization asked in Theorem 3.2 leads to

$$\boldsymbol{w}^0 = 0, \quad \mathbf{z}^0 = 0. \tag{3.73}$$

Let $\mathcal{S} \subset \mathbb{R}^{(n+m)T} \times \mathbb{R}^{q(T+1)} \times \mathbb{R}^{q(T+1)}$ be defined as

$$\mathcal{S} := \{(\boldsymbol{\xi}, \boldsymbol{w}, \mathbf{z}) \mid \mathbf{1}^\top \boldsymbol{w}^0 = 0, \mathbf{1}^\top \mathbf{z}^0 = 0\}. \tag{3.74}$$

Since $\mathcal{W}$ is doubly stochastic (cf. Assumption 3.4), it holds $\mathbf{1}^\top W = \mathbf{1}^\top$ and, thus, $\mathbf{1}^\top(W - I) = 0$. This property allows for verifying that $\mathcal{S}$ is invariant for (3.72). Moreover, W has all the eigenvalues strictly inside the unit circle except for the $q(T+1)$ equal to 1 with eigenvectors equal to columns of $\mathbf{1}$. We leverage these properties to isolate the invariant portion of (3.72). A suitable decomposition of W is encoded into the transformation matrix T defined as $\mathrm{T} := \begin{bmatrix} \frac{\mathbf{1}}{N} & R \end{bmatrix}^\top$, where $R \in \mathbb{R}^{N(q(T+1))\times(N-1)(q(T+1))}$ such that $R^\top R = I$, $R^\top \mathbf{1} = 0$. Moreover, the following useful relation holds true

$$RR^\top = I - \tfrac{1}{N}\mathbf{1}\mathbf{1}^\top. \tag{3.75}$$

Now, with T at hand, we perform the change coordinates

$$\begin{bmatrix}\boldsymbol{\xi}^k\\ \boldsymbol{w}^k\\ \mathbf{z}^k\end{bmatrix} \longmapsto \begin{bmatrix}\boldsymbol{\xi}^k\\ \boldsymbol{w}_{\text{avg}}^k\\ \boldsymbol{\eta}^k\\ \mathbf{z}_{\text{avg}}^k\\ \boldsymbol{\zeta}^k\end{bmatrix} := \begin{bmatrix}I & 0 & 0\\ 0 & \mathrm{T} & 0\\ 0 & 0 & \mathrm{T}\end{bmatrix}\begin{bmatrix}\boldsymbol{\xi}^k\\ \boldsymbol{w}^k\\ \mathbf{z}^k\end{bmatrix}, \tag{3.76}$$

with $\eta^k, \zeta^k \in \mathbb{R}^{(N-1)q(T+1)}$ and $\boldsymbol{w}_{\text{avg}}^k, \mathbf{z}_{\text{avg}}^k \in \mathbb{R}^{q(T+1)}$. System (3.72) in the new coordinates (3.76) reads as

$$\begin{aligned}
\boldsymbol{\xi}^{k+1} &= \boldsymbol{\xi}^k - \gamma\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}^k, \bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k) + R\boldsymbol{\eta}^k + \bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k)\mathbf{1}\boldsymbol{w}_{\text{avg}}^k) - \gamma\nabla\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k)R\boldsymbol{\zeta}^k - \gamma\nabla\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k)\mathbf{1}\mathbf{z}_{\text{avg}}^k && \text{(3.77a)}\\
\boldsymbol{w}_{\text{avg}}^{k+1} &= \boldsymbol{w}_{\text{avg}}^k && \text{(3.77b)}\\
\boldsymbol{\eta}^{k+1} &= R^\top WR\boldsymbol{\eta}^k + R^\top(W - I)\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k) && \text{(3.77c)}\\
\mathbf{z}_{\text{avg}}^{k+1} &= \mathbf{z}_{\text{avg}}^k && \text{(3.77d)}\\
\boldsymbol{\zeta}^{k+1} &= R^\top WR\boldsymbol{\zeta}^k + R^\top(W - I)\Gamma_{\varsigma}(\boldsymbol{\xi}^k, \bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k) + R\boldsymbol{\eta}^k + \mathbf{1}\boldsymbol{w}_{\text{avg}}^k). && \text{(3.77e)}
\end{aligned}$$

We observe that $R^\top WR$ is Schur and that $\boldsymbol{w}_{\text{avg}}^k = \boldsymbol{w}_{\text{avg}}^0$ and $\mathbf{z}_{\text{avg}}^k = \mathbf{z}_{\text{avg}}^0$ for any $k \geq 0$. Further, the initialization (3.73) implies $\boldsymbol{w}^0 = \mathbf{z}^0 = 0$ and, thus, $\boldsymbol{w}_{\text{avg}}^k = 0$ and $\mathbf{z}_{\text{avg}}^k = 0$ for all $k \geq 0$. Hence, with this initialization, we ignore the variables $\boldsymbol{w}_{\text{avg}}^k$ and $\mathbf{z}_{\text{avg}}^k$ rewriting (3.77) as the reduced, equivalent system

$$\begin{aligned}
\boldsymbol{\xi}^{k+1} &= \boldsymbol{\xi}^k - \gamma\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}^k, \bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k) + R\boldsymbol{\eta}^k) - \gamma\nabla\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k)R\boldsymbol{\zeta}^k && \text{(3.78a)}\\
\boldsymbol{\eta}^{k+1} &= R^\top WR\boldsymbol{\eta}^k + R^\top(W - I)\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k) && \text{(3.78b)}\\
\boldsymbol{\zeta}^{k+1} &= R^\top WR\boldsymbol{\zeta}^k + R^\top(W - I)\Gamma_{\varsigma}(\boldsymbol{\xi}^k, \bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k) + R\boldsymbol{\eta}^k). && \text{(3.78c)}
\end{aligned}$$

By using the fact that $\mathcal{W}$ is doubly stochastic (cf. Assumption 3.4) and the relation (3.75), it is possible to show that, for any $\boldsymbol{\xi}^k \in \mathbb{R}^{(n+m)T}$, the point

$$h(\boldsymbol{\xi}^k) := \begin{bmatrix} -R^\top\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k)\\ -R^\top\Gamma_{\varsigma}(\boldsymbol{\xi}^k, \mathbf{1}\sigma(\boldsymbol{\xi}^k))\end{bmatrix} \in \mathbb{R}^{2(N-1)q(T+1)},$$

represents an equilibrium of (3.78b) and (3.78c) parametrized in $\boldsymbol{\xi}^k$. Hence, we define the error coordinates $\boldsymbol{\vartheta}^k \in \mathbb{R}^{2(N-1)(q(T+1))}$ with respect to $h(\mathbf{s}^k)$ as

$$\begin{bmatrix}\boldsymbol{\eta}^k\\ \boldsymbol{\zeta}^k\end{bmatrix} \longmapsto \boldsymbol{\vartheta}^k := \begin{bmatrix}\boldsymbol{\vartheta}_1^k\\ \boldsymbol{\vartheta}_2^k\end{bmatrix} := \begin{bmatrix}\boldsymbol{\eta}^k\\ \boldsymbol{\zeta}^k\end{bmatrix} - h(\boldsymbol{\xi}^k), \tag{3.79}$$

which, exploiting the fact that $\mathcal{W}$ is doubly stochastic (cf. 3.4) and the relation (3.75), allow us to rewrite system (3.78) as

$$\begin{aligned}
\boldsymbol{\xi}^{k+1} &= \boldsymbol{\xi}^k - \gamma\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k) + \begin{bmatrix} R & 0 \end{bmatrix}\boldsymbol{\vartheta}^k) - \gamma\begin{bmatrix} 0 & \nabla\bar{\varphi}(\boldsymbol{\xi}^k)R \end{bmatrix}\boldsymbol{\vartheta}^k \\
\boldsymbol{\vartheta}^{k+1} &= \begin{bmatrix} R^\top W R & 0 \\ 0 & R^\top W R \end{bmatrix}\boldsymbol{\vartheta}^k + \begin{bmatrix} R^\top & 0 \\ 0 & R^\top \end{bmatrix}\begin{bmatrix} \bar{\varphi}(\boldsymbol{\xi}^{k+1}) - \bar{\varphi}(\boldsymbol{\xi}^k) \\ (\Gamma_\varsigma(\boldsymbol{\xi}^{k+1}, \mathbf{1}\varsigma(\boldsymbol{\xi}^{k+1})) - \Gamma_\varsigma(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k))) \end{bmatrix} \\
&\quad + \begin{bmatrix} 0 \\ R^\top(W - I) \end{bmatrix}\Big(\Gamma_\varsigma(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k) + \begin{bmatrix} R & 0 \end{bmatrix}\boldsymbol{\vartheta}^k) - \begin{bmatrix} 0 \\ R^\top(W - I) \end{bmatrix}\Gamma_\varsigma(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k))\Big).
\end{aligned}$$

The proof follows by using the operators in (3.68) and setting using the definitions of $\mathcal{A}$, $\mathcal{B}_1$ and $\mathcal{B}_2$ in Lemma 3.3. ■

As it will be clearer from the proof of Lemma 3.3, the average value across the network of $\mathbf{s}^k$ and $\mathbf{y}^k$ is invariant along the algorithm evolution and it is set to zero via the initialization (3.51). The new variable $\boldsymbol{\vartheta}^k$ describes the evolution of the error of both s^k and $\mathbf{y}^k$ with respect to the global information $\varsigma(\boldsymbol{\mu}^k, \boldsymbol{\alpha}^k)$ and $\sum_{i=1}^N \nabla_3 J_i(\boldsymbol{\mu}_i^k, \boldsymbol{\alpha}_i^k, \varsigma(\boldsymbol{\mu}^k, \boldsymbol{\alpha}^k))$ while neglecting the aforementioned invariant components.

We are now ready to give the proof of the main theorem.

Proof of Theorem 3.2

Leveraging on Lemma 3.3, we provide the proof of Theorem 3.2 by going through three main steps. In step (i), we show that ∇J, Γ_ς, and $\bar{\varphi}$ are locally Lipschitz continuous and compute the related constants. In step (ii), we employ these results to perform a LaSalle-based analysis demonstrating that the sequence $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}_{k\geq 0}$ generated by Algorithm 10 converges to the set of stationary points of the reduced cost function $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \varsigma(\boldsymbol{\alpha}, \boldsymbol{\mu}))$. In step (iii), we show that the state-input trajectory $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ obtained by projecting $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ satisfies the first-order necessary conditions for optimality in correspondence of the costate sequence $\boldsymbol{\lambda}^\star$.

Step (i): Lipschitz continuity of ∇J, Γ_ς, and $\bar{\varphi}$ Let us define the function $V : \mathbb{R}^{(n+m)T} \times \mathbb{R}^{2(N-1)q(T+1)} \to \mathbb{R}$ as

$$V(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\vartheta}) := J(\boldsymbol{\xi}, \varsigma(\boldsymbol{\xi})) + \boldsymbol{\vartheta}^\top P\boldsymbol{\vartheta}, \tag{3.80}$$

where $P \in \mathbb{R}^{2(N-1)q\times 2(N-1)q(T+1)}$ has the diagonal structure

$$P := \begin{bmatrix} P_1 & 0 \\ 0 & pP_2 \end{bmatrix},$$

with $P_1, P_2 \in \mathbb{R}^{(N-1)q(T+1)}$ with $P_1 = P_1^\top > 0$ and $P_2 = P_2^\top > 0$, while $p \in (0, 1]$ is a parameter that we will fix later. In particular, we arbitrarily choose $Q_1, Q_2 \in$

$\mathbb{R}^{(N-1)q(T+1)}$ such that $Q_1 = Q_1^\top > 0$ and $Q_2 = Q_2^\top > 0$ and pick P_1 and P_2 as the solutions of the Lyapunov equations

$$P_1 - (R^\top W R)^\top P_1 (R^\top W R) = Q_1 \tag{3.81a}$$

$$P_2 - (R^\top W R)^\top P_2 (R^\top W R) = Q_2. \tag{3.81b}$$

Indeed, in light of Assumption 3.4, the matrix $R^\top W R$ is Schur and, thus, given $Q_1 = Q_1^\top > 0$ and $Q_2 = Q_2^\top > 0$, there always exist P_1 and P_2 solving (3.81). Now, let $(\boldsymbol{\xi}^0, \boldsymbol{\vartheta}^0) \in \mathbb{R}^{(n+m)T} \times \mathbb{R}^{2(N-1)q(T+1)}$ be the vectors corresponding to the initial conditions $\boldsymbol{x}_i^0$, $\boldsymbol{u}_i^0$, $\mathbf{s}_i^0$, and $\mathbf{y}_i^0$ assumed in Theorem 3.2 and, hence, satisfying (3.51). Let $\boldsymbol{\vartheta} := \operatorname{col}(\boldsymbol{\vartheta}_1, \boldsymbol{\vartheta}_2)$ with $\boldsymbol{\vartheta}_1, \boldsymbol{\vartheta}_2 \in \mathbb{R}^{(N-1)q(T+1)}$ and let $c_0 \in \mathbb{R}$ and $r_0 > 0$ be the smallest number such that

$$J(\boldsymbol{\xi}^0, \varsigma(\boldsymbol{\xi}^0)) + {\boldsymbol{\vartheta}_1^0}^\top P \boldsymbol{\vartheta}_1^0 \le c_0, \qquad \|\boldsymbol{\vartheta}_2^0\| \le r_0. \tag{3.82}$$

Then, for any $c \ge c_0$, the corresponding level set $\Omega_1^c \subset \mathbb{R}^{(n+m)T} \times \mathbb{R}^{(N-1)q(T+1)}$ is denoted as

$$\Omega_1^c := \{(\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \mid J(\boldsymbol{\xi}^0, \varsigma(\boldsymbol{\xi}^0)) + {\boldsymbol{\vartheta}_1^0}^\top P_1 \boldsymbol{\vartheta}_1^0 \le c\}. \tag{3.83}$$

We underline that, as before, with a slight abuse of notation, we joined the first two arguments of J to adapt it for the notation using $\boldsymbol{\xi}$ instead of $(\boldsymbol{\alpha}, \boldsymbol{\mu})$. Since $J(\boldsymbol{\xi}, \varsigma(\boldsymbol{\xi}))$ is the composition of $\mathcal{C}^2$ functions (see Section 3.3.2 and Assumption 3.3), then it is $\mathcal{C}^2$. Specifically, on a compact set the function itself and its derivatives, up to the second order, are bounded. Moreover, being the cost function radially unbounded (cf. Assumption 3.3), also the composition $J(\boldsymbol{\xi}, \varsigma(\boldsymbol{\xi}))$ is radially unbounded. Hence, each level set of J is compact. The same holds for $J(\boldsymbol{\xi}) + \boldsymbol{\vartheta}_1^\top P_1 \boldsymbol{\vartheta}_1$. Let us define

$$\begin{aligned}
M_1 &:= \max\{\|\boldsymbol{\xi}\| \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega_1^c\}\\
M_2 &:= \max\{\|\boldsymbol{\vartheta}_1\| \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega_1^c\}\\
M_3 &:= \max\{\|\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}, \varsigma(\boldsymbol{\xi}) + R\boldsymbol{\vartheta}_1)\| \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega_1^c\}\\
M_4 &:= \max\{\|\nabla \bar{\boldsymbol{\varphi}}(\boldsymbol{\xi})\| \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega_1^c\}\\
M_5 &:= \max\{\|u_1(\boldsymbol{\xi}, \boldsymbol{\vartheta})\| \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega_1^c\}\\
M_6 &:= \max\{\|\Gamma_{\varsigma}(\boldsymbol{\xi}, \mathbf{1}\varsigma(\boldsymbol{\xi}) + R\boldsymbol{\vartheta}_1)\| \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega_1^c\},
\end{aligned} \tag{3.84}$$

whose existence is guaranteed since each continuous function on a compact set is bounded, being J, V $\mathcal{C}^2$ functions on the compact set Ω_1^c. Now, given $r > 0$, we define

$$M_7 := M_6 + M_4 \|R\| r,$$

and the related sets $B^c_{\boldsymbol{\xi}} \subset \mathbb{R}^{(n+m)T}$, $B^r_2 \subset \mathbb{R}^{(N-1)q(T+1)}$ given by

$$B^c_{\boldsymbol{\xi}} := \{\boldsymbol{\xi} \in \mathbb{R}^{(n+m)T} \mid \|\boldsymbol{\xi}\| \leq M_1 + M_7\}. \tag{3.85a}$$

$$B^r_2 := \{\boldsymbol{\vartheta}_2 \in \mathbb{R}^{(N-1)q(T+1)} \mid \|\boldsymbol{\vartheta}_2\| \leq r\}. \tag{3.85b}$$

Now, assume $(\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega^c_1$ and $\boldsymbol{\vartheta}_2 \in B^r_2$. Then, the update (3.69a) allows us to write the bound

$$\begin{aligned}\left\|\boldsymbol{\xi}^+\right\| &\leq \|\boldsymbol{\xi}\| + \gamma \left\|\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}, \varsigma(\boldsymbol{\xi}) + R\boldsymbol{\vartheta}_1)\right\| + \gamma \left\|\nabla\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi})R\boldsymbol{\vartheta}_2\right\| \\ &\overset{(a)}{\leq} M_1 + \gamma(\underbrace{M_6 + M_4 \left\|R\right\| r}_{M_7}),\end{aligned} \tag{3.86}$$

where in (a) we exploited the bounds expressed in (3.84). Thus, by combining the inequality (3.86) and the fact that $\gamma \leq 1$, it holds $\left\|\boldsymbol{\xi}^+\right\| \leq M_1 + M_7$, namely it holds $\boldsymbol{\xi}^+ \in B^c_{\boldsymbol{\xi}}$ (cf. (3.85a)). Now, let us define

$$B^c_1 := \left\{(\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \mathbb{R}^{(n+m)T} \times \mathbb{R}^{(N-1)q(T+1)} \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega^c_1, \boldsymbol{\xi} \in B^c_{\boldsymbol{\xi}}\right\} \subseteq \Omega^c_1.$$

From this definition, it holds $(\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in B^c_1$ for any $(\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega^c_1$. Moreover, such a definition guarantees that, if $(\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in \Omega^c_1$, then it holds $\boldsymbol{\xi} \in B^c_{\boldsymbol{\xi}}$. Now, let $\mathcal{H}_\varsigma : \mathbb{R}^{(n+m)T} \times \mathbb{R}^{Nq(T+1)} \to \mathbb{R}^{(n+m)T}$ be the operator defined as

$$\mathcal{H}_\varsigma(\boldsymbol{\xi}, \mathrm{s}) = \begin{bmatrix} \nabla^2_2 J_1(\boldsymbol{\xi}_1, s_1) \\ \vdots \\ \nabla^2_2 J_N(\boldsymbol{\xi}_N, s_N) \end{bmatrix}, \tag{3.87}$$

where $\boldsymbol{\xi} := \mathrm{col}(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N)$ and $\mathrm{s} := \mathrm{col}(\mathrm{s}_1, \ldots, \mathrm{s}_N)$, with $\boldsymbol{\xi}_i \in \mathbb{R}^{(n_i+m_i)T}$ and $\mathrm{s}_i \in \mathbb{R}^{q(T+1)}$ for all $i \in \{1, \ldots, N\}$. Since ∇J, Γ_ς, and $\bar{\boldsymbol{\varphi}}$ are compositions of continuous and differentiable functions (cf.Assumption 3.3) and B_1 is a compact set, we can define the finite constants L, L_2 and L_3 as

$$\begin{aligned} L &:= \max\{\|\nabla^2 J(\boldsymbol{\xi}, \varsigma(\boldsymbol{\xi}))\| \mid \boldsymbol{\xi} \in B^c_{\boldsymbol{\xi}}\} \\ L_2 &:= \max\{\|\mathcal{H}_\varsigma(\boldsymbol{\xi}, \mathbf{1}\varsigma(\boldsymbol{\xi}) + R\boldsymbol{\vartheta}_1)\| \mid (\boldsymbol{\xi}, \boldsymbol{\vartheta}_1) \in B^c_1\} \\ L_3 &:= \max\{\|\nabla\bar{\boldsymbol{\varphi}}(\boldsymbol{\xi})\| \mid \boldsymbol{\xi} \in B^c_{\boldsymbol{\xi}}\}. \end{aligned}$$

Hence, $\nabla J(\cdot, \cdot)$, Γ_ς, and $\bar{\boldsymbol{\varphi}}$ are Lipschitz with constants L, L_2, and L_3 , respectively, over B^c_1.

Step (ii): LaSalle-based analysis Let us define $\mathbf{c} := c_0 + \|P_2\| r_0^2$ and assume that $(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}_1^k) \in \Omega_1^{\mathbf{c}}$ and $\boldsymbol{\vartheta}_2^k \in B_2^{r_0}$ (see (3.83) for the definitions of these sets). Such an assumption will be verified later by a proper selection of γ. Thus, for the arguments above, it holds $\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1} \in B_{\boldsymbol{\xi}}^{\mathbf{c}}$. Thus, by using the Lipschitz continuity over $B_1^{\mathbf{c}}$ of the functions $\Gamma_{\varsigma}(\cdot,\cdot)$ and $\bar{\varphi}(\cdot)$, we can bound the norm of $u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)$ (cf. (3.67)) as

$$\left\|u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)\right\| \leq L_2 \|R\| \left\|\boldsymbol{\vartheta}_1^k\right\|. \tag{3.88}$$

With the same arguments, we can bound the norm of $u_2(\boldsymbol{\xi}^{k+1}, \boldsymbol{\xi}^k)$ (cf. (3.67)) as

$$\begin{aligned}\left\|u_2(\boldsymbol{\xi}^{k+1}, \boldsymbol{\xi}^k)\right\| &\leq (L_3 + L_2 \left\|\mathbf{1}\mathbf{1}^\top/N\right\| L_3) \left\|\boldsymbol{\xi}^{k+1} - \boldsymbol{\xi}^k\right\| \\ &\overset{(a)}{\leq} \gamma c_1 \left\|\Gamma_{\boldsymbol{\xi}}(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k) + \begin{bmatrix} R & 0 \end{bmatrix} \boldsymbol{\vartheta}^k)\right\| + \gamma c_1 L_3 \|R\| \left\|\boldsymbol{\vartheta}_2^k\right\| \\ &\overset{(b)}{\leq} \gamma c_1 \|\nabla J(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k))\| + \gamma c_2 \|\boldsymbol{\vartheta}_1^k\| + \gamma c_3 \|\boldsymbol{\vartheta}_2^k\|,\end{aligned}$$

where in (a) we defined $c_1 := L_3 + L_2 \left\|\mathbf{1}\mathbf{1}^\top/N\right\| L_3$, used the update (3.69a), the triangle inequality, the Cauchy-Schwarz inequality, and the Lipschitz continuity of $\nabla\varphi$ over $\Omega_1^{\mathbf{c}}$, while in (b) we added and subtracted $\nabla J(\boldsymbol{\xi}^k, \mathbf{1}\varsigma(\boldsymbol{\xi}^k))$ within the first norm, we applied the triangle inequality and the Lipschitz continuity of $\Gamma_{\boldsymbol{\xi}}$ over $\Omega_V^{\mathbf{c}}$, and defined $c_2 := c_1 L_2$ and $c_3 := c_1 L_3$. With these results at hand, we use the function $V(\cdot,\cdot)$ defined in (3.80) as a candidate Lyapunov function. By evaluating $\Delta V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) := V(\boldsymbol{\xi}^{k+1}, \boldsymbol{\vartheta}^{k+1}) - V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)$ along the trajectories of system (3.69), we obtain

$$\begin{aligned}\Delta V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) &= J(\boldsymbol{\xi}^{k+1}, \varsigma(\boldsymbol{\xi}^k)) - J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) - {\boldsymbol{\vartheta}^k}^\top (P - \mathcal{A}^\top P \mathcal{A}) \boldsymbol{\vartheta}^k + 2{\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P \mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \\ &\quad + 2{\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P \mathcal{B}_2 u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1}) + 2u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)^\top \mathcal{B}_1^\top P \mathcal{B}_2 u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1}) \\ &\quad + u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)^\top \mathcal{B}_1^\top P \mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) + u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1})^\top \mathcal{B}_2^\top P \mathcal{B}_2 u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1}).\end{aligned} \tag{3.89}$$

Now, we consider the term ${\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P \mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)$ and exploit the structure of $\mathcal{A}^\top P \mathcal{B}_1$ to write

$$\begin{aligned}{\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P \mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) &= p{\boldsymbol{\vartheta}_2^k}^\top (R^\top W R)^\top P_2 R^\top u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \\ &\overset{(a)}{\leq} p \left\|(R^\top W R)^\top P_2 R^\top\right\| L_2 \|R\| \|\boldsymbol{\vartheta}_2^k\| \|\boldsymbol{\vartheta}_1^k\| \\ &\overset{(b)}{\leq} p c_4 \|\boldsymbol{\vartheta}_2^k\| \|\boldsymbol{\vartheta}_1^k\|,\end{aligned} \tag{3.90}$$

where in (a) we apply the Cauchy-Schwarz inequality and the bound (3.88), and in (b) we set $c_4 := \left\|(R^\top W R)^\top P_2 R^\top\right\| L_2 \|R\|$. Now, consider the term ${\boldsymbol{\vartheta}^k}^\top (P - \mathcal{A}^\top P \mathcal{A}) \boldsymbol{\vartheta}^k +$

$2{\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P\mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)$ and use the results (3.81) and (3.90) to write

$$\begin{aligned}
&{\boldsymbol{\vartheta}^k}^\top (P - \mathcal{A}^\top P\mathcal{A})\boldsymbol{\vartheta}^k + 2{\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P\mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)\\
&\qquad\qquad \le -{\boldsymbol{\vartheta}_1^k}^\top Q_1 \boldsymbol{\vartheta}_1^k - p{\boldsymbol{\vartheta}_2^k}^\top Q_2 \boldsymbol{\vartheta}_2^k + 2c_4 p\|\boldsymbol{\vartheta}_2^k\|\|\boldsymbol{\vartheta}_1^k\|\\
&\qquad\qquad \overset{(a)}{\le} - \begin{bmatrix}\|\boldsymbol{\vartheta}_1^k\| & \|\boldsymbol{\vartheta}_2^k\|\end{bmatrix} \underbrace{\begin{bmatrix} q_1 & -pc_4 \\ -pc_4 & pq_2 \end{bmatrix}}_{\tilde{Q}} \begin{bmatrix}\|\boldsymbol{\vartheta}_1^k\| \\ \|\boldsymbol{\vartheta}_2^k\|\end{bmatrix},
\end{aligned} \tag{3.91}$$

where in (a) we have rearranged the terms in a matrix form by introducing the smallest (positive) eigenvalues q_1 and q_2 of the matrices Q_1 and Q_2, respectively. By using the Sylvester Criterion, we know that if we pick p such that $p < \min\{q_1 q_2/c_4^2, 1\}$, then the symmetric matrix $\tilde{Q} \in \mathbb{R}^{2\times 2}$ introduced in (3.91) to bound (3.91) as

$${\boldsymbol{\vartheta}^k}^\top (P - \mathcal{A}^\top P\mathcal{A})\boldsymbol{\vartheta}^k + 2{\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P\mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \le -\tilde{q}\|\boldsymbol{\vartheta}^k\|^2,$$

where the (positive) smallest eigenvalue $\tilde{q}$ of the matrix $\tilde{Q}$ has been introduced. Thus, by leveraging this inequality, we can upper bound (3.89) as

$$\begin{aligned}
\Delta V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) &\le J(\boldsymbol{\xi}^{k+1}, \varsigma(\boldsymbol{\xi}^k)) - J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) - \tilde{q}\|\boldsymbol{\vartheta}^k\|^2 + 2{\boldsymbol{\vartheta}^k}^\top \mathcal{A}^\top P\mathcal{B}_2 u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1})\\
&\quad + 2u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)^\top \mathcal{B}_1^\top P\mathcal{B}_2 u_2(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) + u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)^\top \mathcal{B}_1^\top P\mathcal{B}_1 u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)\\
&\quad + u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1})^\top \mathcal{B}_2^\top P\mathcal{B}_2 u_2(\boldsymbol{\xi}^k, \boldsymbol{\xi}^{k+1})\\
&\overset{(a)}{\le} J(\boldsymbol{\xi}^{k+1}, \varsigma(\boldsymbol{\xi}^k)) - J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) - \tilde{q}\|\boldsymbol{\vartheta}^k\|^2 + \gamma 2c_5\|\boldsymbol{\vartheta}^k\| \left\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\right\|\\
&\quad + \gamma c_6\|\boldsymbol{\vartheta}^k\|^2 + \gamma^2 c_7\|\boldsymbol{\vartheta}^k\|^2 + \gamma^2 c_8\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\|^2\\
&\quad + \gamma^2 c_9\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\|\|\boldsymbol{\vartheta}^k\|,
\end{aligned} \tag{3.92}$$

where in (b) the triangle inequality, the Cauchy-Schwarz inequality, the trivial relations $\|\boldsymbol{\vartheta}_1^k\| \le \|\boldsymbol{\vartheta}^k\|$ and $\|\boldsymbol{\vartheta}_2^k\| \le \|\boldsymbol{\vartheta}^k\|$ and the result (3.88) has been exploited to bound the terms. Moreover, we shortened the notation by introducing the constants

$$\begin{aligned}
c_5 &:= c_1\left(\left\|\mathcal{A}^\top P\mathcal{B}_2\right\| + L_2\|R\|\left\|G^\top PD\right\|\right)\\
c_6 &:= 2(c_2 + c_3)\left(\left\|\mathcal{A}^\top P\mathcal{B}_2\right\| + L_2\|R\|\left\|\mathcal{B}_1^\top P\mathcal{B}_2\right\|\right)\\
c_7 &:= \left\|\mathcal{B}_1^\top P\mathcal{B}_1\right\| {L_2}^2\|R\|^2 + \left\|\mathcal{B}_2^\top P\mathcal{B}_2\right\|(c_2^2 + c_3^2 + 2c_2c_3)\\
c_8 &:= \left\|\mathcal{B}_2^\top P\mathcal{B}_2\right\| c_1^2 \qquad c_9 := 2\left\|\mathcal{B}_2^\top P\mathcal{B}_2\right\| c_1(c_2 + c_3).
\end{aligned}$$

Now we consider the term $J(\boldsymbol{\xi}^{k+1}, \varsigma(\boldsymbol{\xi}^k)) - J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))$. By performing a Taylor expansion (see, e.g., [96, Theorem 2]) of $J(\boldsymbol{\xi}^{k+1}, \varsigma(\boldsymbol{\xi}^k))$ around the point $\boldsymbol{\xi}^k$ and by using the update (3.69a), we can write

$$\begin{aligned} J(\boldsymbol{\xi}^{k+1}, \varsigma(\boldsymbol{\xi}^k)) - J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) = &- \gamma \|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\|^2 - \gamma \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))^\top u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \\ &+ L_{\boldsymbol{\xi}^k,2}(-\gamma(\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k))), \end{aligned} \tag{3.93}$$

where we shortened the notation by using

$$u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) := [0 \ \nabla \phi(\boldsymbol{\xi}^k) R] \boldsymbol{\vartheta}^k + \nabla \bar{\boldsymbol{\varphi}}(\boldsymbol{\xi}^k) u_1(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k), \tag{3.94}$$

and the remainder term $L_{\boldsymbol{\xi}^k,2}(-\gamma(\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)))$ given by

$$\begin{aligned} &L_{\boldsymbol{\xi}^k,2}(-\gamma(\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k))) \\ &\quad := \sum_{|\alpha|=2} \partial^\alpha J \left(\boldsymbol{\xi}^k - \tau\gamma(\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)) \right) \frac{-\gamma(\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k))}{\alpha!}, \end{aligned}$$

for some $\tau \in (0,1)$. By Lipschitz continuity of the gradients of J, we know that $\partial^\alpha J(\boldsymbol{\xi}, \sigma(\boldsymbol{\xi})) \leq L$ for all $\boldsymbol{\xi} \in B_{\boldsymbol{\xi}}$ and $|\alpha| = 2$. Thus, by applying [96, Corollary 1], we can bound $L_{\boldsymbol{\xi}^k,2}(-\gamma(\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)))$ as

$$\begin{aligned} L_{\boldsymbol{\xi}^k,2}(-\gamma(\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k))) &\leq \gamma^2 \frac{L_2}{2} \left\| \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) + u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \right\|^2 \\ &\overset{(a)}{\leq} \gamma^2 c_{10} \left\| \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) \right\|^2 + \gamma^2 c_{11} \|\boldsymbol{\vartheta}^k\|^2 \\ &\quad + \gamma c_{12} \left\| \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) \right\| \|\boldsymbol{\vartheta}^k\|, \end{aligned} \tag{3.95}$$

where in (a) we exploited the square and applied the bound (3.88) and we introduce the constants $c_{10} := \frac{L_2}{2}$, $c_{11} := \frac{L_2}{2}(L_1 + L_1{}^2)^2 \|R\|^2$, $c_{12} := \frac{L_2}{2}(L_1 + L_1{}^2)\|R\|$. Hence, we can use (3.95) to bound (3.93) as

$$\begin{aligned} J(\boldsymbol{\xi}^{k+1}, \varsigma(\boldsymbol{\xi}^k)) - J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) \leq &- \gamma \left\| \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) \right\|^2 - \gamma \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))^\top u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \\ &+ \gamma^2 c_{10} \left\| \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) \right\|^2 \\ &+ \gamma^2 c_{11} \|\boldsymbol{\vartheta}^k\|^2 + \gamma c_{12} \left\| \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k)) \right\| \|\boldsymbol{\vartheta}^k\| \\ \overset{(a)}{\leq} &- \gamma \|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\|^2 + \gamma^2 c_{10} \|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\|^2 \\ &+ (\gamma c_{13} + \gamma^2 c_{11}) \|\boldsymbol{\vartheta}^k\|^2 \\ &+ \gamma c_{12} \|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\| \|\boldsymbol{\vartheta}^k\|, \end{aligned} \tag{3.96}$$

where in (a) we use the local Lipschitz continuity of $\bar{\boldsymbol{\varphi}}$, the Cauchy-Schwarz inequality, and the result (3.88) to bound the term $-\gamma \nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))^\top u_4^k(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)$ and we set $c_{13} :=$

$(L_1 + L_1{}^2)\|R\|$. Now, we can use (3.96) to upper bound (3.92) as

$$\begin{aligned}
\Delta V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \leq & -\gamma \left\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\right\|^2 + \gamma^2 c_{10} \left\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\right\|^2 \\
& + (\gamma c_{13} + \gamma^2 c_{11})\|\boldsymbol{\vartheta}^k\|^2 + \gamma c_{12} \left\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\right\| \|\boldsymbol{\vartheta}^k\| \\
& - \tilde{q}\|\boldsymbol{\vartheta}^k\|^2 + \gamma 2 c_5 \|\boldsymbol{\vartheta}^k\| \left\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\right\| + \gamma c_6 \|\boldsymbol{\vartheta}^k\|^2 \\
& + \gamma^2 c_7 \|\boldsymbol{\vartheta}^k\|^2 + \gamma^2 c_8 \left\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\right\|^2 + \gamma^2 c_9 \left\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\right\| \|\boldsymbol{\vartheta}^k\| \\
\overset{(a)}{\leq} & - \begin{bmatrix} \|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\| \\ \|\boldsymbol{\vartheta}^k\| \end{bmatrix}^\top M \begin{bmatrix} \|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\| \\ \|\boldsymbol{\vartheta}^k\| \end{bmatrix},
\end{aligned} \tag{3.97}$$

where in (a) we rearranged the inequality in a matrix form with

$$M := \begin{bmatrix} \gamma - \gamma^2 c_{14} & -\gamma c_{15} - \gamma^2 c_{16} \\ -\gamma c_{15} - \gamma^2 c_{16} & \tilde{q} - \gamma c_{17} - \gamma^2 c_{18} \end{bmatrix},$$

and

$$\begin{aligned}
& c_{14} := c_{10} + c_8 \qquad c_{15} := \frac{c_{12} + c_9}{2} + c_5 \qquad c_{16} := c_9 \\
& c_{17} := c_{13} + c_6 \qquad c_{18} := c_{11} + c_7.
\end{aligned}$$

By Sylvester Criterion, we know that the matrix $M = M^\top \in \mathbb{R}^{2\times 2}$ introduced in (3.97) is positive definite if and only if its principal minors are positive, namely if it holds

$$\begin{cases} \gamma > & \gamma^2 c_1 4 \\ \gamma\tilde{q} > & (\gamma c_{15} + \gamma^2 c_{16})^2 + \gamma^2(c_{17} + c_{14}\tilde{q}) - \gamma^3(c_{14}c_{18} + c_{14}c_{17}) - \gamma^4 c_{14}c_{18}. \end{cases} \tag{3.98}$$

We notice the terms on the right-hand sides of (3.98) are linear in γ, while the left ones have higher orders. Then, there exists $\bar{\gamma} > 0$ such that for any $\gamma \in (0, \bar{\gamma})$ both conditions (3.98) are satisfied leading to the positive definiteness of M which allows us to rewrite (3.97)

$$\Delta V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \leq -\mathrm{m}(\|\nabla J(\boldsymbol{\xi}^k, \varsigma(\boldsymbol{\xi}^k))\|^2 + \|\boldsymbol{\vartheta}^k\|^2), \tag{3.99}$$

where m is the smallest (positive) eigenvalue of the matrix M. Notice that, inequality (3.99) implies that, for all $(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}_1^k, \boldsymbol{\vartheta}_2^k) \in \Omega_1^{\mathrm{c}} \times B_2^{r_0}$, it holds

$$V(\boldsymbol{\xi}^{k+1}, \boldsymbol{\vartheta}^{k+1}) \leq V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k). \tag{3.100}$$

By using (i) $p{\boldsymbol{\vartheta}_2^k}^\top P_2 \boldsymbol{\vartheta}_2^k \geq 0$ for any $\boldsymbol{\vartheta}^k$ and (ii) $p \geq 1$, the result (3.100) leads to

$$J(\boldsymbol{\xi}^{k+1}) + {\boldsymbol{\vartheta}_1^{k+1}}^\top P_1 \boldsymbol{\vartheta}_1^{k+1} \leq J(\boldsymbol{\xi}^k) + {\boldsymbol{\vartheta}_1^k}^\top P_1 \boldsymbol{\vartheta}_1^k + \|P_2\| r^2,$$

which guarantees $(\boldsymbol{\xi}^{k+1}, \boldsymbol{\vartheta}_1^{k+1}) \in \Omega_1^{\mathrm{c}}$. This fact, combined with the variables' initialization (cf. (3.82)), guarantees that $(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}_1^k) \in \Omega_1^{\mathrm{c}}$ for all $k \geq 0$, which, in turns, allows us to claim that (3.99) is verified for all $k \geq 0$. However, we point out that the inequality (3.99) does not implies the negative definiteness of $\Delta V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k)$ because the right-hand side term of (3.99) is null on the subspace $E \subset \mathbb{R}^{(n+m)T} + \times \mathbb{R}^{2(N-1)q(T+1)}$ defined as

$$E := \{\mathrm{col}(\boldsymbol{\xi}, \boldsymbol{\vartheta}) \in \mathbb{R}^{(n+m)T} + \times \mathbb{R}^{2(N-1)q(T+1)} \mid \|\nabla J(\boldsymbol{\xi}, \varsigma(\boldsymbol{\xi}))\| = 0, \boldsymbol{\vartheta} = 0\}. \tag{3.101}$$

Now, we study the system (3.69) restricted to the subspace E defined in (3.101). It holds

$$\boldsymbol{\xi}^{k+1}\Big|_{(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \in E} = \boldsymbol{\xi}^k, \qquad \boldsymbol{\vartheta}^{k+1}\Big|_{(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \in E} = \boldsymbol{\vartheta}^k. \tag{3.102a}$$

By inspecting system (3.102), we can conclude that the E is invariant for system (3.69). Thus, we can conclude by LaSalle's Invariance Principle (cf. [Theorem 3.7] [100]) that

$$\lim_{k \to \infty} w\left(\begin{bmatrix} \boldsymbol{\xi}^k \\ \boldsymbol{\vartheta}^k \end{bmatrix}, E\right) = 0.$$

Turning out to the original coordinates $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{s}, \mathbf{y})$, it follows that the trajectories of the system (3.66), i.e., the sequence produced by Algorithm 10, converge within the subspace

$$\mathcal{X} := \Big\{(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star, \mathbf{s}^\star, \mathbf{y}^\star) \mid \nabla J(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star, \varsigma(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)) = 0, \\ \mathbf{s}^\star = \frac{\mathbf{1}^\top}{N} \bar{\varphi}(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star),\ \mathbf{y}^\star = \mathbf{1}^\top \nabla J_a(\varsigma(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star))\Big\}. \tag{3.103}$$

Namely, the sequence $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}$ converges to the set of stationary points of $J(\cdot, \cdot, \varsigma(\cdot, \cdot))$.

Step (iii): first-order necessary conditions for optimality This final step builds on the proof of Theorem 3.1 and extends it to the multi-agent setting. Let us consider any point $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ such that $\nabla J(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star, \varsigma(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)) = 0$. Then, let us consider the corresponding state-input trajectories $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ computed projecting $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ and the costate vectors $\boldsymbol{\lambda}^\star \in \mathbb{R}^{nT}$. Specifically, for all $i = 1, \ldots, N$, we calculate $(\boldsymbol{x}_i^\star, \boldsymbol{u}_i^\star) = \mathcal{P}_i(\boldsymbol{\alpha}_i^\star, \boldsymbol{\mu}_i^\star)$ via (3.48). As for $\boldsymbol{\lambda}^\star \in \mathbb{R}^{nT}$, each $\boldsymbol{\lambda}_i^\star \in \mathbb{R}^{n_i T}$ is computed via (3.46b). Next, we show that the tuple $(\boldsymbol{x}^\star, \boldsymbol{u}^\star, \boldsymbol{\lambda}^\star)$ satisfies the first order necessary optimality conditions for the optimal control problem (3.43). Hence, for all $t \in [0, T-1]$, we introduce the Hamiltonian function of problem (3.43) given by

$$H_t(x_t, u_t, \lambda_{t+1}) = \sum_{i=1}^{N} H_{i,t}(x_{i,t}, u_{i,t}, \lambda_{i,t+1}) \tag{3.104}$$

where

$$H_{i,t}(x_{i,t}, u_{i,t}, \lambda_{i,t+1}) := \ell_{i,t}(x_{i,t}, u_{i,t}) + g_{i,t}(\sigma_t(x_t, u_t)) + f_i(x_{i,t}, u_{i,t})^\top \lambda_{i,t+1}.$$

Next we show that, for all $i = 1, \dots, N$,

$$\begin{bmatrix} \nabla_2 H_{1,t}(x^\star_{1,t}, u^\star_{1,t}, \lambda^\star_{1,t+1}) \\ \vdots \\ \nabla_2 H_{N,t}(x^\star_{N,t}, u^\star_{N,t}, \lambda^\star_{N,t+1}) \end{bmatrix} = 0,$$

and

$$\lambda_{i,t}{}^\star = \nabla_1 H_{i,t}(x^\star_{i,t}, u^\star_{i,t}, \lambda^\star_{t+1}), \tag{3.105}$$

with terminal condition $\lambda^\star_{i,T} = \nabla \ell_{i,T}(x^\star_{i,T}, \sigma_T(x^\star_T))$. In light of the projection-operator step (3.48), the point $(\boldsymbol{x}^\star_i, \boldsymbol{u}^\star_i)$ satisfies the dynamics (3.43b) by construction, i.e., it is a trajectory. Now, let us define the shorthand for the linearization of the cost and the dynamics about the trajectory $(\boldsymbol{x}^*_i, \boldsymbol{u}^*_i)$

$$A^\star_{i,t} := \nabla_1 f_i(x^\star_{i,t}, u^\star_{i,t})^\top, \;\; B^\star_{i,t} := \nabla_2 f_i(x^\star_{i,t}, u^\star_{i,t})^\top.$$

Notice that, since the sequence $\{\mathbf{s}^k, \mathbf{y}^k\}$ generated by Algorithm 10 converges to the subspace defined by (3.103), namely, for all $i = 1, \dots, N$ we have that the proxies in (3.46a) are equal to

$$a_{i,t} = \nabla_1 \ell_{i,t}(x^\star_{i,t}, u^\star_{i,t}, \sigma_t(x^\star_{i,t}, u^\star_{i,t})) + \nabla_1 \varphi_{i,t}(x^\star_{i,t}, u_{i,t}) \sum_{j=1}^{N} \nabla_3 \ell_{j,t}(x^\star_{j,t}, u^\star_{j,t}, \sigma_t(x^\star_t, u^\star_t))$$

$$b_{i,t} = \nabla_2 \ell_{i,t}(x^\star_{i,t}, u^\star_{i,t}) + \nabla_2 \varphi_{i,t}(x^\star_{i,t}, u_{i,t}) \sum_{j=1}^{N} \nabla_3 \ell_{j,t}(x^\star_{j,t}, u^\star_{j,t}, \sigma_t(x^\star_t, u^\star_t))$$

and

$$a_{i,T} = \nabla \ell_{i,T}(x^\star_{i,T}, \sigma_T(x^\star_T)) + \nabla \varphi_{i,T}(x^\star_{i,T}) \sum_{j=1}^{N} \nabla_2 \ell_{j,T}(x^\star_{j,T}, \sigma_T(x^\star_T)).$$

Then, we can define $\boldsymbol{\lambda}_i^\star$ as the stack of the costate vectors $\lambda_{i,t}^\star \in \mathbb{R}^n$, obtained from the adjoint equation (3.46b) evaluated at $(\boldsymbol{\alpha}_i^\star, \boldsymbol{\mu}_i^\star)$, i.e., for all $t \in [T-1, 0]$

$$\lambda_{i,t}^\star = \left(A_{i,t}^\star - B_{i,t}^\star K_{i,t}^\star\right)^\top \lambda_{i,t+1}^\star + a_{i,t} - K_{i,t}^{\star,\top} b_{i,t} \tag{3.106}$$

with terminal condition $\lambda_{i,T}^\star = a_{i,T}$. Equation (3.106) corresponds to the gradient with respect to $x_{i,t}$ of the Hamiltonian evaluated along the trajectory $(\boldsymbol{x}_i^\star, \boldsymbol{u}_i^\star)$, i.e., the first order necessary condition for optimality (3.105) holds by construction. Now, we recall that

$$\nabla_2 H_{i,t}(x_{i,t}^\star, u_{i,t}^\star, \lambda_{i,t+1}^\star) = b_{i,t} + B_{i,t}^{\star\top} \lambda_{i,t+1}^\star, \tag{3.107}$$

for all $i \in \{1, \ldots, N\}$ and $t \in [0, T-1]$. The right hand side of (3.107) corresponds to the derivative of $J(\cdot, \cdot, \varsigma(\cdot, \cdot))$ with respect to $\mu_{i,t}$ evaluated at $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$. Notice that, the costate $\boldsymbol{\lambda}_i^\star$, solution of (3.106), coincides with the Lagrange multipliers associated to the dynamics constraints at $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$. In light of the first part of the proof, this term is equal to zero. Therefore, the first-order necessary conditions for optimality are satisfied by the trajectory $(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*)$, for all i. Hence, for any point $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ such that $\nabla J((\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star), \varsigma(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)) = 0$, it holds that the corresponding point $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ is such that $(\boldsymbol{x}^\star, \boldsymbol{u}^\star) \in \Xi^\star$. This concludes the proof.

Notice that, any stationary state-input curve $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ corresponds to a trajectory $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ satisfying the first-order optimality conditions for (3.43).

Remark 3.4. If the reduced cost $J(\cdot, \cdot, \varsigma(\cdot, \cdot))$ (cf. (3.54)) is strongly convex, then problem (3.54) has a unique minimizer $\xi^\star \in \mathbb{R}^{(n+m)T}$. Thus, in this case, one can apply the well-known Polyak-Lojasiewicz (PL) inequality to further bound the right-hand side of (3.99) as

$$\Delta V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k) \leq -\tilde{\mathrm{m}} V(\boldsymbol{\xi}^k, \boldsymbol{\vartheta}^k), \tag{3.108}$$

for some $\tilde{\mathrm{m}} > 0$ depending on m and the strong convexity parameter. In turn, global exponential stability of $\mathrm{col}(\xi^\star, 0)$ holds for (3.69) and thus linear convergence of Algorithm 10 toward the (unique) solution of optimal control problem (3.43). Notice that PL inequality holds true in the neighborhood of local minima of (3.43) that satisfy the second order sufficient conditions for optimality. △

3.3.3 Numerical Simulations

In this section, we show simulations of the proposed algorithm in a multi-agent cooperative robotics setting. We consider a team of $N = 50$ quadrotors in a robotic surveillance scenario. Each robot has to patrol a pre-defined area trying to stay as close as possible to a given reference position while keeping the formation barycenter over a (moving) target. Each quadrotor i has a continuous-time dynamics described by

$$\ddot{p}_{x,i} = \frac{u_{i,1}+u_{i,2}}{M_i}\sin\theta_i \qquad \ddot{p}_{y,i} = \frac{u_{i,1}+u_{i,2}}{M_i}\cos\theta_i - g$$
$$\ddot{\theta}_i = \frac{L_i}{2J_i}(u_{i,1} - u_{i,2})$$

where $p_{x,i}, p_{y,i}$ is the position of the robot in the $x-y$ plane, θ_i its orientation, $u_{i,1}$ and $u_{i,2}$ are the control inputs, i.e., the thrust applied to the right and left motor respectively. Agents have heterogeneous dynamics. The mechanical parameters are randomly assigned to each agent, i.e., for all i, $M_i \in \{1,5,10\}$ [kg], $L_i \in \{0.5, 1.0\}$ [m], while $J_i = M_i L_i^2/12$ [m $\cdot$ kg^2].

For each agent i, the discrete-time dynamics, with state $x_{i,t} \in \mathbb{R}^6$ with $x_{i,t} := (p_{x,i,t}, \dot{p}_{x,i,t}, p_{y,i,t}, \dot{p}_{y,i,t}, \theta_{i,t}, \dot{\theta}_{i,t})^\top \in \mathbb{R}^6$ and input $u_{i,t} := (u_{i,1,t}, u_{i,2,t}) \in \mathbb{R}^2$ is obtained via a Runge-Kutta integrator of order 4 with sampling period $\delta = 0.05$ seconds. Hence, we have $n_i = 6$, $m_i = 2$ and $n = 60$, $m = 20$. The time horizon is $t_f = 30$ $[s]$, resulting in $T = 600$ samples. The network modeled as a randomly generated Erdós-Rényi graph with edge probability $p = 0.5$. The weighted adjacency matrix $\mathcal{W}$ is generated via the Metropolis-Hastings rule using disropt [88]. Each quadrotor i wants to track its reference signal $(\boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}})$. The team also aims to maintain the formation barycenter, modeled via $\sigma_t(x_t, u_t)$, as close as possible to the target located at $b_t^{\text{target}} \in \mathbb{R}^2$ at time t. Each stage cost is defined as $\ell_{i,t}(x_{i,t}, u_{i,t}, \sigma_t(x_t)) = \|x_{i,t} - x_{i,t}^{\text{des}}\|_{Q_{i,t}}^2 + \|u_{i,t} - u_{i,t}^{\text{des}}\|_{R_{i,t}}^2 + \|\sigma(x_t) - b_t\|_{Q_{i,t}^\sigma}^2$ where $Q_{i,t} \in \mathbb{R}^{n_i \times n_i}$, $R_{i,t} \in \mathbb{R}^{m_i \times m_i}$, $Q_{i,t}^\sigma \in \mathbb{R}^{q \times q}$, and $\|\cdot\|_M$ denotes the L2 norm weighted by matrix M. The terminal cost is defined by $\ell_{i,T}(\cdot) = \|x_{i,T} - x_{i,T}^{\text{des}}\|_{Q_{i,f}}^2 + \|\sigma(x_T) - b_T\|_{Q_{i,f}^\sigma}^2$ where $Q_{i,f} \in \mathbb{R}^{n_i \times n_i}$, and $Q_{i,f}^\sigma \in \mathbb{R}^{q \times q}$. The aggregate variable is defined as $\sigma(x_t) := \sum_{i=1}^N H_i x_{i,t}$ where $H_i = [I_2 \;\; 0_{2\times 4}]$ for all $i = 1, \dots, N$. We set $Q_{i,t} = \text{diag}(10, 1, 10, 1, 0.01, 0.01)$, $Q_{i,f} = Q_{i,t}$, $R_{i,t} = \text{diag}(0.1, 0.1)$, $Q_{i,t}^\sigma = \text{diag}(100, 100)$, $Q_{i,f}^\sigma = 10\, Q_{i,t}^\sigma$. The reference signal consists of a randomly generated fixed position in the $x-y$ plane, while the reference input is the equilibrium thrust. Formally, we define $(x_{i,t}^{\text{des}}, u_{i,t}^{\text{des}})$ as $p_{x,i,t}^{\text{des}} = \bar{p}_x$, $p_{y,i,t}^{\text{des}} = \bar{p}_y$, $u_{i,t}^{\text{des}} = \frac{1}{2} M_i g \mathbf{1}_{m_i}$, where $\bar{p}_x$ and $\bar{p}_y$ are drawn randomly from an uniform distribution between $\bar{p}_{x,\min} = -5$, $\bar{p}_{x,\max} = 5$, and $\bar{p}_{y,\min} = -5$, $\bar{p}_{y,\max} = 5$, respectively. All the other reference states are set identically equal to 0. The target position in $b_t \in \mathbb{R}^2$ is defined via a second-order polynomial between the checkpoints $(0,0)$, $(1,1)$ and $(1,1.5)$ reached at the time $0, T/2$ and T respectively. Algorithm 10 is run with a fixed stepsize $\gamma = 10^{-4}$. The initial trajectory $(\boldsymbol{x}^0, \boldsymbol{u}^0)$ is chosen as the equilibrium trajectory at the local reference signal for each agent. The positions associated to the optimal trajectories $(\boldsymbol{x}_i^\star, \boldsymbol{u}_i^\star)$ of the 50 quadrotors are depicted in Fig. 3.11. In Fig. 3.12 (above and center) we compare the cost and the input trajectory of the distributed algorithm with the ones computed by a centralized solver. Finally, below, notice that the trackers converge to the desired value.

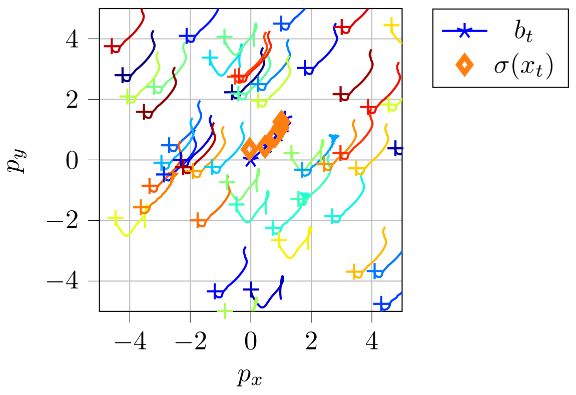


Figure 3.11: Quadrotor positions in the x-y plane: optimal trajectories $(\boldsymbol{x}_i^\star, \boldsymbol{u}_i^\star)$ (solid), reference position $\boldsymbol{x}_i^{\text{des}}$ (crosses), target position b_t (starred blue), formation barycenter $\sigma_t(x_t^\star)$ (orange diamonds).

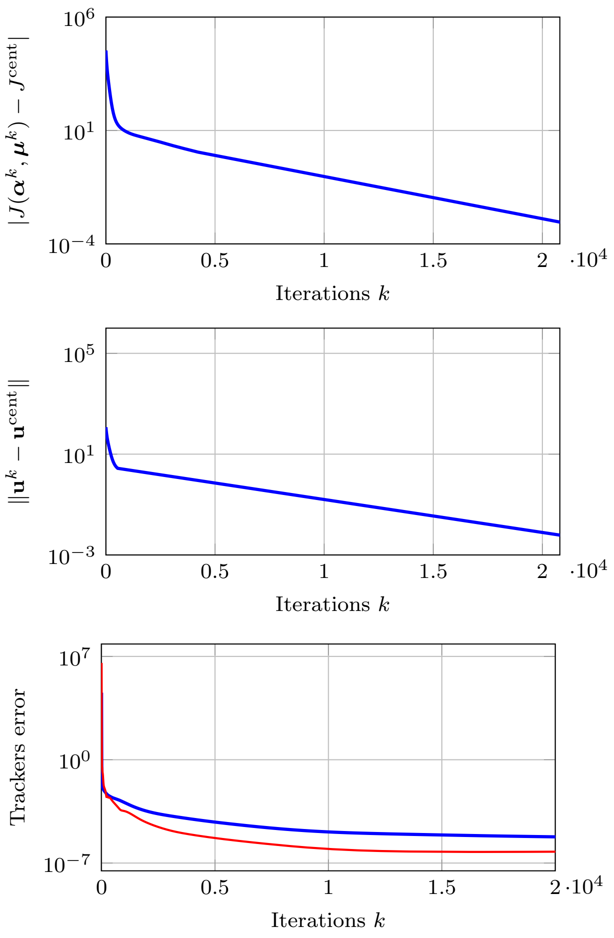


Figure 3.12: (Above) Error between the cost $J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ with respect to the cost J^{cent} computed by a centralized solver. (Center) Error between the input sequence $\boldsymbol{u}^k$ with respect to the input sequence $\boldsymbol{u}^{\text{cent}}$ computed by a centralized solver. (Below) Tracking error for the tracking variables $\mathbf{s}^k$, $\mathbf{y}^k$. In blue $\|\mathbf{s}^k - \mathbf{1}\sigma(\mathbf{x}^k, \mathbf{u}^k)\|$. In red $\|\mathbf{y}^k - \mathbf{1}\frac{1}{N}\sum_i^N G_{i,3}^k\|$ where we define $G_{i,3}^k := G_{i,3}(\boldsymbol{x}_i^k, \boldsymbol{u}_i^k, \boldsymbol{\sigma}(\boldsymbol{x}^k, \boldsymbol{u}^k))$.

3.4 Data-driven Optimal Control of Nonlinear Uncertain Dynamics

In this section we introduce a learning-driven optimal control algorithm designed to deal with discrete-time nonlinear systems described by dynamics

$$x_{t+1} = f(x_t, u_t), \qquad t \in [0, T-1], \tag{3.109}$$

with $[0, T-1] := \{0, 1, \dots, T-1\}$, where $x_t \in \mathbb{R}^{n_x}$ is the state, $u_t \in \mathbb{R}^{n_u}$ is the input at time t, and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the vector field modeling the dynamics. The initial condition $x_0 = x_{\text{init}}$ with $x_{\text{init}} \in \mathbb{R}^{n_x}$ given. The key challenge addressed is that the dynamics $f(\cdot)$ is composed by a nominal, known model (e.g., derived from first principles), and by an unknown part as

$$x_{t+1} = f_\star(x_t, u_t) + g_\star(x_t, u_t), \qquad t \in [0, T-1], \tag{3.110}$$

where $f_\star : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ models the nominal dynamics while $g_\star : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is unknown. We will refer both to (3.110), when we want to highlight the peculiar structure of the dynamics, and to (3.109) when we mean the real, though unknown, system. We investigate nonlinear optimal control problems in which we look for trajectories of the unknown system (3.109) that minimize a performance criterion defined over a fixed, time horizon $[0, T]$. Formally, we aim to solve the problem

$$\min_{\substack{x_1,\dots,x_T \\ u_0,\dots,u_{T-1}}} \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{3.111a}$$

$$\begin{aligned} \text{subj.to } & x_{t+1} = f_\star(x_t, u_t) + g_\star(x_t, u_t), \quad t \in [0, T-1], \\ & x_0 = x_{\text{init}}, \end{aligned} \tag{3.111b}$$

with stage costs $\ell_t : \mathbb{R}^{mT} \to \mathbb{R}$, for all t, and terminal cost $\ell_T : \mathbb{R}^{mT} \to \mathbb{R}$.

Assumption 3.5. *All functions $\ell_t(\cdot,\cdot)$, $\ell_T(\cdot)$ and $f(\cdot,\cdot)$ are twice continuously differentiable, i.e., they are of class $\mathcal{C}^2$ with respect to their arguments.* $\triangle$

The main challenge of the optimal control problem (3.111) is that the dynamics is only partially known and the presence of the unknown term calls for novel learning techniques to be combined into an optimal control scheme. Indeed, we point out that Algorithms like GoPRONTO, introduced in Section 3.2, cannot be implemented if the dynamics are partially unknown.

Let $\boldsymbol{x} := \text{col}(x_1, \dots, x_T) \in \mathbb{R}^{nT}$ and $\boldsymbol{u} := \text{col}(u_0, \dots, u_{T-1}) \in \mathbb{R}^{mT}$. A pair $(\boldsymbol{x}, \boldsymbol{u})$ is called a *trajectory* of the real system if its components satisfy the dynamics (3.109) for all $t \in [0, T-1]$, i.e., it satisfies Definition 3.1. Let us also recall the shorthand notation

$$a_t^k := \nabla_1 \ell_t(x_t^k, u_t^k), \qquad b_t^k := \nabla_2 \ell_t(x_t^k, u_t^k), \tag{3.112a}$$

$$A_t^k := \nabla_1 f(x_t^k, u_t^k)^\top, \qquad B_t^k := \nabla_2 f(x_t^k, u_t^k)^\top. \tag{3.112b}$$

3.4.1 Learning-driven Optimal Control via Gaussian Process Regression

In this section we present an optimization-based control strategy for nonlinear systems with partially unknown dynamics. We start by reviewing the key concepts behind Gaussian Process regression. Then, we show how to implement the iterative learning phase specifically tailored for dynamics learning and then we embed it in the novel optimal control strategy.

Review on Gaussian Process Regression

We now recall a popular nonparametric regression technique in machine learning based on Gaussian processes as presented, e.g., in [185]. It represents a powerful tool to infer from data a nonlinear vector-valued function $\varphi : \mathbb{R}^{n_z} \to \mathbb{R}$ describing a nonlinear map between the input z and its corresponding output $y = \varphi(z)$. We suppose to have access to a data-set

$$D := \Big((z^1, y^1), \dots, (z^H, y^H) \Big)$$

with each pair $(z^h, y^h) \in \mathbb{R}^{n_z} \times \in \mathbb{R}^{n_y}$ obtained as

$$y^h = \varphi(z^h) + \epsilon^h$$

where $\epsilon^h \in \mathbb{R}^{n_y}$ is a white Gaussian noise with covariance matrix $\sigma_\epsilon^2 I_{n_y}$.

In Gaussian process regression, we assume that values of the components φ_a, $a = 1, \dots, n_y$, of φ are drawn from independent Gaussian distributions. A GP is fully specified by a mean m function $m : \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ and a kernel covariance function $\kappa : R^{n_z} \times R^{n_z} \to \mathbb{R}$. To maintain computational feasibility, it is customary to train independent GPs for each component $\varphi_a(\cdot)$ of the vector field $\varphi(\cdot)$.

We choose the commonly adopted squared exponential kernel defined for any $z, z' \in \mathbb{R}^{n_z}$ as

$$\kappa(z, z') = \sigma_\varphi^2 \exp\left(-\frac{\|z - z'\|^2}{2L_\varphi} \right) \tag{3.113}$$

where $L_\varphi > 0$ denotes the signal length scale while σ_φ^2 is its variance.

Given the data-set D and the kernel covariance function, we introduce the Gram matrix $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{H \times H}$ whose (i, j)-th entry is $\boldsymbol{\mathcal{K}}_{hi} = \kappa(z^h, z^i)$ and the kernel vector $\boldsymbol{\kappa}(z) \in$

$\mathbb{R}^H$ at a generic $z \in \mathbb{R}^{n_z}$ with h-th component $\boldsymbol{\kappa}_h(z) = \kappa(z^h, z)$.

We specify a zero mean prior on $\varphi(\cdot)$ which means that no prior knowledge is available. Based on the previous definitions, the posterior predictive distribution of each component $\varphi_a(\cdot)$ conditioned on the data-set D at a given point z is Gaussian with mean and covariance

$$m_a(z) = \boldsymbol{\kappa}(z)^\top (\boldsymbol{\mathcal{K}} + \sigma_\epsilon^2 I_H)^{-1} Y_a \tag{3.114a}$$

$$\sigma^2(z) = \kappa(z, z) - \boldsymbol{\kappa}(z)^\top (\boldsymbol{\mathcal{K}} + \sigma_\epsilon^2 I_H)^{-1} \boldsymbol{\kappa}(z), \tag{3.114b}$$

for all $a = 1, \ldots, n_y$, where $Y_a \in \mathbb{R}^H$ collects only the a-th component of measurements $y^h \in Y$. The resulting posterior of the vector field $\varphi(\cdot)$ is

$$\varphi(z) \sim \mathcal{N}(m(z), \Sigma(z))$$

with

$$\begin{aligned} m(z) &:= \operatorname{col}\big(m_1(z), \ldots, m_{n_y}(z)\big) \\ \Sigma(z) &:= I_{n_y} \otimes \sigma^2(z) \end{aligned}$$

with $\otimes$ being the Kronecker product.

In the forthcoming strategy, we will also use the derivative of a GP, which since differentiation is a linear operator, is another Gaussian process [185]. Specifically, each component $m_a(\cdot)$ of $m(\cdot)$ has a posterior multivariate Gaussian

$$\nabla m_a(z) \sim \mathcal{N}(m'_a(z), \Sigma'_a(z))$$

with mean and covariance

$$m'_a(z) := \nabla \boldsymbol{\kappa}(z)^\top (\boldsymbol{\mathcal{K}} + \sigma_\epsilon^2 I_H)^{-1} Y_a \tag{3.115a}$$

$$\Sigma'_a(z) := \nabla^2 \kappa(z, z) - \nabla \boldsymbol{\kappa}(z)^\top (\boldsymbol{\mathcal{K}} + \sigma_\epsilon^2 I_H)^{-1} \nabla \boldsymbol{\kappa}(z) \tag{3.115b}$$

where $\nabla^2 \kappa(z, z) = \frac{\sigma_\varphi^2}{L_\varphi^2} I_H$ while the h-th row of the matrix $\nabla \boldsymbol{\kappa}(z) \in \mathbb{R}^{H \times n_z}$ is

$$\nabla \boldsymbol{\kappa}_h(z) = \frac{1}{L_\varphi^2} \kappa(z^h, z)\, (z^h - z)^\top.$$

Remark 3.5. A well-known issue in GP regression is the computational burden of the prediction which involves the inverse of the Gram matrix. Therefore, the computational complexity is cubic in the number of data points H. One can employ sparsification schemes to improve performance, but this is out of the scope of the present work. △

Dynamics Learning via Gaussian Process Regression

In this subsection we exploit Gaussian Process regression in the dynamics learning process. To this end, we start by modeling the unknown dynamics $g_\star(\cdot)$ in (3.110) using a Gaussian process. To ease the presentation and without loss of generality, we consider in this part a scalar system, i.e., with state $x \in \mathbb{R}$ and input $u \in \mathbb{R}$. Therefore, the unknown function is $g_\star(\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. As a consequence, also the mean and the variance of the GP are scalar functions. The results can be extended to vector-valued functions by considering one scalar GP for each component.

We consider observations of $g_\star(\cdot)$ taken in the following form. For each trajectory $(\boldsymbol{x}, \boldsymbol{u})$ of the real system (3.109), we define T observation pairs $((z^1, y^1), \ldots, (z^T, y^T))$ as

$$\begin{aligned} z^t &:= (x_t, u_t) \\ y^t &:= \underbrace{x_{t+1} - f_0(x_t, u_t)}_{\varphi(z^t)} + \epsilon^t \end{aligned} \tag{3.116}$$

for all $t \in [0, T-1]$, where ϵ^t is a white Gaussian noise with variance σ_ϵ^2.

The leading idea that will be explored in Section 3.4.2 is iteratively refine the GP approximation as the optimization process proceeds. Therefore, we suppose to arrive at a given iteration k with data-set $D^k = (Z^k, Y^k)$ collecting state-input trajectories explored up to k. We assume that $g(\cdot)$ is drawn from a GP prior, and we compute its posterior distribution which is Gaussian

$$g(x, u) \sim \mathcal{N}\Big(m^k(x, u), \Sigma^k(x, u\Big)$$

where the mean and the variance are computed as in (3.114).

Once the GP regression on $g(\cdot)$ has been posed, we choose to approximate the partially unknown dynamics (3.109) using a deterministic approach where $g(\cdot)$ is approximated by the posterior mean of the GP, i.e.,

$$x_{t+1} = f_0(x_t, u_t) + m^k(x_t, u_t), \quad t \in [0, T-1]. \tag{3.117}$$

Notice that the use of the squared exponential kernel (3.113) induces differentiability and boundedness properties to all functions represented by the GP [185]. Thus also the posterior mean function $m^k(x, u)$ is smooth and its derivative is a GP itself

$$\nabla m^k(x_t, u_t) \sim \mathcal{N}(m'^k(x_t, u_t), \Sigma'^k(x_t, u_t)) \tag{3.118}$$

with mean and covariance computed as in (3.115).

Clearly, we are also interested in providing a quantitative measure on the quality of the approximation made using the approximation described so far. We define the model estimation error $\Delta g_\star^k(x_t, u_t) \in \mathbb{R}$, for all k as

$$\Delta g_\star^k(x_t, u_t) := |g_\star(x_t, u_t) - m^k(x_t, u_t)|.$$

At each iteration k new data are collected and we expect that $\Delta g_\star^k(x_t, u_t)$ becomes smaller and smaller. This behavior strongly depends on the data informativity. Moreover, due to the stochastic nature of regression framework, the approximation error $\Delta g_\star^k(x_t, u_t)$ can be characterized only probabilistically. According to [218], the maximum distance of the true function $g_\star(\cdot)$ from the mean function $m^k(x_t, u_t)$ can be bounded with high-probability only for a restricted class of functions as stated in the next assumption.

Assumption 3.6. *The function $g_\star(\cdot)$ belongs to the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ associated to the kernel function $\kappa(\cdot,\cdot)$ in* (3.113)*. Moreover, it has bounded RKHS norm with respect to $\kappa(\cdot,\cdot)$, i.e., $\|g_\star(\cdot)\|_\kappa^2 \leq B_g$.* △

The concepts of RKHS and RKHS norm, required by Assumption 3.6, are intimately related to kernel methods and Gaussian process regression (see, e.g., [53]). Indeed, given a (positive definite) kernel function κ, the Moore-Aronszajn Theorem (cf. [11]) provides a one-to-one correspondence between κ and particular Hilbert spaces of functions $\mathcal{H}$ known as Reproducing Kernel Hilbert Spaces. That is, given a nonempty set $\mathcal{X}$ and an Hilbert space $\mathcal{H}$ of real functions $g : \mathcal{X} \to \mathbb{R}$, $\mathcal{H}$ is a Reproducing Kernel Hilbert Space endowed with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ and norm $\|g(\cdot)\| := \sqrt{\langle g(\cdot), g(\cdot)\rangle_\kappa}$, if there exists a kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that (i) $\kappa(\cdot,\cdot)$ has the reproducing property, i.e., $\langle g(\cdot), \kappa(\cdot, x)\rangle_{\mathcal{H}} = g(x)$ and, (ii) for every x, $\kappa(\cdot, x) \in \mathcal{H}$. A key property of RKHS is that, by Mercer's Theorem, the kernel function κ can be expressed as

$$\kappa(x, x') = \sum_{i=1}^{d} \beta_i \rho_i(x) \rho_i(x').$$

Then, it can be proved that (i) the basis functions $\rho_i : \mathcal{X} \to \mathbb{R}$, $i = 1, \ldots, d$ span the whole RKHS $\mathcal{H}$, (ii) each function $g \in \mathcal{H}$ can be expressed as

$$g(x) = \sum_{i=1}^{d} \alpha_i \rho_i(x)$$

and, (iii) given a function $g \in \mathcal{H}$, the RKHS norm with respect to κ, i.e., $\|g\|_\kappa^2 = \sqrt{\langle g(\cdot), g(\cdot)\rangle_\kappa}$ is defined via the inner product

$$\langle g(\cdot), g(\cdot)\rangle_\kappa = \sum_{i=1}^{d} \frac{\alpha_i^2}{\beta_i}. \tag{3.119}$$

Consequently, the functions in the RKHS $\mathcal{H}$ strongly inherit the properties of the kernel

κ and, in the context of our applications, it results that the unknown function $g_\star$ is searched for in the high-dimensional space represented by the RKHS induced by the kernel function κ.

We now recall a result based on [218, Thm. 6].

Lemma 3.4. *Let Assumption 3.6 hold and define*

$$\rho_\delta^k := \sqrt{2B_g + 300 \cdot \gamma^k \log^3((|D^k| + 1)/\delta)}$$

for all $\delta \in (0,1)$, where $|D^k|$ is the cardinality of the D^k, γ^k is the maximum mutual information[3] *that can be obtained about $g_\star(\cdot)$ from the data-set D^k. Then, for all $\delta \in (0,1)$, it holds*

$$\Pr\left\{|m^k(x,u) - g_\star(x,u)| \leq \rho_\delta^k \Sigma^k(x,u), \forall\, (x,u),\ k \in \mathbb{N}\right\} \geq 1 - \delta,$$

with $m^k(\cdot)$ and $\Sigma^k(\cdot)$ being the posterior mean and variance given data-set D^k. $\triangle$

If, moreover, we assume that the data points belong to a compact set, i.e., $(x,u) \in \mathbb{X} \times \mathbb{U}$ with $\mathbb{X}$ and $\mathbb{U}$ being compact sets, then a uniform bound on γ^k can be established [218]. Therefore, under the compactness requirement, Lemma 3.4 also provides a uniform quantitative bound on the error made by the approximation.

3.4.2 GP-Enhanced GoPRONTO: Algorithm Description and Analysis

Consider the optimal control problem (3.111) in which the dynamics is only partially known. We exploit the Gaussian process regression presented in Section 3.4.1 to include an iteratively refined approximation the unknown term $g_\star(\cdot)$ in the optimal control strategy. Specifically, we insert the learning procedure in the optimal control strategy and let both optimization and learning be concurrently performed as shown in Figure 3.13.

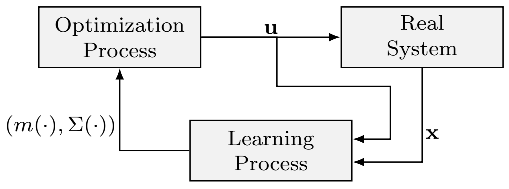


Figure 3.13: Scheme representing the information flow of the proposed learning-driven optimal control strategy.

Given the data-set D^k and the corresponding approximation at iteration k, the

[3]See [218, Sec. IV] for a definition of γ^k. Informally, it quantifies the quality of the data for the learning purposes.

optimal control problem to be solved can be written as

$$\begin{aligned} \min_{\substack{x_1,\dots,x_T\\u_0,\dots,u_{T-1}}} \quad & \sum_{t=0}^{T-1} \ell_t(x_t,u_t) + \ell_T(x_T) \\ \text{subj.to} \quad & x_{t+1} = f_0(x_t,u_t) + m^k(x_t,u_t), \;\; t \in [0, T-1], \\ & x_0 = x_{\text{init}}, \end{aligned} \tag{3.120}$$

in which the dynamics constraint includes the posterior mean $m^k(\cdot)$ in place of the unknown term $g_\star(\cdot)$. With problem formulation (3.120) in place, we can resort to the approach described previously Section 3.2.

First of all, we assume to have access to a nonlinear feedback policy in the form

$$\begin{aligned} u_t &= \mu_t + K_t(\alpha_t - x_t), \\ x_{t+1} &= f(x_t, u_t) \end{aligned} \tag{3.121}$$

which allows us to map state-input curves $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ into trajectories of system (3.109). The matrix $K_t \in \mathbb{R}^{n\times m}$ is known and designed to ensure local stability about a given state-input trajectory for the nonlinear dynamics (3.109). We can now rewrite problem (3.120) in its closed-loop reformulation, i.e.,

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\mu}} \quad & \sum_{t=0}^{T-1} \ell_t(x_t,u_t) + \ell_T(x_T) \\ \text{subj.to} \quad & \left.\begin{aligned} & x_{t+1} = f_0(x_t,u_t) + m^k(x_t,u_t) \\ & u_t = \mu_t + K_t(\alpha_t - x_t) \end{aligned}\right. \qquad t \in [0, T-1], \\ & x_0 = x_{\text{init}}. \end{aligned} \tag{3.122}$$

Problem (3.122) admits a reduced problem formulation, amenable to resolution via a first-order algorithm, e.g. Algorithm 6, (cf. Section 3.2). Notice that the descent direction in (3.12) cannot be computed as done earlier in Section 3.2 since $g_\star(\cdot)$ and, hence, A_t^k, B_t^k are not known. Therefore, we consistently adapt the adjoint system (3.12) to compute the descent direction based on the approximated dynamics. In fact, the Gaussian process approximation of the unknown vector field allows us to easily evaluate the linearization matrices about any point (x_t^k, u_t^k). Specifically, the derivative of the mean function is a Gaussian process itself. Thus, let the shorthands in (3.112b) be adapted as

$$A_t^k \longmapsto \hat{A}_t^k + m_x'^k(x_t^k, u_t^k), \qquad\qquad B_t^k \longmapsto \hat{B}_t^k + m_u'^k(x_t^k, u_t^k),$$

where $m_x'^k(x_t^k, u_t^k)$ and $m_u'^k(x_t^k, u_t^k)$ are the components of the mean function $m'^k(x_t^k, u_t^k)$ in (3.118) corresponding to the state and to the input, respectively. As for the cost linearization a_t^k and b_t^k they are defined as before, i.e., as in (3.112a). The descent direction based on the approximation of the dynamics based on GP can be therefore computed as shown in (3.123).

Next, we take advantage of the iterative nature of the optimization algorithm to design a concurrent learning phase. Given a (possibly suboptimal) state-input curve $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, it is applied to the real system (mathematically modeled by the unknown dynamics (3.109)) via (3.121) in an experimental session. This results in T novel measurements taken from the resulting state-input trajectory $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ as in (3.116). These measurements are then included in the new data-set $D^{k+1} = (Z^{k+1}, Y^{k+1})$. The proposed method described so far is summarized in Algorithm 11.

Algorithm 11 GP-Enhanced Gradient Method for Optimal Control

```
**Require:** trajectory $(\boldsymbol{x}^0, \boldsymbol{u}^0)$, data-set $(Z^0, Y^0)$ with associated posterior $m^0(\cdot)$ and $\Sigma^0(\cdot)$
  **for** $k = 0, 1, 2 \dots$ **do**
    set $\lambda_T^k = \nabla \ell_T(x_T^k)$
    **for** $t = T-1, \dots, 0$ **do**
      compute descent direction $v_t^k$ as
$$\begin{aligned} \lambda_t^k &= \left[(\hat{A}_t^k + m_x'^k(x_t^k, u_t^k)) - (\hat{B}_t^k + m_u'^k(x_t^k, u_t^k))K_t\right]^\top \lambda_{t+1}^k \\ &\qquad + a_t^k - K_t^\top b_t^k && (3.123a) \\ \Delta\mu_t^k &= -\left(\hat{B}_t^k + m_u'^k(x_t^k, u_t^k)\right)^\top \lambda_{t+1}^k - b_t^k && (3.123b) \\ \Delta\alpha_t^k &= K_t^\top \Delta\mu_t^k && (3.123c) \end{aligned}$$
    **for** $t = 0, \dots, T-1$ **do**
      compute the perturbed state-input curve
$$\begin{aligned} \alpha_t^{k+1} &= \alpha_t^k + \gamma^k \, \Delta\alpha_t^k \\ \mu_t^{k+1} &= \mu_t^k + \gamma^k \, \Delta\mu_t^k \end{aligned} \qquad (3.124)$$
      run the real system
$$\begin{aligned} u_t^{k+1} &= \mu_t^{k+1} + K_t(\alpha_t^{k+1} - x_t^{k+1}) \\ x_{t+1}^{k+1} &= f(x_t^{k+1}, u_t^{k+1}) \end{aligned} \qquad (3.125)$$
      collect a measurement
$$y_t^{k+1} = x_{t+1}^{k+1} - f_0(x_t^{k+1}, u_t^{k+1}) + \epsilon_t^{k+1}$$
    update data-set $(Z^{k+1}, Y^{k+1})$.
```

Remark 3.6. We implicitly assumed in Algorithm 11 that the state trajectory is initialized with $x_0^k = x_{\text{init}}$ for all k. △

Remark 3.7. The GP regression included in Algorithm 11 requires that at each iteration k the overall inference phase is repeated. One can reduce the computational complexity by employing, e.g., a dictionary-based strategy. △

Remark 3.8. In order to improve the learning performance and to reduce uncertainty one can devise various strategies based on the usual exploitation-exploration trade-off. For instance, data informativity can be enriched by injecting exploration noise $\epsilon_{\text{exp},t}^k$ in the perturbed input, i.e., one can set $u_t^{k+1} = u_t^{k+1} = \mu_t^{k+1} + K_t(\alpha_t^{k+1} - x_t^{k+1}) + \epsilon_{\text{exp},t}^k$ to reduce the variance of the posterior GP approximation. △

In the following, we analyze Algorithm 11. Inspired by the gradient-based Algorithm 15, tailored for fully known systems, also Algorithm 11 is based on a gradient method. Specifically, we show that the GP-enhanced version realizes a gradient descent method with error.

Let us rewrite the approximate dynamics (3.117) into an equivalent form as

$$\begin{aligned} x_{t+1} &= f_0(x_t, u_t) + m^k(x_t, u_t) \pm g_\star(x_t, u_t) \\ &= f(x_t, u_t) + \Delta^k(x_t, u_t) \end{aligned} \tag{3.126}$$

where

$$\Delta^k(x_t, u_t) := m^k(x_t, u_t) - g_\star(x_t, u_t).$$

Let $\Delta_{x,t}'^k$ and $\Delta_{u,t}'^k$ be the matrices obtained, respectively, by differentiating $\Delta^k(x, u)$ with respect to x and u about any state-input pair (x_t^k, u_t^k).

Assumption 3.7. *The time-varying matrices $\{\Delta_{x,t}'^k\}_{t\in[0,T-1]}$ and $\{\Delta_{u,t}'^k\}_{t\in[0,T-1]}$ are uniformly bounded in t and k.* △

Assumption 3.8. *The sets $\mathbb{A} \subset \mathbb{R}^{nT}$ and $\mathbb{M} \subset \mathbb{R}^{mT}$ are compact and the vectors $\boldsymbol{\alpha}^k \in \mathbb{A}$ and $\boldsymbol{\mu}^k \in \mathbb{M}$, for all $k \geq 0$.* △

Theorem 3.3. *Let Assumptions 3.6, 3.7 and 3.8 hold. Hence, any limit point $(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$ of the sequence $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}_{k\geq 0}$ generated by Algorithm 11 is such that the corresponding state-input trajectory $(\boldsymbol{x}^\star, \boldsymbol{u}^\star)$ calculated via* (3.125) *belongs to a neighborhood of a stationary point of problem* (3.111) *with high probability.* △

Proof. The proof relies on showing that the strategy implemented in Algorithm 11 is equivalent to a gradient method with error, which is then proved to be bounded. In light of (3.126), the adjoint system in (3.123a) can be rearranged equivalently as

$$\lambda_t^k = \underline{(A_t^k - B_t^k K_t)^\top \lambda_{t+1}^k + a_t^k - K_t^\top b_t} + (\Delta'^k_{x,t} - \Delta'^k_{u,t} K_t)^\top \lambda_{t+1}^k, \tag{3.127}$$

where the underlined quantity are the term associated with the true dynamics $f(\cdot)$.

We can write

$$\lambda_t^k = \bar{\lambda}_t^k + \Delta\lambda_t^k, \tag{3.128}$$

where $\bar{\lambda}_t^k$ is the term computed with by the full knowledge algorithm 7 and $\Delta\lambda_t^k$ evolves according to

$$\Delta\lambda_t^k = \left[(A_t^k + \Delta'^k_{x,t}) - (B_t^k + \Delta'^k_{u,t})K_t\right]^\top \Delta\lambda_{t+1}^k + (\Delta'^k_{x,t} - \Delta'^k_{u,t} K_t)\bar{\lambda}_{t+1}^k \tag{3.129}$$

with terminal condition $\Delta\lambda_T^k = 0$.

The descent direction in (3.123b) is an algebraic time-varying map depending on λ_{t+1}^k and b_t^k and can be expressed as

$$\begin{aligned}\Delta\mu_t^k &\overset{(a)}{=} -B_t^{k\top}\bar{\lambda}_{t+1}^k - b_t^k - B_t^{k\top}\Delta\lambda_{t+1}^k - \Delta'^{k\top}_{u,t}\lambda_{t+1}^k \\ &\overset{(b)}{=} \Delta\bar{\mu}_t^k + \Delta\tilde{\mu}_t^k\end{aligned}$$

where in (a) we have used (3.128), and in (b) the term $\Delta\bar{\mu}_t^k$ is the same computed by the full-knowledge Algorithm 3.12b and we have defined

$$\Delta\tilde{\mu}_t^k := -(B_t^k - \Delta'^k_{u,t})^\top \Delta\lambda_{t+1}^k - \Delta'^{k\top}_{u,t}\bar{\lambda}_{t+1}^k. \tag{3.130}$$

Similarly, we can express (3.123c) as:

$$\begin{aligned}\Delta\alpha_t^k &= K_t\Delta\bar{\mu}_t^k + K_t\Delta\tilde{\mu}_t^k \\ &\overset{(c)}{=} \Delta\bar{\alpha}_t^k + \Delta\tilde{\alpha}_t^k\end{aligned}$$

where in (c) the term $\Delta\bar{\alpha}_t^k$ is the same computed by the full-knowledge Algorithm 3.12c and we have defined $\Delta\tilde{\alpha}_t^k := K_t\Delta\tilde{\mu}_t^k$.

Therefore, the components of the updated curve $(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})$ are

$$\alpha_t^{k+1} = \underline{\alpha_t^k + \gamma^k\,\Delta\bar{\alpha}_t^k} + \gamma^k\,\Delta\tilde{\alpha}_t^k, \qquad \mu_t^{k+1} = \underline{\mu_t^k + \gamma^k\,\Delta\bar{\mu}_t^k} + \gamma^k\,\Delta\tilde{\mu}_t^k \tag{3.131}$$

where $(\Delta\bar{\alpha}_t^k, \Delta\bar{\mu}_t^k)$ are the t-th component of the negative gradient of the reduced cost function $J(\boldsymbol{\alpha}, \boldsymbol{\mu})$ associated with problem (3.111) using the true, though unknown, dynamics.

By forward simulation of the closed-loop dynamics (3.121) we can compute $\boldsymbol{x}^k := \phi(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ and $\boldsymbol{u}^k := \psi(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, for all k, with $\phi(\cdot,\cdot)$ and $\psi(\cdot,\cdot)$ being the continuous shooting maps described in (3.8). Since f is smooth, it holds that $\boldsymbol{x}^k \in \mathbb{X}$ and $\boldsymbol{u}^k \in \mathbb{U}$ for all k, where $\mathbb{X}$ and $\mathbb{U}$ are compact set defined as

$$\begin{aligned}\mathbb{X} &:= \{\boldsymbol{x} \in \mathbb{R}^{nT} \mid \boldsymbol{x} = \phi(\boldsymbol{\alpha}, \boldsymbol{\mu}),\ \ \forall (\boldsymbol{\alpha}, \boldsymbol{\mu}) \in \mathbb{A} \times \mathbb{M}\},\\ \mathbb{U} &:= \{\boldsymbol{u} \in \mathbb{R}^{mT} \mid \boldsymbol{u} = \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}),\ \ \forall (\boldsymbol{\alpha}, \boldsymbol{\mu}) \in \mathbb{A} \times \mathbb{M}\}\end{aligned}$$

Consider now the linearization of the dynamics as in (3.112b). From $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k) \in \mathbb{A} \times \mathbb{M}$, we have that $A_t^k = \nabla_1 f(\phi_t(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k), \psi_t(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k))^\top$ is uniformly bounded in t and in k. That is, for all t and k it holds

$$\|A_t^k\| \leq A_0$$

for some $A_0 > 0$. Similarly, for all t and k we have

$$\|B_t^k\| \leq B_0$$

for some $B_0 > 0$. By exploiting the linearity of (3.12a) and defining $\bar{\boldsymbol{\lambda}}^k := (\bar{\lambda}_1^k, \ldots, \bar{\lambda}_T^k)$, we can write

$$\bar{\boldsymbol{\lambda}}^k = \hat{\Phi}_1^k \bar{\lambda}_T^k + \mathcal{R}_1^k \mathbf{a}^k + \mathcal{K}_1^k \mathbf{b}^k$$

for suitably defined matrices $\hat{\Phi}_1^k$ (collecting the state transition matrices for each t, made by bounded state matrices), $\hat{R}_1^k$ and $\mathcal{K}_1^k$ (involving the convolution between the state and the input matrices, both bounded) where $\mathbf{a}^k := (a_0^k, \ldots, a_{T-1}^k)$ and $\mathbf{b}^k := (-b_0^k, \ldots, -b_{T-1}^k)$. Since $\bar{\lambda}_T^k$, $\mathbf{a}^k$ and $\mathbf{b}^k$ are bounded, we can write

$$\|\bar{\boldsymbol{\lambda}}^k\| \leq c_1$$

for all k and for some $c_1 > 0$. Then, using similar arguments for the linear system (3.129), we introduce $\Delta\boldsymbol{\lambda}^k := (\Delta\lambda_1^k, \ldots, \Delta\lambda_T^k)$ and write

$$\Delta\boldsymbol{\lambda}^k = \hat{\Phi}_2^k \underbrace{\Delta\lambda_T^k}_{=0} + \hat{R}_2^k \bar{\boldsymbol{\lambda}}^k + \mathcal{K}_2^k \mathbf{b}^k$$

for suitable matrices $\hat{\Phi}_2^k$ and $\hat{R}_2^k$, which are defined similarly to $\hat{\Phi}_1^k$ and $\hat{R}_1^k$. Leveraging on Lemma 3.4, the norm of $\Delta\boldsymbol{\lambda}^k$ can be bounded with high probability as

$$\|\Delta\boldsymbol{\lambda}^k\| \leq c_2 \|\bar{\boldsymbol{\lambda}}^k\| \leq c_2 c_1$$

for all k and for some $c_2 > 0$. Finally, leveraging on their definition and stacking all the components $\Delta\tilde{\alpha}_t^k$ and $\Delta\tilde{\mu}_t^k$ in two vectors $\Delta\tilde{\boldsymbol{\alpha}}^k$ and $\Delta\tilde{\boldsymbol{\mu}}^k$, we can compactly write

$$\Delta\tilde{\boldsymbol{\mu}}^k = \hat{C}_1^k \Delta\boldsymbol{\lambda}^k + \hat{D}^k \bar{\boldsymbol{\lambda}}^k = (\hat{C}_1^k R_2^k + \hat{D}^k)\bar{\boldsymbol{\lambda}}^k$$

and

$$\Delta\tilde{\boldsymbol{\alpha}}^k = \hat{C}_2^k \Delta\tilde{\boldsymbol{\mu}}^k = \hat{C}_2^k(\hat{C}_1^k R_2^k + \hat{D}^k)\bar{\boldsymbol{\lambda}}^k$$

for suitably defined matrices $\hat{C}_1^k$, $\hat{C}_2^k$, and $\hat{D}^k$. Taking the norms we can write

$$\|\Delta\tilde{\boldsymbol{\mu}}^k\| \leq \|\hat{C}_1^k R_2^k + \hat{D}^k\|\|\bar{\boldsymbol{\lambda}}^k\| \leq c_3c_2c_1 \tag{3.132a}$$

and

$$\|\Delta\tilde{\boldsymbol{\alpha}}^k\| \leq \|\hat{C}_2^k\hat{C}_1^k R_2^k + \hat{D}^k\|\|\bar{\boldsymbol{\lambda}}^k\| \leq c_4c_3c_2c_1 \tag{3.132b}$$

for all k and for some $c_3 > 0$ and $c_4 > 0$.

Then, Algorithm 11 generates a sequence of state-input trajectories $\{\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k\}_{k\geq 0}$ that can be associated to a gradient method with error given by (3.131). Being the error uniformly bounded by (3.132), we can invoke convergence results for a gradient method with error (see, e.g., [28, Chap. 1]) to conclude the proof. ■

It is worth noting that Assumption 3.7 is reasonable since one expects $m^k(\cdot)$ get close to $g(\cdot)$ when data are informative and Assumption 3.6 is satisfied. Assuming that $(x, u) \in \mathbb{X} \times \mathbb{U}$, with $\mathbb{X}$ and $\mathbb{U}$ compact sets, Lemma 3.4 gives a uniform quantitative bound on the distance of the posterior mean from $g(\cdot)$. Moreover, if also $\nabla g(\cdot)$ lives in a RKHS then a uniform bound on the derivatives can be established as well.

Theorem 3.3 states that the scheme generates a sequence of curves $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ such that the corresponding state-input trajectories $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ converge to a neighborhood of a stationary point. However, if, as just discussed, $m(\cdot)$ approaches the unknown value of $g_\star(\cdot)$, then in the limit $\Delta\boldsymbol{\lambda}^k$ vanishes, thus giving a vanishing error in the gradient scheme.

3.4.3 Numerical Simulations

We consider a canonical testbed for nonlinear control given by the inverted pendulum, shown in Figure 3.14, which is governed by the following nonlinear continuous-time dynamics

$$Ml^2\ddot{\theta} = +Mgl\sin(\theta) - \mathrm{f}_\ell l\dot{\theta} - \mathrm{f}_c l\dot{\theta}^3 + u, \tag{3.133}$$

where θ is the angle, $\dot{\theta}$ is the velocity and u is the input with the same sign of θ. Moreover, M is the mass, l is the pendulum length, g is the gravity acceleration, $\mathrm{f}_\ell = \mathrm{f}_{\ell,0} + \Delta\mathrm{f}_\ell$ is the linear friction coefficient and $\mathrm{f}_c = \mathrm{f}_{c,0} + \Delta\mathrm{f}_c$ is the cubic friction coefficient.

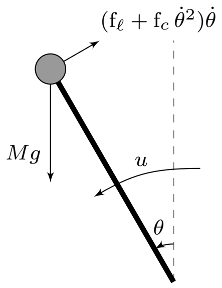

$(\mathrm{f}_\ell + \mathrm{f}_c\,\dot{\theta}^2)\dot{\theta}$
Mg
u
θ

Figure 3.14: Scheme of the pendulum.

We suppose to partially know only the linear coefficient, while the cubic term is totally unmodeled, i.e., $\mathrm{f}_c = \Delta\mathrm{f}_c$. It is a realistic framework since unmodeled friction-like effects are usually present in more complex applications as, e.g., drag forces in quadrotors or tyre frictions in cars.

Let us consider a discrete-time state-space representation of (3.133), obtained for simplicity via forward Euler, given by

$$\begin{bmatrix} x_{1,t+1} \\ x_{2,t+1} \end{bmatrix} = \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} + \delta \begin{bmatrix} x_{2,t} \\ a\sin(x_{1,t}) + bx_{2,t} + cx_{2,t}^3 \end{bmatrix} + \delta \begin{bmatrix} 0 \\ d \end{bmatrix} u_t$$

where x_1 corresponds to θ, x_2 corresponds to $\dot{\theta}$, δ is the sampling period, $a = \frac{g}{l}$, $b = -\frac{\mathrm{f}_\ell}{Ml}$, $c = \frac{\mathrm{f}_c}{Ml}$ and $d = \frac{1}{Ml^2}$.

By making explicit the uncertain terms, we can obtain a system in the form (3.109), i.e., as the sum of a known and unknown term given respectively by

$$f_0(x_t, u_t) = \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} + \delta \begin{bmatrix} x_{2,t} \\ a_0\sin(x_{1,t}) + b_0 x_{2,t} \end{bmatrix} + \delta \begin{bmatrix} 0 \\ d_0 \end{bmatrix} u_t$$

with $a_0 = \frac{g}{l}$, $b_0 = -\frac{\mathrm{f}_{\ell,0}}{M_0 l}$, $d_0 = \frac{1}{Ml^2}$, and

$$g_\star(x_t, u_t) = \delta \begin{bmatrix} x_{2,t} \\ \Delta b x_{2,t} + \Delta c x_{2,t}^3 \end{bmatrix}$$

with $\Delta b = -\frac{\Delta\mathrm{f}}{M_0 l}$, $\Delta c = -\frac{\Delta\mathrm{f}_c}{M_0 l}$. We consider as nominal parameters $l = 1$ m, $M_0 = 1$ Kg, $\mathrm{f}_{\ell,0} = 0.5$ Nm$\frac{\mathrm{s}}{\mathrm{rad}}$ and $\mathrm{f}_{c,0} = 0$ Nm$\frac{\mathrm{s}}{\mathrm{rad}}$. The uncertain terms are $\Delta M = 1$ Kg, $\Delta\mathrm{f}_\ell = 37.5$ Nm$\frac{\mathrm{s}}{\mathrm{rad}}$ and $\Delta\mathrm{f}_c = 30$ Nm$\frac{\mathrm{s}}{\mathrm{rad}}$.

We consider a tracking problem, which translates in the following quadratic cost function

$$\sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) = \sum_{t=0}^{T-1} \left(\left\|x_t - x_{\text{ref},t}\right\|_Q^2 + \left\|u_t - u_{\text{ref},t}\right\|_R^2 \right) + \left\|x_T - x_{\text{ref},T}\right\|_{Q_f}^2$$

where $Q \in \mathbb{R}^{2\times 2}$, $Q_f \in \mathbb{R}^{2\times 2}$ and $R \in \mathbb{R}$ are

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Q_f = \begin{bmatrix} 10 & 0 \\ 0 & 10^2 \end{bmatrix}, \qquad R = 10^{-3}.$$

We set the sampling period to $\delta = 10^{-3}s^{-1}$ so that the horizon length $T = 20s/\delta = 20{\cdot}10^3$. As reference trajectory we used a step signal between two equilibrium configurations, namely from $(x_{\text{ref, i}}, u_{\text{ref, i}}) = (0, 0, 0)$ to $(x_{\text{ref, f}}, u_{\text{ref, f}}) = (0.5\text{rad}, 0, -M_0 gl\sin(0.5\text{rad}))$.

The step-size is diminishing with $\beta^k = 0.1/k^{0.009}$. The parameters of the squared exponential kernel (3.113) are $\sigma_\varphi^2 = 1$ and $L_\varphi = \text{diag}(10^{-4}, 10, 10^{-4})$.

The cost error is represented in Figure 3.15 and shows the difference between $J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, the cost evaluated at the k-th iteration of the GP-enhanced algorithm, and $J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, the optimal cost computed with a full knowledge of $f(\cdot)$. The cost error diminishes across iterations as the optimization proceeds with a sublinear rate, as customary in gradient methods with diminishing step-size.

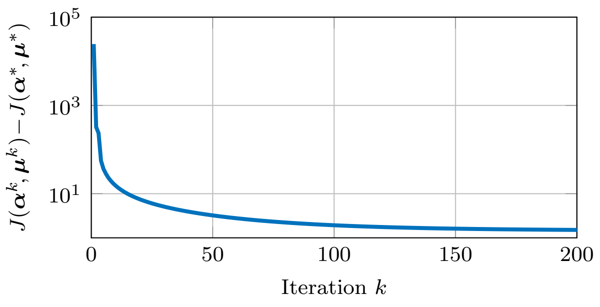


Figure 3.15: Evolution of the cost error across iterations.

In Figure 3.16 a comparison between the state-input trajectories of the two algorithms is proposed. It can be appreciated that the discrepancies between the real system and its nominal model are well captured by the GP regression. Indeed the red and the blue trajectories overlap showing the effectiveness of the proposed approach.

3.5 Optimal Control of Nonlinear Systems with Stochastic Uncertainties

In this section, we specialize the GoPRONTO framework to the stochastic optimal control scenario. The considered discrete-time nonlinear dynamics subjected to stochastic

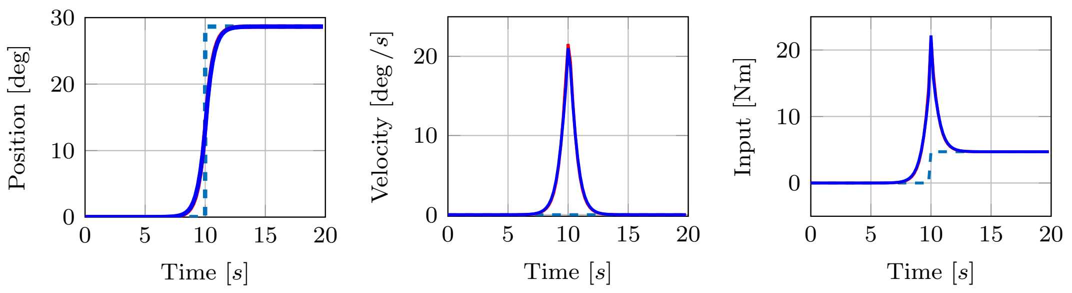


Figure 3.16: Comparison among the reference curve (dashed blue) and the results of the algorithms based on full-knowledge (red) and on the Gaussian process regression (blue).

disturbance reads as

$$x_{t+1} = f(x_t, u_t, w_t), \qquad x_0 = x_{\text{init}}, w_t \sim p_{\mathrm{w}_t} \tag{3.134}$$

in which $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$ and $x_t \in \mathbb{R}^n$ represent the state of the system at time t, $u_t \in \mathbb{R}^m$ the control input and $w_t \in \mathbb{R}^p$ is stochastic disturbance, realization of a random variable $\mathrm{w}_t(\omega)$ with probability density function p_{w_t}. Considering a time-horizon of length T, we define the sequence of T random variable w_t as

$$\mathbf{w}(\omega) := \operatorname{col}(\mathrm{w}_0(\omega), \dots, \mathrm{w}_{T-1}(\omega)) \tag{3.135}$$

which is a random variable itself, i.e., $\mathbf{w} : \Omega \to \mathbb{R}^{pT}$. Notice that we denote as $\boldsymbol{w} \in \mathbb{R}^{pT}$ a realization of $\mathbf{w}(\omega)$. For the sake of compactness, we omit the dependence of the random variables on the samples ω. The expected value taken with respect to $\mathbf{w}$ of a function of random variable $h(\mathbf{w})$, $h : \mathbb{R}^{pT} \to \mathbb{R}$ is defined as

$$\mathbb{E}_{\mathbf{w}}\left[h(\mathbf{w})\right] := \int_{\mathbb{R}^{pT}} h(w) p_{\mathbf{w}}(w) dw \tag{3.136}$$

where $p_{\mathbf{w}}$ represents the probability density function associated to $\mathbf{w}$[4].

To each trajectory $(\boldsymbol{x}, \boldsymbol{u})$, we associate a performance metrics by means of the nonlinear cost function

$$\ell(\boldsymbol{x}, \boldsymbol{u}) := \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{3.137}$$

where $\ell_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the so-called stage cost and $\ell_T : \mathbb{R}^n \to \mathbb{R}$ represents the

[4]It holds that

$$\mathbb{E}_{\mathbf{w}}\left[h(\mathbf{w})\right] := \int_{\Omega} h(\boldsymbol{w}(\omega))\, d\mathbb{P}(\omega) = \int_{\mathbb{R}^p} h(w)\, p_{\mathbf{w}}(w)\, dw.$$

terminal cost.

Assuming a deterministic sequence of inputs $\boldsymbol{u} \in \mathbb{R}^{mT}$, for a given initial condition $x_0 = x_{\text{init}}$ and a specified time horizon T, one can see the resulting state trajectory $\boldsymbol{x}$ as a function of the random variable $\mathbf{w}$. This comes from the composition of each random variable w_t, for $t = 0, \dots, T-1$, through the nonlinear dynamics (3.134). Hence, we have resort to the minimization of the performance index (3.137) taken in expectation with respect to the random variable $\mathbf{w}$.

Remark 3.9. Notice that, we can interpret the state sequence $\boldsymbol{x}$ as the realization of a random variable $\mathbf{x} : \Omega \to \mathbb{R}^{nT}$ characterized by composition of each random variable w_t, $t = 0, \dots, T-1$ via the nonlinear dynamics (3.134). $\triangle$

Overall, our goal is to choose the input sequence $\boldsymbol{u}$ such that the performance index (3.137) is minimized. Due to the presence of the stochastic disturbance w_t in the system dynamics, each system trajectory represents a realization of a function of the random variable $\mathbf{w}$. Hence, we resort to the minimization of the performance index taken in expectation with respect to the random variable $\mathbf{w}$. Overall, we aim at solving the following stochastic optimal control problem

$$\min_{\boldsymbol{x},\boldsymbol{u}} \ \mathbb{E}_{\mathbf{w}}\Bigg[\sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T)\Bigg] \tag{3.138a}$$

$$\text{subj.to} \ \ x_{t+1} = f(x_t, u_t, w_t) \tag{3.138b}$$

$$x_0 = x_{\text{init}} \tag{3.138c}$$

$$w_t \sim p_{\mathrm{w}_t} \tag{3.138d}$$

where $x_{\text{init}} \in \mathbb{R}^n$ is the (given) initial condition. Notice that, although the random variable $\mathbf{w}$ and its realizations do not appear in the performance index explicitly, in (3.138a) we take the expected value with respect to $\mathbf{w}$ due to its effect via the nonlinear dynamics (3.134). Our theoretical analysis is based on some smoothness assumptions which are standard in nonconvex stochastic optimization. Before introducing said assumptions, let us give an useful definition.

Definition 3.4 (Lipschitz function). *A function $h : \mathbb{R}^n \to \mathbb{R}$ is said to be uniformly Lipschitz continuous with Lipschitz constant $\bar{L}$ (or uniformly L-Lipschitz continuous) if there exists a constant $0 < L < \infty$ such that*

$$\|h(x) - h(x')\| \leq L\|x - x'\| \tag{3.139}$$

for all $x, x' \in \mathbb{R}^n$. The Lipschitz constant $\bar{L}$ is the smallest L such that (3.139) *holds. A function $g : \mathbb{R}^n \to \mathbb{R}$ is said to have L'-Lipschitz continuous gradient if its gradient $\nabla g : \mathbb{R}^n \to \mathbb{R}^n$ is uniformly L'-Lipschitz continuous.* $\triangle$

Let us now take the following assumptions.

Assumption 3.9 (Dynamics). *Function $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^n$ is twice continuously differentiable, i.e., is of class $\mathcal{C}^2$. Moreover, f is L_f-Lipschitz continuous and with $L_{f'}$-Lipschitz continuous gradient.* △

Assumption 3.10 (Cost Functions). *The cost function $\ell_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is twice continuously differentiable, i.e., is of class $\mathcal{C}^2$, and with $L_{\ell_t'}$-Lipschitz continuous gradient for all $t = 0, \dots, T-1$. Similarly, function $\ell_T : \mathbb{R}^n \to \mathbb{R}$ is is twice continuously differentiable, i.e., is of class $\mathcal{C}^2$, and with $L_{\ell_T'}$-Lipschitz continuous gradient.* △

3.5.1 Stochastic Gradient Descent-based GoPRONTO

We now extend the strategy proposed in Section 3.2 by introducing a stochastic version of the nonlinear tracking feedback policy represented by the closed-loop dynamics in (3.6). Specifically, given a stabilizing gain K_t and a certain $\mu_t \in \mathbb{R}^m$ and $\alpha_t \in \mathbb{R}^n$, we can define a feedback-based nonlinear controller as

$$u_t = \mu_t + K_t(\alpha_t - x_t) \tag{3.140a}$$

$$x_{t+1} = f(x_t, u_t, w_t) \tag{3.140b}$$

Therefore, for a given curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$, the feedback system (3.140) implements a stochastic projection operator $\mathcal{P} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^p \to \mathcal{T}_{x_{\text{init}}}$ which maps state-input curves into trajectories of the nonlinear system. This *stochastic* projection operator can be compactly represented as

$$\begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\mu} \end{bmatrix} \longmapsto \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} := \mathcal{P}(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w}) = \begin{bmatrix} \phi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w}) \\ \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w}) \end{bmatrix}, \tag{3.141}$$

where $\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})$ and $\psi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})$ are the state and input components of $\mathcal{P}(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})$. Notice that, for a deterministic curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$, the resulting $(\mathbf{x}, \mathbf{u})$.state-input trajectory $(\mathbf{x}, \mathbf{u})$ is a function of the random variable $\mathbf{w}$. The embedding feedback reformulation of the original stochastic optimal control problem (3.138), reads as

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\mu}} \; \mathbb{E}_{\mathbf{w}}\left[\sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T)\right] \tag{3.142a}$$

$$\text{subj.to} \;\; x_{t+1} = f(x_t, u_t, w_t), \tag{3.142b}$$

$$u_t = \mu_t + K_t(\alpha_t - x_t) \quad t = 0, \dots, T-1 \tag{3.142c}$$

$$x_0 = x_{\text{init}} \tag{3.142d}$$

$$w_t \sim p_{\mathrm{w}_t} \tag{3.142e}$$

Remark 3.10. We point out that x_t and u_t must be regarded as realizations of random processes, since they depend on the specific realization of w. Conversely, the state-input curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ is a deterministic state-input sequence, i.e., a non-random parameter of the optimal control problem.

Indeed, to each *deterministic* curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ we are able to associate, for each time-instant t, a pair of random variables x_t and u_t defined by the composition of the closed-loop dynamics (3.140) and $\boldsymbol{w}$.

Stochastic Optimization Reduced Problem Formulation

For a given initial condition $x_0 = x_{\text{init}}$ and curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ and a random variable realization $\boldsymbol{w}$, the stochastic counterparts of the maps (3.8) read

$$x_t = \phi_t(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w}) \tag{3.143a}$$

$$u_t = \psi_t(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w}) \tag{3.143b}$$

where $\phi_t : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{pT} \to \mathbb{R}^n$ and $\psi_t : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{pT} \to \mathbb{R}^m$. Again, these functions can be seen as functions of the random variable w. With (3.143) at hand, we can rewrite the performance index (3.137) as

$$\begin{aligned}\ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}; \boldsymbol{w}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w})) &= \sum_{t=0}^{T-1} \ell_t(\phi_t(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w}), \psi_t(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w})) + \ell_T(\phi_T(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w})) \\ &= J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w}) \end{aligned} \tag{3.144a}$$

where $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w})$ is the so-called reduced cost. Notice that, due to its dependence on a specific realization $\boldsymbol{w}$ of w, we can write $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathrm{w})$ which is a function of random variable. Now, we are in the position to reformulate the original optimal control problem (3.142) in its reduced form, i.e.,

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\mu}} \ \mathbb{E}_{\mathrm{w}}\Big[J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathrm{w})\Big] = \min_{\boldsymbol{\alpha}, \boldsymbol{\mu}} \ \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu}), \tag{3.145}$$

Problem (3.145) is an unconstrained stochastic optimization problem, which falls in the setting detailed in Section A.3. Notice that, the expected value in (3.145), according to its definition in (3.136) is equivalent to

$$\mathbb{E}_{\mathrm{w}}\Big[J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathrm{w})\Big] = \int_{\mathbb{R}^{pT}} J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w}) p_{\mathrm{w}}(\boldsymbol{w})\, d\boldsymbol{w} = \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu}) \tag{3.146}$$

where $p_{\mathrm{w}}(\cdot)$ is the probability density function coming from the composition of each random variable w_t, $t = 0, \ldots, T-1$ via the nonlinear dynamics (3.134) and, in general, it is hard to characterize. To overcome these issues, in this work we resort to a stochastic gradient descent method (introduced in Section A.3).

Stochastic GoPRONTO Algorithm

In the following, we detail the unconstrained stochastic optimization algorithm used to solve problem (3.145). Our algorithm implements a stochastic gradient descent method (see Section A.3), where the solution to problem (3.145) is iteratively improved from an initial guess $(\boldsymbol{\alpha}^0, \boldsymbol{\mu}^0)$ following, for all $k \geq 0$, the update rule

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k - \gamma^k G_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^k) \tag{3.147}$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k - \gamma^k G_{\boldsymbol{\mu}}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^k) \tag{3.148}$$

where $G_\alpha(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^k) \in \mathbb{R}^{nT}$ and $G_\mu(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^k)$ represent the estimators for the gradient of the reduced cost $\bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$ with respect to its components $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$ respectively. These estimators are computed in correspondence the noise realization $\boldsymbol{w}^k$. Here, assume we have access to an oracle which gives us a realization $\boldsymbol{w}^k$ of $\mathbf{w}$.

Meta-algorithm: Gradient estimator We start by providing the meta-algorithm used for the gradient estimator $G(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})$ of $\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$, needed to solve problem (3.145) via SGD (cf. Appendix A.3). The procedure is detailed in Algorithm 12, where, we consider a realization $\boldsymbol{w}$ of the random variable $\mathbf{w}$ and we adopt the following notation

$$a_t := \nabla_1 \ell_t(x_t, u_t), \quad A_t(\boldsymbol{w}) := \nabla_1 f(x_t, u_t, w_t)^\top, \tag{3.149a}$$

$$b_t := \nabla_2 \ell_t(x_t, u_t), \quad B_t(\boldsymbol{w}) := \nabla_2 f(x_t, u_t, w_t)^\top. \tag{3.149b}$$

Algorithm 12 Gradient estimator for $\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$

```
Require: State-input curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ and realization $\boldsymbol{w}$
  Compute trajectory $(\boldsymbol{x}, \boldsymbol{u})$ projecting the curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ via forward integration of
  for $t = 0, \ldots, T-1$ do
      $$\begin{aligned} u_t &= \mu_t + K_t(\alpha_t - x_t) \\ x_{t+1} &= f(x_t, u_t, w_t) \end{aligned} \qquad (3.150)$$
  Set $\lambda_T = \nabla \ell_T(x_T)$
  for $t = T-1, \ldots, 0$ do
      Compute $A_t(\boldsymbol{w})$, $B_t(\boldsymbol{w})$, $a_t(\boldsymbol{w})$ and $b_t(\boldsymbol{w})$.
      Compute estimator via
      $$\lambda_t = (A_t(\boldsymbol{w}) - B_t(\boldsymbol{w})K_t)^\top \lambda_{t+1} + a_t(\boldsymbol{w}) - K_t^\top b_t(\boldsymbol{w}) \qquad (3.151a)$$
      $$\Delta\mu_t = -B_t(\boldsymbol{w})^\top \lambda_{t+1} - b_t(\boldsymbol{w}) \qquad (3.151b)$$
      $$\Delta\alpha_t = K_t^\top \Delta\mu_t \qquad (3.151c)$$
  return $G(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w}) = \begin{bmatrix} \boldsymbol{\Delta\alpha} \\ \boldsymbol{\Delta\mu} \end{bmatrix}$
```

SGD GoPRONTO With the Gradient Estimator procedure at hand, we are now in the position to solve problem (3.145) by means of a SGD scheme. The proposed procedure in summarized in Algorithm 13.

Algorithm 13 Stochastic GoPRONTO

```
  Start with an initial trajectory $(\boldsymbol{x}^0, \boldsymbol{u}^0)$ and disturbance realization $\boldsymbol{w}^0$.
  for $k = 0, 1, 2 \ldots$ do
      Given a state-input curve $(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ and disturbance realization $\boldsymbol{w}^k$ compute $G(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^k)$ via Algorithm 12.
      for $t = 0, \ldots, T-1$ do
          Update the deterministic (unfeasible) curve
          $$\begin{aligned} \alpha_t^{k+1} &= \alpha_t^k + \gamma^k G_{\boldsymbol{\alpha}_t}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^k) \\ \mu_t^{k+1} &= \mu_t^k + \gamma^k G_{\boldsymbol{\mu}_t}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^k) \end{aligned} \qquad (3.152)$$
```

Before giving the convergence result, let us introduce some preliminary assumptions.

Assumption 3.11. *The dynamics f is L_f-Lipschitz continuous. Moreover, the functions f, ℓ_t and ℓ_T have, respectively, $L_{f'}$-, $L_{\ell_t'}$-,$L_{\ell_T'}$-Lipschitz continuous gradients for all $t \geq 0$.*

Assumption 3.12. *The gradient of ℓ_t, ℓ_T is uniformly bounded for all $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$.*

Assumption 3.13. *The sets $\mathbb{A} \subset \mathbb{R}^{nT}$ and $\mathbb{M} \subset \mathbb{R}^{mT}$ are compact and the vectors $\boldsymbol{\alpha}^k \in \mathbb{A}$ and $\boldsymbol{\mu}^k \in \mathbb{M}$, for all $k \geq 0$.* △

Finally, we make the following assumptions about the stepsize.

Assumption 3.14. *Assume the stepsize sequence $\{\gamma^k\}_{k>0}$ in Algorithm 13 be chosen such that*

$$\sum_{k=0}^{\infty} \gamma^k = \infty \qquad \sum_{k=0}^{\infty} (\gamma^k)^2 \leq \infty \tag{3.153}$$

e.g., a diminishing stepsize. △

We are now able to state the following result.

Theorem 3.4. *Let Assumptions 3.9, 3.10, 3.11, 3.13, and 3.12 hold and consider as descent direction the output of the gradient estimator generated by Algorithm 12. Assume also that the random variable* $\mathbf{w}$ *is bounded. Then, Algorithm 13 implements a Stochastic Gradient Descent method and inherits its convergence results. Specifically, if the stepsize is chosen accordingly to Assumption 3.14, it holds true, with $\Gamma^N := \sum_{k=1}^N \gamma^k$*

$$\lim_{N\to\infty} \mathbb{E}\left[\frac{1}{\Gamma^N} \sum_{k=1}^{N} \gamma^k \|\nabla \bar{J}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)\|\right] = 0. \qquad △$$

The proof of Theorem 3.4 is given in the following Section.

3.5.2 Algorithm Analysis

Before giving a formal proof of Theorem 3.4 let us introduce some preparatory lemmas.

We start by showing that the gradient estimator implemented via Algorithm 12 represents an unbiased estimator of the true gradient of $\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$.

Proposition 3.1. *The gradient estimator $G(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w})$ obtained via Algorithm 12 is an unbiased estimator of $\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$.* △

Proof. The proof goes through two main steps. Firstly, we show that Algorithm 12 performs an exact computation of the gradient of the random variable $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})$ associated to a given realization of the noise $\boldsymbol{w}$. Secondly, we complete the proof showing that if gradient estimator computes exactly $\nabla J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})$ then it is an unbiased estimator of $\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$. Assume to have access to a realization $\bar{\boldsymbol{w}}$ of the disturbance $\mathbf{w}$, i.e., to gather the sequence $\{\bar{w}_t\}_{t=0,\dots,T-1}$, via e.g., an oracle. We can see that the quantity $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}})$ is a deterministic quantity representing the reduced cost associated to a *deterministic* optimal control problem parametrized in $\bar{\boldsymbol{w}}$. Namely, for a given $\bar{\boldsymbol{w}}$, the original problem associated with $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}})$ reads as

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\alpha},\boldsymbol{\mu}} \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{3.154a}$$

$$\text{subj.to } x_{t+1} = f(x_t, u_t, \bar{w}_t) \tag{3.154b}$$

$$u_t = \mu_t + K_t(\alpha_t - x_t) \qquad t = 0, \dots, T-1 \tag{3.154c}$$

$$x_0 = x_{\text{init}}. \tag{3.154d}$$

Along the lines of Section 3.2, we manipulate (3.154) to compute the gradient of $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}})$ with respect to its components $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$. The nonlinear dynamics in (3.154b) and the closed-loop structure on the control input (3.154c), can be written as an equality constraint $h : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{pT} \to \mathbb{R}^{nT}$ defined as

$$h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}) := \begin{bmatrix} f(x_0, u_0, \bar{w}_0) - x_1 \\ \vdots \\ f(x_{T-1}, u_{T-1}, \bar{w}_{T-1}) - x_T \\ \mu_0 + K_t(\alpha_0 - x_0) - u_0 \\ \vdots \\ \mu_{T-1} + K_t(\alpha_{T-1} - x_{T-1}) - u_{T-1} \end{bmatrix}. \tag{3.155}$$

We can now introduce an auxiliary function say $\mathcal{L} : \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{nT} \times \mathbb{R}^{mT} \times \mathbb{R}^{pT} \times \mathbb{R}^{nT+m} \to \mathbb{R}$, defined as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}, \boldsymbol{\lambda}) := \ell(\boldsymbol{x}, \boldsymbol{u}) + h(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}, \bar{\boldsymbol{w}})^\top \boldsymbol{\lambda} \tag{3.156}$$

where the (multiplier) vector $\boldsymbol{\lambda} \in \mathbb{R}^{nT+mT}$ is arranged as

$$\boldsymbol{\lambda} := \text{col}(\lambda_1, \dots, \lambda_T, \tilde{\lambda}_1, \dots, \tilde{\lambda}_T)$$

with each $\lambda_t \in \mathbb{R}^n$ and $\tilde{\lambda}_t \in \mathbb{R}^m$. By defining $\phi(\cdot)$ and $\psi(\cdot)$ as the vertical stack of the maps $\phi_t(\cdot)$ and $\psi_t(\cdot)$, we can see that, by construction, for all $(\boldsymbol{\alpha}, \boldsymbol{\mu}) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ it holds

$$\tilde{h}(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}), \boldsymbol{\alpha}, \boldsymbol{\mu}) = 0. \tag{3.157}$$

Since $J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}) \equiv \ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu}, \bar{\boldsymbol{w}}))$ (Cf. (3.137)), the auxiliary function (3.156) enjoys the following property

$$\mathcal{L}(\phi(\boldsymbol{\alpha},\boldsymbol{\mu},\bar{\boldsymbol{w}}),\psi(\boldsymbol{\alpha},\boldsymbol{\mu},\bar{\boldsymbol{w}}),\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{\lambda}) = J(\boldsymbol{\alpha},\boldsymbol{\mu}) \tag{3.158}$$

for all $(\boldsymbol{\alpha},\boldsymbol{\mu})$, $\bar{\boldsymbol{w}}$, and *for all* $\boldsymbol{\lambda} \in \mathbb{R}^{nT+mT}$. Therefore, in this formulation we can think about $\boldsymbol{\lambda}$ as a parameter or a degree of freedom. Building on (3.158), one can simplify the computation of the gradient of the reduced cost function $J(\bar{\boldsymbol{\alpha}},\bar{\boldsymbol{\mu}},\bar{\boldsymbol{w}})$ with a proper choice $\boldsymbol{\lambda}$. Such a choice will eventually give rise to (3.151). We refer the interested reader to [212] for a detailed derivation. Since Algorithm 12 computes exactly the gradient of a realization of $J(\boldsymbol{\alpha},\boldsymbol{\mu},\mathbf{w})$ we can prove, by Leibniz integral rule that this is an unbiased estimator of the gradient $\bar{J}(\boldsymbol{\alpha},\boldsymbol{\mu})$. First of all, notice that the random variable $\mathbf{w}$ can be seen as the seed for generating the output of Algorithm 12, i.e., $G(\boldsymbol{\alpha},\boldsymbol{\mu},\mathbf{w})$ is a function of the random variable $\mathbf{w}$. Since $G(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}) = \nabla J(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w})$ for all $\boldsymbol{w}$, we can take its expectation with respect to $\mathbf{w}$ and observe that

$$\begin{aligned}
\mathbb{E}_{\mathbf{w}}[G(\boldsymbol{\alpha},\boldsymbol{\mu},\mathbf{w})] &= \mathbb{E}_{\mathbf{w}}[\nabla J(\boldsymbol{\alpha},\boldsymbol{\mu},\mathbf{w})] \\
&= \int_{\mathbb{R}^{pT}} \nabla J(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}) p_{\mathbf{w}}(\boldsymbol{w}) d\boldsymbol{w} \\
&\overset{(a)}{=} \nabla \left[\int_{\mathbb{R}^{pT}} J(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}) p_{\mathbf{w}}(\boldsymbol{w}) d\boldsymbol{w} \right] \\
&= \nabla \bar{J}(\boldsymbol{\alpha},\boldsymbol{\mu})
\end{aligned}$$

where in (a) we exploited the Leibniz integral rule. This chain of equalities show that $G(\boldsymbol{\alpha},\boldsymbol{\mu},\mathbf{w})$ is an unbiased estimator of the gradient of $\bar{J}(\boldsymbol{\alpha},\boldsymbol{\mu})$. The proof follows. ■

Lemma 3.5. *Let Assumption 3.11 hold. Then, the stacks of the shooting maps* (3.143) *are Lipschitz continuous with Lipschitz continuous gradient.* △

Proof. Being the composition of Lipschitz functions with Lipschitz gradient, see Assumption 3.11, $\phi_t(\cdot,\cdot,\cdot)$ and $\psi_t(\cdot,\cdot,\cdot)$ are Lipschitz functions with Lipschitz gradient. Their stacks inherit the same properties. ■

Lemma 3.6. *The reduced cost function $\bar{J}(\boldsymbol{\alpha},\boldsymbol{\mu})$ has $L_{J'}$-Lipschitz gradient.* △

Proof of Lemma 3.6. First of all, given a noise realization $\boldsymbol{w}$ consider the gradient of $J(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w})$. It holds, for all $\boldsymbol{\alpha} \in \mathbb{R}^{nT}$ and $\boldsymbol{\mu} \in \mathbb{R}^{mT}$,

$$\begin{aligned}
\nabla J(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}) = &\underbrace{\nabla\phi(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w})\nabla_x \ell(\phi(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}),\psi(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}))}_{(a)} \\
&+ \underbrace{\nabla\psi(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w})\nabla_u \ell(\phi(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}),\psi(\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{w}))}_{(b)}.
\end{aligned}$$

namely, $\nabla J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w})$ is the sum of the two terms (a) and (b). Now we compute the Lipschitz constant of term (a). The gradient of ϕ is Lipschitz continuous by Lemma 3.5, moreover, since by the same lemma ϕ is Lipschitz continuous its gradient needs to be bounded. Gradient $\nabla_x \ell(\phi(\cdot), \psi(\cdot))$ is Lipschitz continuous being the composition of Lipschitz continuous functions (by Assumption 3.11 and Lemma 3.5) and bounded due to Assumption 3.12. It follows that term (a) is Lipschitz continuous. The same reasoning applies to (b). Hence, $\nabla J(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{w})$ is Lipschitz continuous being the sum of Lipschitz continuous functions. The proof follows. ■

Proof of Theorem 3.4

We are now in the position to prove Theorem 3.4. Our goal is to show that Algorithm 13 implements a SGD scheme, thus inheriting its convergence results (see Theorem A.2 in Appendix A.3), assuming that the reduced cost function is bounded from below, i.e., the problem (3.138) is well-posed. More specifically, in order to apply Theorem A.2, we need to verify

(i) The reduced cost function $\bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$ is continuously differentiable and the gradient of $\bar{J}$, namely, is Lipschitz continuous,

(ii) The stochastic direction $G(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathrm{w})$ is an unbiased estimator of $\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$,

(iii) The second moment of the random variable $\|G(x, \mathrm{w}) - \nabla \bar{\ell}(x, \mathrm{w})\|$ is bounded, i.e.,

$$\mathbb{E}_{\mathbf{w}}[\|G(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w}) - \nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})\|^2] \leq M^2. \tag{3.159}$$

for all $(\boldsymbol{\alpha}, \boldsymbol{\mu})$ and for $M > 0$.

As for condition (i), it follows from Lemma 3.6. Condition (ii) is ensured by Proposition 3.1). Finally, in order to check that condition (iii) holds, note that

$$\mathbb{E}_{\mathbf{w}}[\|G(\boldsymbol{\alpha}\text{'}\boldsymbol{\mu}, \mathbf{w}) - \nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})\|^2] = \underbrace{\mathbb{E}_{\mathbf{w}}[\|G(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})\|^2]}_{(a)} + \underbrace{\|\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})\|^2}_{(b)}$$

where we used the fact that $G(\boldsymbol{\alpha}, \boldsymbol{\mu}, \mathbf{w})$ is an unbiased estimator of $\nabla \bar{J}(\boldsymbol{\alpha}, \boldsymbol{\mu})$ (see Proposition 3.1). Boundedness of (a) can be proved leveraging on similar arguments as in Theorem 3.3, observing that the random variable $\mathbf{w}$ is bounded by assumption. As for (b) it follows being $\bar{J}$ and $\nabla \bar{J}$ continuous functions on compact sets. Hence, we can choose M sufficiently large such that (A.16) holds. The necessary conditions of Theorem A.2 are then satisfied, the proof follows. △

3.5.3 Numerical Simulations

We consider an inverted pendulum, with nonlinear continuous-time dynamics

$$Ml^2\ddot{\theta} = +Mgl\sin(\theta) - \mathrm{f}(w_t)l\dot{\theta} + u, \tag{3.160}$$

where θ is the angle, $\dot{\theta}$ is the velocity and u is the input with the same sign of θ. Moreover, $M = 1$ [kg] is the mass, $l = 1$ [m] is the pendulum length, g is the gravity acceleration, $\mathrm{f}(w_t)$ is the linear friction coefficient. We assume the friction coefficient to be uncertain and normally distributed with (known) mean value $\mathrm{f}_0 = 0.5$ [N m s/rad] and variance $\sigma_f = 3 \cdot 10^{-2}$. The discrete-time state-space representation of dynamics (3.160), discretized, via a forward Euler discretization with sampling period $\delta = 10^{-2}$, reads as

$$\begin{bmatrix} x_{1,t+1} \\ x_{2,t+1} \end{bmatrix} = \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} + \delta \begin{bmatrix} x_{2,t} \\ a\sin(x_{1,t}) + b(w_t)x_{2,t} \end{bmatrix} + \delta \begin{bmatrix} 0 \\ c \end{bmatrix} u_t$$

where $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ correspond, respectively, to the angle θ and the velocity $\dot{\theta}$, while $a = \frac{g}{l}$, $b = -\frac{\mathrm{f}(w_t)}{Ml}$ and $c = \frac{1}{Ml^2}$.

We consider a tracking problem, which translates in the quadratic cost function $\sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) = \sum_{t=0}^{T-1} \|x_t - x_{\text{ref},t}\|_Q^2 + \|u_t - u_{\text{ref},t}\|_R^2 + \|x_T - x_{\text{ref},T}\|_{Q_f}^2$ where $Q = \text{diag}(10, 1)$, $Q_f = \text{diag}(10, 100)$ and $R = 10^{-3}$. The horizon length is set to $T = 10$, [s]. As reference trajectory we used a step signal between two equilibrium configurations, namely from $(x_{\text{ref, i}}, u_{\text{ref, i}}) = (0, 0, 0)$ to $(x_{\text{ref, f}}, u_{\text{ref, f}}) = (0.5\text{rad}, 0, -M_0 gl\sin(0.5\text{rad}))$.

We implemented Algorithm 13 in a Heavy-Ball fashion, cf. Section 3.2.2. Moreover, we implemented the batch version of the SGD algorithm, where, at each iteration, the gradient estimator G computed via Algorithm 12 is calculated as the empirical mean over a batch of $\mathcal{B} = 50$ realizations of w. These improvements provide better convergence rate. In Figure 3.17 we show the optimal state-input trajectory computed by Algorithm 13. The trajectory is computed in correspondence of a single realization of **w**.

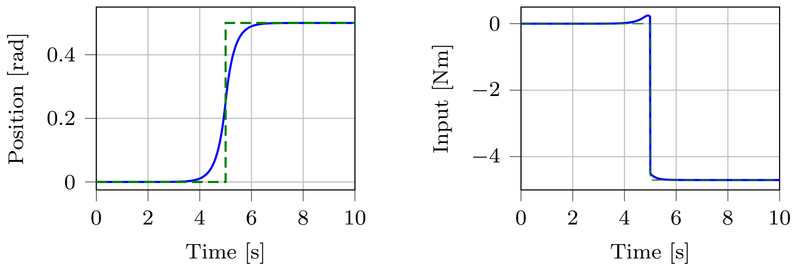
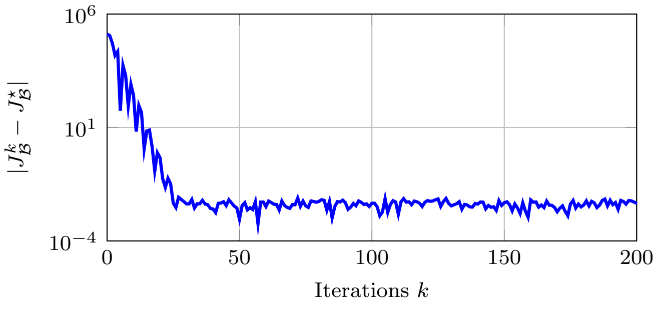


Figure 3.17: Optimal position and input trajectory computed by Algorithm 13.

The cost error is represented on the left of Figure 3.18 and shows the difference between an approximation of both $\bar{J}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$, the expected cost evaluated at the k-th iteration, and of $\bar{J}(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star)$, the cost at the last iteration. The values of $\bar{J}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$ are approximated by the empirical mean computed over a batch of $\mathcal{B} = 50$ realizations of $\mathbf{w}$. Notice that the cost error diminishes and converges to a neighborhood of the optimal value. This is due to the approximation introduced by the empirical mean used to compute the expected values $\bar{J}$. On the right of Figure 3.18, the norm of the empirical mean among the directions computed by Algorithm 12 over a batch of $\mathcal{B} = 50$ realizations of $\mathbf{w}$ is depicted. The norm decreases as expected, but it saturates to a negligible value due to the approximation introduced by the empirical mean.



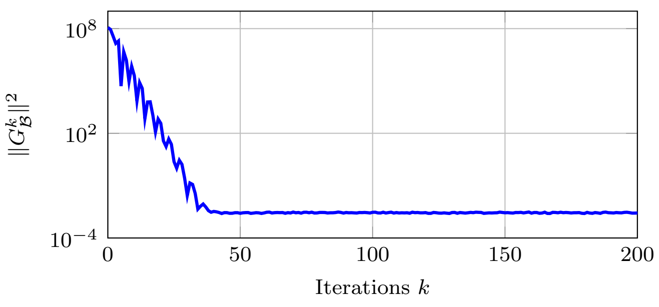


Figure 3.18: (Left) Cost error: difference between $\bar{J}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k) \approx \frac{1}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} J(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^i) =: J_{\mathcal{B}}^k$ and of $\bar{J}(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star) \approx \frac{1}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} J(\boldsymbol{\alpha}^\star, \boldsymbol{\mu}^\star, \boldsymbol{w}^i) =: J_{\mathcal{B}}^\star$, for all $k \geq 0$. (Right) Descent direction: norm of the descent $\left[\begin{smallmatrix}\boldsymbol{\Delta\alpha} \\ \boldsymbol{\Delta\mu}\end{smallmatrix}\right] = \frac{1}{\mathcal{B}} \sum_{i=1}^{\mathcal{B}} G(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k, \boldsymbol{w}^i) =: G_{\mathcal{B}}^k$, for all $k \geq 0$.

Chapter 4

Optimization-based Safe Control of Multi-layer Systems

In this chapter, we focus on the design of safe trajectory generation strategies for continuous time control systems. The core of our investigation are multi-layer control systems, i.e., control architectures consisting of (at least) two, interconnected, dynamical systems with different roles and objectives. We propose an high-level, continuous-time trajectory generation strategy, based on optimal control, which ensures satisfaction of safety-critical constraints. In this optimal control formulation, we ensure constraint enforcement through Control Barrier Functions (CBFs). To this end, we provide a new characterization of CBFs as necessary and sufficient for the controlled invariance of safe sets. The results of this chapter are based on [211].

4.1 Literature Review

In recent years, the challenge of designing safe trajectories for autonomous and robotic systems has intensified. The concept of safety, in a formal sense, is linked to maintaining the forward invariance of a so-called safe set, as explored in [37]. The framework of Control Barrier Functions (CBFs), detailed in [7], presents a mechanism for ensuring safety in nonlinear dynamics through safety filters. These filters, manifesting as quadratic programs (QPs), aim to provide point-wise optimal control inputs that minimally alter potentially unsafe control inputs to maintain safety.

Nevertheless, a significant limitation is the potential of the CBF approach for myopic behavior, allowing trajectories to closely approach the boundary of the safe set before intervention. This might lead to QP infeasibility and failure, as discussed in [241]. Furthermore, this framework may introduce undesired minima, a challenge highlighted in [189]. The generation of safe trajectories that are both optimal and feasible is addressed through state constrained optimization problems, as investigated in [78, 97].

Recent studies have delved into the relationship between CBFs and optimal control, as seen in [5, 6, 61, 66]. Additionally, methodologies based on Hamilton-Jacobi-Bellman (HJB) equations for the development of CBFs have been examined in [64, 226]. An innovative approach for controller synthesis, employing high-order CBFs to track optimally generated trajectories from unconstrained control problems, is proposed in [242].

Recently, the design of multi-layer control schemes gained traction in developing safe control policies. In the precursory work [193] it is proposed a layered control approach based on Model Predictive Control. This method utilizes a layered architecture, combining a low-level controller with a high-level planner operating at different frequencies. This approach has seen successful application across diverse areas, from robotics [2, 163, 195] to the control of feedback linearizable systems [68].

4.2 Safe Trajectory Generation for Multi-layer Control Architectures

In this section we introduce the concept of multi-layer control architectures together with the problem setup. Next, we provide some preliminaries on safe trajectory generation.

Multi-layer Control Architectures

A multi-layer control architecture consists of (at least) two, interconnected, dynamical systems with different roles and objectives. In this work, we consider an architecture where a trajectory generation layer is coupled with a low-level tracking controller. More formally, we frame this control architecture considering (i), a *low-level* model, with dynamics

$$\dot{x}_\ell(t) = f_\ell(x_\ell(t), u_\ell(t)) \tag{4.1}$$

where $f_\ell : \mathbb{R}^n \to \mathbb{R}^n$ and state $x_\ell \in \mathbb{R}^{n_\mathrm{l}}$, and input $u_\ell \in \mathbb{R}^{m_\mathrm{l}}$. This dynamics usually captures the "true" system dynamics. And, (ii) an *high-level* reference model

$$\dot{x}_\mathrm{h}(t) = f_\mathrm{h}(x_\mathrm{h}(t), u_\mathrm{h}(t)) \tag{4.2}$$

where $f_\mathrm{h} : \mathbb{R}^{n_\mathrm{h}} \to \mathbb{R}^{n_\mathrm{h}}$ and $x_\mathrm{h} \in \mathbb{R}^{n_\mathrm{h}}$, high-level state, $u_\mathrm{h} \in \mathbb{R}^{m_\mathrm{h}}$, high-level control signal. The two models are interconnected via a projection map $\pi_{x_\ell}(x_\ell) = x_\mathrm{h}$ relating low- and the high-level state, and an embedding of the high-level input u_h into the low-level dynamics via $\pi_{u_\mathrm{h}}(u_\mathrm{h}) = u_\ell$. Given a sampling time $T_s \in \mathbb{R}_{\geq 0}$, the set $\mathcal{T} = \cup_{i=0}^{\infty}(iT_s, (i+1)T_s)$, $i \in \mathbb{N}$, collects the open time intervals from time iT_s to time $(i+1)T_s$, and its complement $\mathcal{T}^c = \cup_{i=0}^{\infty}\{iT_s\}$ collects the time instances iT_s. In general, while the control input $u_\ell(t)$ is designed such that the low-level state $x_\ell(t)$ tracks the projected high-level state trajectory $x_{\mathrm{h}}(t)$ for all $t \in \mathcal{T}$, the high-level state-input trajectory is updated only at discrete time instants, i.e., each $t \in \mathcal{T}^c$. In this work, we design a control law such that the state trajectory $x_{\mathrm{h}}(t)$, $t \in \mathcal{T}$, updated at each $t \in \mathcal{T}^c$, satisfies some user-defined safety constraints.

Safe Trajectory Generation for Robotic Systems

The high-level dynamics is represented as a nonlinear system with control affine dynamics

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t), \qquad x(0) = x_{\text{init}} \tag{4.3}$$

with state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, input $u \in \mathcal{U} \subseteq \mathbb{R}^n$, drift dynamics $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$, $x_{\text{init}} \in \mathbb{R}^n$ is the initial condition. For notational convenience, we drop the h subscript. We assume $x(t) \equiv 0$ is an equilibrium of (4.3) if $u(t) \equiv 0$ and $\mathcal{X}$ is an open and connected neighborhood of $x = 0$. The system (4.3) is assumed to be forward complete. In the considered control scenario, the state $x(t)$ is constrained to satisfy state constraints in the form

$$h(x(t)) \geq 0, \qquad t \geq 0 \tag{4.4}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. These constraints may represent some safety conditions for dynamics (4.3), i.e., they can encode a set $\mathbb{S}$ of *safe* states

$$\mathbb{S} := \{x(t) \in \mathbb{R}^n \mid h(x(t)) \geq 0\} \subseteq \mathcal{X}. \tag{4.5}$$

The goal is to ensure that $\mathbb{S}$ is forward invariant. Note that $h(x)$ implicitly depends on $u(t)$ via the nonlinear dynamics (4.3). That is, since $x(t)$ needs to satisfy (4.3) $\forall t \geq 0$, differentiating h with respect to time yields

$$\dot{h}(x,u) = \underbrace{\frac{\partial h}{\partial x}|_x f(x)}_{:=L_f h(x)} + \underbrace{\frac{\partial h}{\partial x}|_x g(x)}_{:=L_g h(x)} u. \tag{4.6}$$

This observation allows us to leverage the CBF formalism.

Definition 4.1 (CBF). *Let $\mathbb{S} \subset \mathbb{R}^n$ be the 0-superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ with 0 regular value*[1]. *Then, $h(x)$ is a CBF for the system* (4.3) *if there*

[1]A point $a \in \mathbb{R}$ is a regular value of h if $\frac{\partial h}{\partial x} \neq 0$ for all x s.t. $h(x) = a$.

exists a class $\mathcal{K}$ function[2] α *such that, for all* $x \in \mathcal{X}$*,*

$$\sup_{u\in\mathbb{R}^m} \left[\dot{h}(x,u)\right] = \sup_{u\in\mathbb{R}^m} \left[L_f h(x) + L_g h(x) u\right] \geq -\alpha(h(x)) \tag{4.7}$$

Notice that, an alternative formulation can be considered. The function h is a CBF if (4.7) is replaced by

$$L_g h(x) = 0 \Longrightarrow L_f h(x) > -\alpha(h(x)). \tag{4.8}$$

CBFs are widely employed in the design of safety filters, which are known to be only point-wise optimal. In this work, we propose a design strategy to generate a control action $u(t)$ such that the resulting system trajectory $(\boldsymbol{x}, \boldsymbol{u})$[3] minimizes a user-defined cost functional with instantaneous cost $q(x(t)) + u(t)^\top u(t)$, where $q : \mathbb{R}^n \to \mathbb{R}$ positive-definite function, i.e., $q(x) > 0$ for all $x \neq 0$, and continuously differentiable, i.e., of class $\mathcal{C}^1$.

Review of safe optimization-based control

In this section, we recall different optimization-based strategies to generate safe trajectories for robotic systems.

Constrained Optimal Control: Consider the case where the state is constrained to satisfy the safety constraints (4.4). Given an initial condition $x_{\text{init}} \in \mathcal{X}$, a safe trajectory can be retrieved by designing a controller $\Phi^{\text{opt}}_{\text{safe}} : \mathbb{R}^n \to \mathbb{R}^m$, such that the resulting state-input trajectory is solution of

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u}} \quad & \int_0^\infty q(x) + u^\top u \, \mathrm{d}\tau && (4.9\text{a}) \\ \text{subj.to} \quad & \dot{x} = f(x) + g(x)u, \quad x(0) = x_{\text{init}} && (4.9\text{b}) \\ & h(x) \geq 0 && (4.9\text{c}) \end{aligned}$$

where constraint (4.9c) ensures the trajectory $(\boldsymbol{x}^{\text{safe}}, \boldsymbol{u}^{\text{safe}})$ solution of (4.9) to be safe. In general, problem (4.9) is solved numerically leveraging on Euler-Lagrange equations or calculus of variations (cf. [47, 148]).

Remark 4.1. In the unconstrained case, i.e., without (4.9c), the optimal feedback controller $\Phi^{\text{opt}} : \mathbb{R}^n \to \mathbb{R}^m$ such that for an initial condition x_{init} the corresponding state-

[2]A scalar function, $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing, and it is such that $\alpha(0) = 0$.

[3]For a given initial condition x_{init}, a given control signal $u(t)$ – or feedback controller Φ, s.t., $u(t) = \Phi(x(t))$ – and time interval $[t_0, t_f]$, $t_0, t_f \in \mathbb{R}_{\geq 0}$, $t_f > t_0$, we denote as $\boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^n$ the state trajectory solution of (4.3) with initial condition x_{init}. Similarly, $\boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^m$ represent the corresponding input signal.

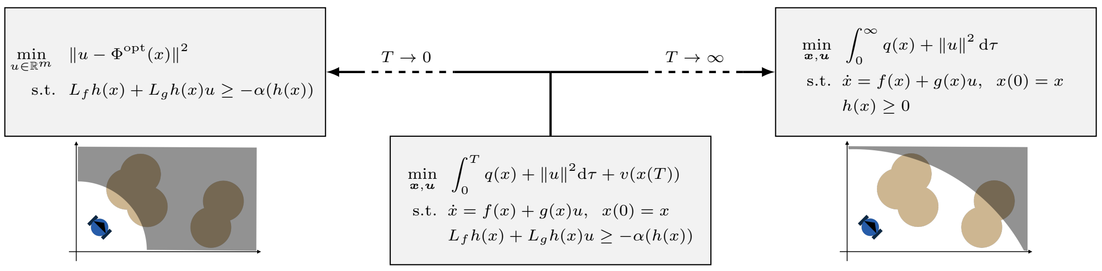

$\min_{u\in\mathbb{R}^m} \|u-\Phi^{\text{opt}}(x)\|^2$ s.t. $L_fh(x)+L_gh(x)u \geq -\alpha(h(x))$
$T\to 0$
$T\to\infty$
$\min_{\boldsymbol{x},\boldsymbol{u}} \int_0^\infty q(x)+\|u\|^2\,\mathrm{d}\tau$ s.t. $\dot{x}=f(x)+g(x)u,\ x(0)=x$, $h(x)\geq 0$
$\min_{\boldsymbol{x},\boldsymbol{u}} \int_0^T q(x)+\|u\|^2\mathrm{d}\tau + v(x(T))$ s.t. $\dot{x}=f(x)+g(x)u,\ x(0)=x$, $L_fh(x)+L_gh(x)u\geq -\alpha(h(x))$

Figure 4.1: Conceptual representation of the proposed control strategy considering a collision avoidance task (obstacles in orange, robot in blue). As the parameter T approaches zero, a safety filter is applied, prioritizing close obstacles for collision avoidance. Conversely, as T approaches infinity, the strategy retrieves the safe optimal controller, enabling the generation of trajectories over the entire map for the collision avoidance task.

input trajectory $(\boldsymbol{x}^{\text{opt}}, \boldsymbol{u}^{\text{opt}})$ minimizes the cost functional (4.9a) can be characterized via HJB equations. Define the *value function* $J^* : \mathbb{R}^n \to \mathbb{R}$ as the function that associates, to each initial condition $x_{\text{init}} \in \mathcal{X}$ the cost associated to the optimal controller Φ^{opt}. This function satisfies the *HJB equation*, i.e., for all t, (where the time index has been suppressed)

$$0 = \min_{u\in\mathbb{R}^m} \left[\frac{\partial J^*}{\partial x}\bigg|_x (f(x)+g(x)u) + u^\top u + q(x) \right]. \tag{4.10}$$

Importantly, the optimal controller can be found by solving the minimization problem (4.10), i.e.,

$$u = \Phi^{\text{opt}}(x) = -\tfrac{1}{2} g^\top(x) \left(\frac{\partial J^*}{\partial x}\bigg|_x \right)^\top := -\tfrac{1}{2} L_g J^*(x)^\top \tag{4.11}$$

Substituting (4.11) into (4.10) results in a reformulation of the HJB equation that is only function of the state. This partial differential equation, if it can be solved, yields J^* and therefore the optimal controller (4.11). $\triangle$

Safety Filters: Definition 4.1 leads to an entire class of safe controllers that makes $\mathbb{S}$ forward invariant. As shown in [7], any locally Lipschitz continuous controller $u = \Phi(x)$ s.t. $L_fh(x) + L_gh(x)u \geq -\alpha(h(x))$ for all $x \in \mathbb{S}$ renders (4.3) safe w.r.t. $\mathbb{S}$. A standard example is represented by safety filters [7]. Given a desired (but not necessarily safe) controller $\Phi^{\text{d}}(x)$, a minimally invasive safety critical controller can be synthesized solving the QP

$$\begin{aligned} &\min_{u\in\mathbb{R}^m} \ \|u-\Phi^{\text{d}}(x)\|^2 \\ &\text{subj.to} \ \ L_fh(x)+L_gh(x)u \geq -\alpha(h(x)). \end{aligned} \tag{4.12}$$

Our objective is to design a control strategy that bridges safety filters and constrained optimal control. In order to do that, we firstly need to extend the concept of CBFs to control systems.

4.2.1 CBF Characterization of Control Invariance

In this section, CBFs are extended to control systems (instead of dynamical systems) where direct knowledge of the specific controller yielding forward invariance is not required. We first make precise the notion of control invariance:

Definition 4.2. *The constraint set* $\mathbb{S} = \{x \ \in \mathbb{R}^n : h(x) \geq 0\}$ *is* control invariant *if there exists a controller* $u = \Phi(x)$ *such that* $\mathbb{S}$ *is forward invariant for the closed loop system* $\dot{x} = f(x) + g(x)\Phi(x)$*, i.e.,* $x(0) \in \mathbb{S}$ *implies* $x(t) \in \mathbb{S}\ \forall\, t \geq 0$.

It was established in [7] that, in the case when $\mathbb{S}$ is compact, barrier functions are necessary and sufficient for forward set invariance, i.e., invariance of $\mathbb{S}$ is equivalent to:

$$\dot{h}(x) \geq -\alpha_s(h(x)) \qquad \forall\, x \in \mathbb{S}, \tag{4.13}$$

for some $\mathcal{K}$ function α_s. Since this result was proven for dynamical systems, and not control systems, it requires knowledge of a feedback controller $u = \Phi(x)$ that renders $\mathbb{S}$ forward invariant (i.e., Φ determines α_s). We extend this result to control systems such that direct knowledge of the feedback controller is not required.

Theorem 4.1. *Let Assumption 4.1 hold, assume* $\mathbb{S}$ *to be control invariant and* h *to have constant rank on* $\mathbb{S}$*. Then there exists a constant* $\kappa_{\min} > 0$ *s.t.*

$$\sup_{u\in\mathbb{R}^m} \dot{h}(x,u) \geq -\kappa h(x), \qquad \forall\, \kappa > \kappa_{\min},\ x \in \mathbb{S}. \tag{4.14}$$

Namely, CBFs with linear class $\mathcal{K}$ *functions* $\alpha(r) = \kappa r$ *are necessary and sufficient for control invariance, i.e.,* $\forall \kappa \geq \kappa_{\min}$

$$\mathbb{S}\ \textit{control invariant} \quad \Longleftrightarrow \quad \sup_{u\in\mathbb{R}^m} \dot{h}(x,u) \geq -\kappa h(x). \qquad \triangle$$

Proof. For any $r \geq 0$, the set $h^{-1}_{[0,r]} = \{x \in \mathbb{R}^n : 0 \leq h(x) \leq r\}$ is a compact subset of $\mathbb{S}$. Under the assumption that $\mathbb{S}$ can be rendered forward invariant by a feedback controller $u = \Phi(x)$, since 0 is a regular value of h:

$$\sup_{u\in\mathbb{R}^m} \dot{h}(x,u) \geq \dot{h}(x,\Phi(x)) \geq 0 \qquad \forall\, x \in \partial\mathbb{S}.$$

Define function $\gamma : \mathbb{R} \to \mathbb{R}$ by:

$$\gamma(r) := -\inf_{x \in h^{-1}_{[0,r]}} \Big[\dot{h}(x, \Phi(x))\Big]. \tag{4.15}$$

Since $h^{-1}_{[0,0]} = \partial\mathbb{S}$, we have that $\gamma(0) \le 0$, cf. (4.13). Being $\gamma(r)$ non-decreasing, for $r_2 > r_1$, since $h^{-1}_{[0,r_1]} \subset h^{-1}_{[0,r_2]}$, it holds $\gamma(r_2) \ge \gamma(r_1)$. Hence, it follows that

$$\alpha_s(r) := \gamma(r) - \gamma(0) + \alpha(r) \ge \gamma(r) \tag{4.16}$$

for any class $\mathcal{K}$ function α. We conclude that $\alpha_s(0) = 0$, α_s is strictly increasing, and $-\alpha_s(r) \le -\gamma(r)$ Therefore, α_s is a class $\mathcal{K}$ function satisfying:

$$\begin{aligned}\sup_{u \in \mathbb{R}^m} \dot{h}(x,u) \ge \dot{h}(x, \Phi(x)) \ge \inf_{x \in h^{-1}_{[0,h(x)]}} \Big[\dot{h}(x, \Phi(x))\Big] \\ = -\gamma(h(x)) \ge -\alpha_s(h(x)),\end{aligned} \tag{4.17}$$

from which the necessary condition in (4.13) derives. To establish (4.14) we must further characterize γ.

Lemma 4.1. *The function γ given in* (4.15) *is continuously differentiable if h has constant rank ($\frac{\partial h}{\partial x}|_x \neq 0$ for all $x \in \mathbb{S}$).* $\triangle$

Proof. Note that γ can be expressed as:

$$\begin{aligned}\gamma(r) = \max_{x \in \mathbb{S}} \ & -\dot{h}(x, \Phi(x)) \\ \text{s.t. } & h(x) \ge 0, \quad h(x) \le r\end{aligned} \tag{4.18}$$

where $\sup = \max$ since $\mathbb{S}$ is compact. We can view r as a "parameter" of this optimization problem, and thereby leverage the Envelope Theorem. Consider the Lagrangian associated with the optimization problem (4.18):

$$\mathcal{L}(x, \mu, r) = -\dot{h}(x, \Phi(x)) + \mu_1 h(x) + \mu_2 (r - h(x))$$

with Lagrange multipliers $\mu = (\mu_1, \mu_2) \in \mathbb{R}^2$. Let $x^*(r)$ and $\mu^*(r)$ be a solution of (4.18), by the Envelope Theorem,

$$\frac{\partial \gamma}{\partial r}\Big|_r = \frac{\partial \mathcal{L}}{\partial r}\Big|_{(x^*(r), \mu^*(r), r)} = \mu_2^*(r)$$

Therefore, γ is continuously differentiable, with derivative $\mu_2^*(r)$, if μ_2 is continuous. To establish this, consider the function $F : (\mathbb{S} \times \mathbb{R}) \times \mathbb{R} \to \mathbb{R}$ given by:

$$F((x, \mu_1), \mu_2) := -\frac{\partial \dot{h}}{\partial x}\Big|_{(x,\Phi(x))} + \mu_1 \frac{\partial h}{\partial x}\Big|_x - \mu_2 \frac{\partial h}{\partial x}\Big|_x$$

The solution $(x^*(r), \mu^*(r))$ must be a saddle point of the Lagrangian, i.e., $F((x^*, \mu_1^*), \mu_2^*) = 0$. Then, by the implicit function theorem, $\mu_2(x, \mu_1)$ is locally smooth if

$$\frac{\partial F}{\partial \mu_2^*(r)}\Big|_{(x^*(r), \mu_1^*(r))} = -\frac{\partial h}{\partial x}\Big|_{x^*(r)} \neq 0.$$

This holds by assumption that h has constant rank. It follows that there exists a continuously differentiable function G such that $\mu_2^*(r) = G(x^*(r), \mu_1^*(r))$ and $F((x, \mu_1), G(x, \mu_1)) = 0$ for all x, μ_1 in a neighborhood W of $(x^*(r), \mu_1^*(r))$. Therefore, locally, it holds that

$$\frac{\partial \gamma}{\partial r}\Big|_r = G(x, \mu_1), \qquad \forall\, (x, \mu_1) \in W.$$

As this holds uniformly over $\mathbb{S}$, it follows that γ is continuously (twice) differentiable. ■

Proof of Theorem 4.1 (Continued). The fact that γ is continuously differentiable implies that it is locally Lipschitz. Since $\mathbb{S}$ is compact, let $r_{\max} = \max_{\mathbb{S}} h(x)$ wherein $\gamma : [0, r_{\max}] \to \mathbb{R}$. It follows that γ is globally Lipschitz on $[0, r_{\max}]$. Call this Lipschitz constant L_γ, wherein $\gamma(r) - \gamma(0) \leq L_\gamma r$. Additionally, pick $\alpha(r) = kr$ with $k > 0$, i.e., a linear class $\mathcal{K}$ function. Then (4.16) yields:

$$\begin{aligned} \gamma(r) &\leq L_\gamma r + kr = \kappa_{\min} r, \qquad \kappa_{\min} := L_\gamma + k \\ &\implies \quad -\gamma(r) \geq -\alpha_s(r) \geq -\kappa r, \qquad \kappa \geq \kappa_{\min} \end{aligned}$$

From (4.17), it follows that (4.14) holds with $\kappa_{\min} = L_\gamma + k$. ■

Remark 4.2. Note that the assumption that $\mathbb{S}$ is compact and h has constant rank is, in fact, rather strong. Compact sets often results in points $\frac{\partial h}{\partial x}|_x \neq 0$. Yet, these points (and the potential loss of differentiability in γ) are not practically relevant if they are sufficiently far from the boundary, i.e., $h > 0$ large. To see this, note that even if γ is not differentiable (4.17) holds. And, for $\kappa > 0$ sufficiently large, it follows $\alpha_s(h) < \kappa h$ (since $h > 0$ is large). △

Remark 4.3. While calculating $\kappa_{\min}$ requires knowledge of the optimal safe controller $\Phi_{\text{safe}}^{\text{opt}}$, Theorem 4.2 holds for all $\kappa > \kappa_{\min}$. Therefore, taking κ sufficiently large means that direct knowledge of $\Phi_{\text{safe}}^{\text{opt}}$ is not required for (4.19c) = (4.9c). △

4.2.2 Safe Trajectory Generation via CBF-based Receding Horizon Controllers

In this section, we present a receding horizon scheme that extends the concept of CBFs to incorporate a user-defined prediction horizon. Leveraging on the length of the time-horizon, the resulting system trajectory achieves a balance between safety filters (as $T \to 0$) and safe optimal control (as $T \to \infty$). This idea is conceptually represented in Figure 4.1. Consider the following optimal control problem where, for notational convenience, we dropped the dependence on the integration variable

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \int_0^T q(x) + u^\top u \,\mathrm{d}\tau + v(x(T)) \tag{4.19a}$$

$$\text{subj.to} \quad \dot{x} = f(x) + g(x)u, \quad x(0) = x_{\text{init}} \tag{4.19b}$$

$$L_f h(x) + L_g h(x) u \geq -\alpha(h(x)), \tag{4.19c}$$

where $T \in \mathbb{R}_{\geq 0}$ is the *time-horizon*, the term $q(x) + u^\top u$ is the Lagrange term, $v : \mathbb{R}^n \to \mathbb{R}$ is the Meyer term, a positive-definite function of class $\mathcal{C}^1$, and α some class $\mathcal{K}$ function. Notice that, differently from (4.9), in this formulation we impose the barrier condition (4.19c) instead of (4.9c). In the following we show that, ***(i)*** a safe trajectory for system (4.3) is generated for all $T \geq 0$, ***(ii)*** for a specific choice of class $\mathcal{K}$ function $\alpha(h) = \kappa h$, for $\kappa > 0$ sufficiently large, as $T \to \infty$ problem (4.19) is equivalent to the constrained optimal control problem (4.9) ***(iii)*** for a specific choice of the terminal cost $v(\cdot)$, as $T \to 0$ the safety filter (4.12) over the optimal unconstrained controller (4.11) is retrieved.

Assumption 4.1. *The safe set $\mathbb{S} = \{x \in \mathbb{R}^n : h(x) \geq 0\}$ is compact with* 0 *regular value of h.* △

Lemma 4.2. *Let Assumption 4.1 hold. Then, for all $T \geq 0$ and for any class $\mathcal{K}$ function α, the optimal solution $(\boldsymbol{x}^{\text{rhc}}, \boldsymbol{u}^{\text{rhc}})$ of* (4.19)*, is a safe trajectory for* (4.3) *w.r.t. the safe set $\mathbb{S}$.* △

Proof. It follows observing that (4.19c) implies (4.4). ■

Theorem 4.2. *Let Assumption 4.1 hold and assume that h has constant rank. Then, as $T \to \infty$, there exists a constant $\kappa_{\min} > 0$, such that, for all $\mathcal{K}$ functions $\alpha(r) = \kappa r$ with $\kappa \geq \kappa_{\min}$, $r \in \mathbb{R}$, the CBF constrained optimal control problem* (4.19) *is equivalent to the state constrained optimal control problem* (4.9). △

Remark 4.4. Given a class $\mathcal{K}$ function α one obtains suboptimal solutions to the optimal control problem (4.9). In fact, different class $\mathcal{K}$ functions, while they enforce $h(x) \geq 0$, they may require more control effort to do so. △

Proof. First, notice that, as $T \to \infty$, (4.19a) goes as (4.9a). Let $\Phi^{\text{rhc}}_{\infty,\kappa}$ be the controller such that the closed-loop trajectory $(\boldsymbol{x}^{\text{rhc}}, \boldsymbol{u}^{\text{rhc}})$ is the solution of (4.19) as $T \to \infty$, for a class $\mathcal{K}$ function of the form $\alpha(r) = \kappa r$. Define $J_{\rm rhc}(x, \Phi^{\rm rhc}_{\infty,\kappa})$ as the optimal cost of problem (4.19) as $T \to \infty$ associated to $\Phi^{\rm rhc}_{\infty,\kappa}$ with initial condition x. Similarly, define as $J_{\rm safe}(x, \Phi^{\rm opt}_{\rm safe})$ the optimal cost of problem (4.9) associated to the optimal controller $\Phi^{\rm opt}_{\rm safe}$. To prove Theorem 4.2, we show that $\exists \kappa_{\min} \geq 0$, s.t.,, $J_{\rm rhc}(x, \Phi^{\rm rhc}_{\infty,\kappa}) = J_{\rm safe}(x, \Phi^{\rm opt}_{\rm safe})$, establishing inequalities in both directions. We begin by showing that $J_{\rm safe}(x, \Phi^{\rm opt}_{\rm safe}) \leq J_{\rm rhc}(x, \Phi^{\rm rhc}_{\infty,\kappa})\ \forall \kappa > 0$. We need to establish that the system trajectory associated with $\Phi^{\rm rhc}_{\infty,\kappa}(x)$ represents a suboptimal solution to problem (4.9), i.e., the trajectory is feasible for problem (4.9). This follows from the fact that (4.19c) implies (4.9c) for every class $\mathcal{K}$ function α, and therefore $\alpha(r) = \kappa r$ for all $\kappa > 0$. Next, we prove that $J_{\rm rhc}(x, \Phi^{\rm rhc}_{\infty,\kappa}) \leq J_{\rm safe}(x, \Phi^{\rm opt}_{\rm safe})$, $\forall \kappa \geq \kappa_{\min} > 0$. Namely, it must be established that $\dot{h}(x, \Phi^{\rm opt}_{\rm safe}(x)) \geq -\kappa h(x)$ for all $\kappa \geq \kappa_{\min}$ That is, that (4.9c) implies (4.19c), for certain $\kappa_{\min}$ associated with $\Phi = \Phi^{\rm opt}_{\rm safe}$. ■

This last result will be established in Section 4.2.1 via Theorem 4.1, but first, we consider the case when $T \to 0$.

Theorem 4.3. *Let Assumption 4.1 hold. Moreover, let $v(\cdot)$ be chosen as the value function $J^*(\cdot)$ associated to the unconstrained optimal control problem, cf.* (4.1). *Then, as $T \to 0$, the optimal controller solution of* (4.19) *is equivalent to the safety filter* (4.12) *with $\Phi^{\rm d} = \Phi^{\rm opt}$.*△

Proof. With $x = x(t)$, $u = u(t)$, we expand $x(t+T) = x + (f(x) + g(x)u)\,T + o(T)$ and $v(x(t+T)) = v(x) + \big[L_f v(x) + L_g v(x) u\big]\,T + o(T)$. Notice that $\int_0^T q(x) + u^\top u \; \mathrm{d}\tau = \big[q(x) + u^\top u\big]\,T + o(T)$. We can now substitute the above equations in (4.19) and take the limit as $T \to 0$, thus obtaining

$$\begin{aligned} \min_{u} \;\; & q(x) + u^\top u + v(x) + L_f v(x) + L_g v(x) u \\ \text{subj.to} \;\; & L_f h(x) + L_g h(x) u \geq -\alpha(h(x)). \end{aligned} \tag{4.20}$$

namely, the minimum is taken over the instantaneous value of u at time t in correspondence of the state $x(t)$. The Lagrangian of (4.20), with Lagrange multiplier $\mu \in \mathbb{R}$, reads

$$\begin{aligned} \mathcal{L}(u,\mu) =& q(x) + r(u) + v(\hat{x}) + L_f v(x) + L_g v(x) u \\ & + \mu\big[L_f h(x) + L_g h(x) u + \alpha(h(x))\big]. \end{aligned} \tag{4.21}$$

Suppressing the dependence of all functions on x for notational simplicity, we can write the KKT conditions as

$$\text{KKT conditions} \begin{cases} u = -\frac{1}{2}L_g v^\top - \frac{1}{2}\mu L_g h \\ \mu \geq 0 \\ L_f h + L_g h u + \alpha(h) \geq 0 \\ \mu[L_f h + L_g h u + \alpha(h)] = 0 \end{cases} \tag{4.22}$$

which are necessary and sufficient since the cost is quadratic in u, and the constraints are affine. From (4.22), we have

$$L_f h + L_g h u + \alpha(h) > 0 \Rightarrow \mu = 0 \ \Rightarrow u = -\tfrac{1}{2}L_g v^\top$$

and, with $\mu > 0$, $L_f h + L_g h u + \alpha(h) = 0$

$$\begin{aligned} &\Rightarrow \mu = -2\frac{L_f h + L_g h\big(-\frac{1}{2}L_g v^\top\big) + \alpha(h)}{L_g h L_g h^\top} = \frac{2\mathbf{S}}{\|L_g h\|^2} \\ &\Rightarrow u = \mathbf{S}\frac{L_g h^\top}{\|L_g h\|^2} - \tfrac{1}{2}L_g v^\top \end{aligned}$$

where $\mathbf{S} := L_f h + L_g h\big(-\frac{1}{2}L_g v^\top\big) + \alpha(h)$. Therefore, we define the controller solution of (4.19) with $T \to 0$ as

$$\Phi_0^{\text{rhc}}(x) := -\tfrac{1}{2}L_g v(x)^\top + \begin{cases} 0 & \mathbf{S}(x) \leq 0 \\ \mathbf{S}(x)\frac{L_g h(x)^\top}{\|L_g h(x)\|^2} & \mathbf{S}(x) > 0. \end{cases} \tag{4.23}$$

Now, consider the safety filter (4.12) applied to the optimal unconstrained controller $\Phi^{\text{opt}} = -\frac{1}{2}L_g J^{*\top}$, i.e., $\Phi^{\text{d}} = \Phi^{\text{opt}}$. Choosing the Meyer term to be the optimal unconstrained value function, i.e., $v = J^*$, and noticing that for $\Phi^{\text{d}} = -\frac{1}{2}L_g J^{*\top}$ and $v = J^*$,

$$\underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \|u - \Phi^{\text{d}}\|^2 = \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \left[u^\top u + L_g J^* u\right] \tag{4.24}$$

which is the argument minimizing (4.20). Since the constraints both in (4.12) and (4.20) are the same, the result follows. ■

4.2.3 Numerical Simulations

In this section, we present numerical simulations demonstrating the implementation of the proposed control strategy within a two-layer control architecture modeled as

$$\begin{aligned} \textbf{High:}\quad & \dot{x}_\text{h}(t) = f_\text{h}(x_\text{h}(t)) + g_\text{h}(x_\text{h}(t))u_\text{h} \\ \textbf{Low:}\quad & \dot{x}_\ell(t) = f_\ell(x_\ell(t)) + g_\ell(x_\ell(t))u_\text{fb}(e(t), u_\text{ff}(t)) + w \end{aligned} \qquad t \in \mathcal{T}$$

$$\begin{aligned} \textbf{Coupling:}\quad & x_\text{h}(iT_s) = \pi_{x_\ell}(x_\ell(iT_s)), \quad u_\text{ff}(t) = \pi_{u_\text{h}}(u_\text{h}(t)), \\ & e(t) = x_\text{h}(t) - \pi_{x_\ell}(x_\ell(t)) \end{aligned}$$

where $u_\text{fb} : \mathbb{R}^{n_\text{h}} \times \mathbb{R}^{m_\text{l}} \to \mathbb{R}^m$ is a feedback controller. The interconnection between the layers relies on a feedforward term $u_\text{ff} \in \mathbb{R}^{m_\text{l}}$, the tracking error $e(t) \in \mathbb{R}^{n_\text{h}}$, and the projection of the low-level state, $\pi_{x_\ell}(x_\ell(iT_s))$ at each $t \in \mathcal{T}^c$. The low-level dynamics is assumed to be affected by a bounded disturbance, i.e., the signal $w(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n_\text{l}}$. The *low-level* feedback controller u_fb can be designed such that the error $e(t)$ belongs to a robust control invariant set $\mathcal{D} \subset \mathbb{R}^{n_\text{h}}$, see, e.g., [2, 68]. As for the *high-level* control, a new trajectory of length $T = T_s$ is generated solving an instance of problem (4.19) defined for the high-level dynamics, with initial condition $x_\text{h}(iT_s) = \pi_{x_\ell}(x_\ell(iT_s))$, thus defining a reference trajectory $(x_\text{h}(t), u_\text{h}(t))$, $t \in [iT_s, (i+1)T_s]$. The choice of the initial condition ensures $e(iT_s) = 0 \in \mathcal{D}$. To ensure *safety* for the low-level system, $\mathbb{S}$ is defined considering the maximum tracking error, i.e., the size of $\mathcal{D}$.

Multi-layer control of a unicycle robot

The low-level model is the unicycle dynamics

$$\dot{p}_{\ell,x} = v\cos\theta, \qquad \dot{p}_{\ell,y} = v\sin\theta, \qquad \dot{\theta} = \omega. \tag{4.25}$$

with $x_\ell = (p_{\ell,x}, p_{\ell,y}, \theta) \in \mathbb{R}^3$, and $u_\ell = (v, \omega) \in \mathbb{R}^2$. Being (4.25) differentially flat, a valid high-level dynamics is

$$\dot{x}_\text{h} = Ax_\text{h} + Bu_\text{h} \tag{4.26}$$

with $x_\text{h} \in \mathbb{R}^4$, $u_\text{h} \in \mathbb{R}^2$, $A = \left[\begin{smallmatrix} 0_2 & I_2 \\ 0_2 & 0_2 \end{smallmatrix}\right]$, $B = \left[\begin{smallmatrix} 0_2 \\ I_2 \end{smallmatrix}\right]$, where 0_n, I_n denote the zero and the identity $n \times n$ matrices respectively. The low level dynamics (4.25) is coupled with (4.26) via

$$\pi_{x_\ell}(x_\ell) = \begin{bmatrix} p_{\ell,x} & p_{\ell,x} & \dot{p}_{\ell,x} & \dot{p}_{\ell,x} \end{bmatrix}^\top \tag{4.27a}$$

$$\pi^{u_\text{h}}(u_\text{h}, x_\text{h}) = \begin{bmatrix} \sqrt{x_{\text{h},3}^2 + x_{\text{h},4}^2}, & \frac{-x_{\text{h},4}u_{\text{h},1} + x_{\text{h},3}u_{\text{h},2}}{x_{\text{h},3}^2 + x_{\text{h},4}^2} \end{bmatrix}^\top \tag{4.27b}$$

In [2], the following low-level feedback controller is proposed: $u_{\text{fb}} = \pi_{u_{\text{h}}}(u_e(t), x_{\text{h}}(t))$, where $u_e(t) = u_{\text{h}}(t) + \tilde{K}(x_{\text{h}}(t) - \pi_{x_\ell}(x_\ell(t)))$ with $\tilde{K}$ the gain matrix associated to a modified Riccati equation s.t. u_{fb} ensures the existence of a robustly control invariant set $\mathcal{D}$. The objective is to reach the origin optimally with respect to an instantaneous cost with $q(x_{\text{h}}) = x_{\text{h}}^\top x_{\text{h}}$. The safe set $\mathbb{S}$ of obstacle free configurations is defined as the intersection of N safety constraints $h_i(x_{\text{h}}) \geq 0$ where, for all $i = 1, \dots, N$, $h_i(x_{\text{h}}) = \|x - x_{c,i}\|^2 - r_i$ defines a circle of radius r_i centered at $x_{c,i}$ (randomly generated). The radius is tightened considering the diameter of the ellipsoid $\mathcal{D}$. Being h_i of degree 2 with respect to (4.26), we consider its extended version $h_{e,i} = \dot{h}_i(x_{\text{h}}) + \alpha(h_i(x_{\text{h}}))$. Terminal cost in (4.19) is defined as $v(x_{\text{h}}) = x_{\text{h}}^\top P x_{\text{h}}$, with P solution of the algebraic Riccati equation associated with the linear dynamics (4.26). Hence, v is the optimal value function of the unconstrained optimal control problem. The time-horizon is set as $T = T_s$. Fig. 4.2 shows the simulation results with $T_s = 2$ [s].

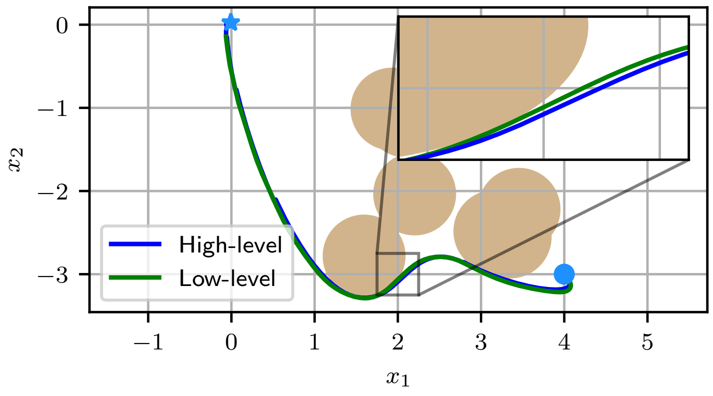


Figure 4.2: State trajectory in the 2-d plane of high- (blue) and low-level (green) positions. Initial condition (●), goal position ($\star$). Zoomed-in focus on the differences due to noise. Safety is ensured by shrinking the safe set.

Fixed safe set Firstly, we consider a fixed safe set with $N = 1$. Fig. 4.3 shows the high-level trajectory where (4.19) is solved numerically $\forall t \in \mathcal{T}^c$. Note that, the safe optimal trajectory solution of (4.9) minimally deviates to avoid the obstacle. Moreover, observe the practical ramifications of Theorem 4.2: for small κ the trajectory is suboptimal, but as κ grows the solutions of (4.19) converge to the safe optimal trajectory solution of (4.9).

Time-varying safe set: on-the-fly trajectory replanning In this section, the safe set varies through time with $N = 7$. At each $t \in \mathcal{T}^c$, problem (4.19) is solved with a safe set including only the obstacle in range, i.e., within a radius of 2 [m] from the current position. The high-level trajectory in the 2 dimensional plane is depicted in Figure 4.4. Notice that, for $T = 5$ [s], the trajectory is unsafe, namely, it results to an impact with an

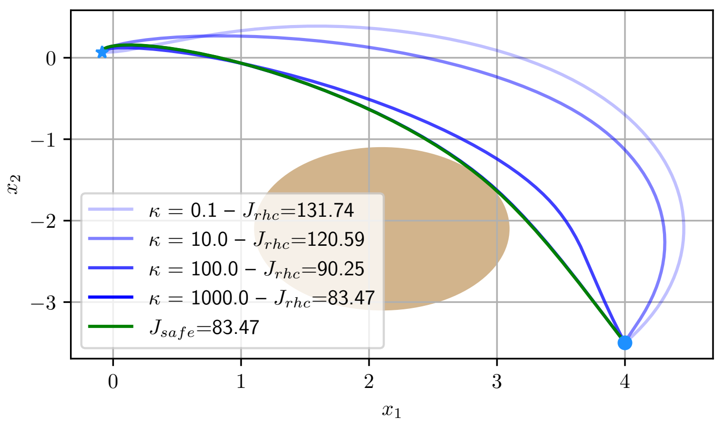


Figure 4.3: High-level state trajectory in the 2-d plane for different κ and corresponding total cost. Obstacle is fixed. $J_{\text{safe}} := J_{\text{safe}}(x, \Phi^{\text{opt}}_{\text{safe}})$, $J_{\text{rhc}} := J_{\text{rhc}}(x, \Phi^{\text{rhc}}_{\infty,\kappa})$. Of particular note, $J_{\text{rhc}} \to J_{\text{safe}}$ as κ becomes large.

obstacle not in range at planning time. $T = 2$ [s] results in the safest trajectory with the lower cost.

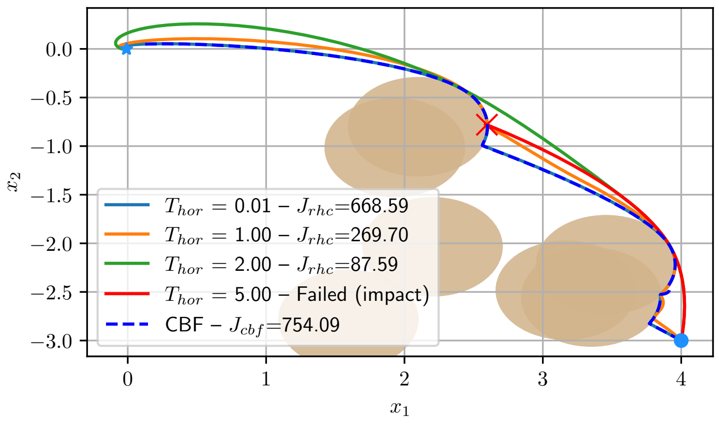


Figure 4.4: State trajectory in the 2-d plane for different prediction horizon T and corresponding total cost along the trajectory. Impacts with obstacles ($\times$). CBF = (4.12) with $\Phi^{\text{d}} = \Phi^{\text{opt}}$. Notice that, $\Phi^{\text{rhc}} \to$ (4.12) as $T \to 0$.

Conclusions

In recent decades, the control community's interest has significantly shifted towards large-scale systems, whose complexity challenges the development of efficient control policies. This thesis contributed to the field by (i) applying data-driven strategies to address the complexities of large-scale systems, (ii) leveraging inherent patterns and connections for optimal control policy design, and (iii) creating scalable algorithms for both centralized and distributed control. Our exploration began with the application of data-driven techniques to the Linear Quadratic Regulator (LQR) problem under uncertain dynamics. This included an on-policy algorithm for simultaneous system matrix identification and control policy optimization. Always within the LQR domain, we proposed numerical solutions to generate feedback controllers that comply with specific structural constraints, thus mirroring the large-scale system's architecture. Subsequently, we considered the nonlinear optimal control setting, proposing a novel first-order framework for nonlinear optimal control, termed GoPRONTO, based on a feedback embedding paradigm. This framework was then adapted to distributed control scenarios, applying an aggregative optimization approach to model collective behaviors. A learning-driven variant of GoPRONTO was also developed, combining optimization and learning. The GoPRONTO framework is then adapted for stochastic optimal control implementing a SGD scheme. Finally, we tackled the challenge of formulating inherently safe control policies for multi-layer systems.

Future research directions involve the combination of the developed schemes and methodologies within a nonlinear data-driven setting, leveraging on the algorithmic strategies established in the first part of this thesis for designing the tracking control policy essential to the feedback embedding paradigm. Additionally, extending the proposed strategies for designing structured control policies to the on-policy framework presents another challenging direction, thus facilitation the application of our approaches in real-world scenarios, where data must be collected while the system is governed by a feasible sparse controller. Another intriguing area of research involves extending the tools examined and developed in the final chapter of this thesis, i.e., safe controllers and multi-layer architectures, to a learning-driven context. Finally, the implementation of the proposed solutions in real-world robotics applications is an active area of work.

Appendix A

Basics on Nonlinear Optimization

A.1 First-order methods for Nonlinear Programming

For the sake of completeness, we briefly recall here the alterative, gradient-based methods to solve unconstrained optimization problems in the form

$$\min_{x \in \mathbb{R}^d} \; g(x) \tag{A.1}$$

where $g : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable.

The plain gradient descent update reads is an iterative procedure in which a tentative estimate x^k of a stationary point of (A.1) is updated for all $k > 0$ as

$$x^{k+1} = x^k - \gamma^k \nabla g(x^k)$$

where $\gamma^k \in \mathbb{R}$ is the step-size.

A.1.1 Conjugate Gradient Method

The Conjugate Gradient (CG) iteration applied to problem (A.1) reads

$$x^{k+1} = x^k + \gamma^k d^k$$

where $\gamma^k \in \mathbb{R}$ is the step-size obtained by line search and $d^k \in \mathbb{R}^n$ is computed at $k = 0$ as $d^0 = -\nabla f(x^0)$, while for $k = 1, 2, \dots$ as

$$d^k = -\nabla f(x^k) + \rho^k d^{k-1}. \tag{A.2}$$

The CG update parameter $\rho^k \in \mathbb{R}$ can be chosen adopting alterative methods. A common way [180] to compute ρ^k is

$$\rho^k = \frac{\nabla f(x^k)^\top (\nabla f(x^k) - \nabla f(x^{k-1}))}{\nabla f(x^{k-1})^\top \nabla f(x^{k-1}))}.$$

A.1.2 Heavy-ball Method

The Heavy-ball method introduces a so-called momentum term with step-size $\gamma_{\text{hb}}^k \in \mathbb{R}$ to the plain gradient step. Formally, it reads for all $k > 0$ as

$$x^{k+1} = x^k - \gamma^k \nabla g(x^k) + \gamma_{\text{hb}}^k (x^k - x^{k-1})$$

The term $x^k - x^{k-1}$ nudges x^{k+1} in the direction of the previous step, hence the name momentum.

A.1.3 Nesterov's Accelerated Gradient Method

The Nesterov's accelerated gradient iteration applied to problem (A.1) reads

$$\tilde{x}^k = x^k + \tfrac{k}{k+3}(x^k - x^{k-1}) \tag{A.3a}$$

$$x^{k+1} = \tilde{x}^k - \gamma^k \nabla g(\tilde{x}^k) \tag{A.3b}$$

where $\gamma^k \in \mathbb{R}$ is the step-size. It differs from the plain gradient since it perturb the point at which the gradient step is computed by exploiting information from past iterates.

A.2 Inexact Augmented Lagrangian Method for Constrained Minimization

Consider the constrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x) \tag{A.4a}$$

$$\text{subj.to} \;\; h(x) = 0 \tag{A.4b}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ cost function and $h : \mathbb{R}^n \to R^m$ constraint function. Assume $f \in \mathcal{C}^1$ and $h \in \mathcal{C}^1$. The augmented Lagrangian $\mathcal{L}_c : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}$ associated to problem (A.4) is

$$\mathcal{L}_c(x, \lambda) = f(x) + \lambda^\top h(x) + \frac{c}{2}\|h(x)\|^2 \tag{A.5}$$

where $c > 0$ is a penalty parameter and $\lambda \in \mathbb{R}^m$ is the Lagrange multiplier. In the augmented Lagrangian algorithm, at each iteration $\ell \in \mathbb{N}$, the inner minimization problem

$$\min_{x \in \mathbb{R}^n} \ \mathcal{L}_{c^\ell}(x, \lambda^\ell) \tag{A.6}$$

is terminated a point x^ℓ such that

$$\|\nabla_x \mathcal{L}_{c^\ell}(x^\ell, \lambda^\ell)\| \leq \epsilon^\ell \tag{A.7}$$

where with $\mathcal{L}_{c^\ell}(x^\ell, \lambda^\ell)$ we denote the augmented Lagrangian evaluated at $x^\ell, \lambda^\ell, c^\ell$. The following holds true.

Theorem A.1 ([30, Proposition 5.2.2]). *Assume f and h are continuously differentiable. For $\ell = 0, 1, \ldots$ let the following hold*

(i) x^ℓ satisfies, for all ℓ,

$$\|\nabla_x \mathcal{L}_{c^\ell}(x^\ell, \lambda^\ell)\| \leq \epsilon^\ell; \tag{A.8}$$

(ii) the sequence $\{\lambda^\ell\}_{\ell \geq 0}$ is bounded;

(iii) the sequence $\{c^\ell\}_{\ell \geq 0}$ is such that $c^\ell \to \infty$, and, for all ℓ,

$$0 \leq c^\ell \leq c^{\ell+1}; \tag{A.9}$$

(iv) the sequence $\{\epsilon^\ell\}_{\ell \geq 0}$ is such that $\epsilon^\ell \to 0$ and $\epsilon \geq 0$ for all ℓ.

For any subsequence $\{x^\ell\}_{\ell \in \mathcal{C}}$ that converges to a point $\bar{x}$ such that $\nabla h(\bar{x})$ is full-row rank, the subsequence

$$\{\lambda^\ell c^\ell h(x^\ell)\}_{\ell \in \mathcal{C}} \to \bar{\lambda} \tag{A.10}$$

where the pair $(\bar{x}, \bar{\lambda})$ satisfies the first order necessary conditions for optimality associated to problem (A.4). $\triangle$

A.3 Unconstrained Stochastic Optimization

Consider the minimization of the expected value

$$\min_{x \in \mathbb{R}^n} \ \mathbb{E}_{\mathrm{w}} \left[\ell(x, \mathrm{w})\right] = \min_{x \in \mathbb{R}^n} \ \bar{\ell}(x) \tag{A.11}$$

where $\ell : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ is a function of decision variable $x \in \mathbb{R}^n$ and a random variable w induced by a certain probability space. We refer as $\bar{\ell}(x)$ the expected value of the function of random variable $\ell(x, \mathrm{w})$. In many cases, the calculation of the expected value in (A.11) is not of practical use, e.g., the probability density function p_{w} may be not known. To overcome this issue, numerous resolution strategies were proposed in the literature.

Empirical Expectation Approximation The expected value in (A.11) can be approximated by the so-called empirical expectation, i.e., considering M realizations $w_1, \dots, w_M$ of the random variable, the expected value $\mathbb{E}_{\mathrm{w}}[\cdot]$ is approximated as

$$\mathbb{E}_{\mathrm{w}}\left[\ell(x, \mathrm{w})\right] \approx \frac{1}{M} \sum_{i=1}^{M} \ell(x, w_i). \tag{A.12}$$

The approximation (A.12) leads to a numerically tractable approximation of problem (A.11), which can then be solved by means of, e.g., a gradient descent method.

Stochastic Gradient Descent Stochastic Gradient Descent (SGD) represents an iterative method to solve problem (A.11) without recurring to numerical approximation strategies. Indeed, SGD takes the random variable w as a seed for generating a descent direction. Specifically, at each iteration of the algorithm, the gradient of $\bar{\ell}(x)$, i.e., the descent direction, is computed by means of a gradient estimator $G(x, \mathrm{w})$ built over some observations of w. The update law at iteration $k > 0$ reads

$$x^{k+1} = x^k - \gamma^k G(x^k, w^k) \tag{A.13}$$

where w^k represent a realization of w at iteration k and $\gamma^k \in \mathbb{R} \geq 0$ is the so-called stepsize chosen at iteration k. For the classic SGD algorithm convergence the following result holds. See [41] for a detailed discussion.

Theorem A.2. *Let the following assumptions hold:*

1. *The objective function $\bar{\ell}$ is bounded from below, i.e., there exists a $\ell_{inf} \in \mathbb{R}$ such that $\bar{\ell}(x) \geq \bar{\ell}_{inf}$ for all $x \in \mathbb{R}^n$.*

2. *The objective function $\bar{\ell}$ is continuously differentiable and the gradient of $\bar{\ell}$, namely, $\nabla\bar{\ell}(x) : \mathbb{R}^n \to \mathbb{R}^n$, is Lipschitz continuous with Lipschitz constant L, i.e.,*

$$\|\nabla\bar{\ell}(x) - \nabla\bar{\ell}(x')\| \leq L\|x - x'\| \qquad \forall x, x' \in \mathbb{R}^n. \tag{A.14}$$

3. *The stochastic direction $G(x, \mathrm{w})$ is an unbiased estimator of $\nabla\bar{\ell}(x)$, i.e.,*

$$\mathbb{E}_{\mathrm{w}}\big[G(x, \mathrm{w})\big] = \nabla\bar{\ell}(x). \tag{A.15}$$

4. *The second moment of the random variable $\|G(x, \mathrm{w}) - \nabla\bar{\ell}(x, \mathrm{w})\|$ is bounded, i.e.,*

$$\mathbb{E}_{\mathbf{w}}[\|G(x, \mathrm{w}) - \nabla\bar{\ell}(x, \mathrm{w})\|^2] \leq M^2 \tag{A.16}$$

   for all x and with $M > 0$.

Then, for the sequence $\{x^k\}_{k>0}$ generated via iteration (A.13) *with stepsize sequence $\{\gamma^k\}_{k>0}$ selected such that*

$$\sum_{k=0}^{\infty} \gamma^k = \infty \qquad \sum_{k=0}^{\infty} (\gamma^k)^2 \leq \infty \tag{A.17}$$

it holds true

$$\lim_{N\to\infty} \mathbb{E}\left[\frac{1}{\Gamma^N} \sum_{k=1}^{N} \gamma^k \|\nabla\bar{\ell}(x^k)\|\right] = 0 \tag{A.18}$$

where $\Gamma^N := \sum_{k=1}^{N} \gamma^k$. △

Appendix B

First- and Second-order Numerical Methods for Optimal Control

B.1 Linear Quadratic Regulator

Next, we recall the key ingredients for devising a model-based gradient method to address problem

$$\min_{\substack{x_1,x_2,\ldots,\\ \mathrm{u}_0,\mathrm{u}_1,\ldots}} \ \tfrac{1}{2}\,\mathbb{E}\Big[\sum_{t=0}^{\infty}\Big(x_t{}^\top Q x_t + u_t{}^\top R u_t\Big)\Big] \tag{B.1a}$$

$$\text{subj.to } \ x_{t+1} = A_\star x_t + B_\star u_t, \quad x_0 \sim p_{\mathrm{x}_0} \tag{B.1b}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the state and the input at time $t \in \mathbb{N}$, while $A_\star \in \mathbb{R}^{n\times n}$ and $B_\star \in \mathbb{R}^{n\times m}$ are the state and input matrices, respectively. The operator $\mathbb{E}[\cdot]$ denotes the expectation taken with respect to the probability distribution p_{x_0}. The initial condition x_0 is assumed to be randomly distributed according to a known uniform probability distribution p_{x_0}.

First of all, we recall an equivalent (unconstrained) formulation of Problem (B.1) that explicitly imposes the linear feedback structure to the optimal input and is amenable for gradient-based algorithmic solutions. Letting $K \in \mathbb{R}^{m\times n}$, Problem (B.1) is rewritten by substituting in the dynamics and in the cost function the input in linear feedback form

$$u_t = K x_t.$$

where K is to be computed. Such a formulation clearly highlights that (i) the overall problem actually depends on the gain K only, and, (ii) the optimal gain $K^\star$ does not depend on the initial condition x_0. First of all, given any gain K, the original (open-loop) dynamics (B.1b) admits the closed-loop formulation $x_{t+1} = (A + BK)x_t$. So that, for all

$t \geq 0$, the state is uniquely determined as

$$x_t = (A + BK)^t x_0, \qquad x_0 \sim p_{\mathrm{x}_0}. \tag{B.2}$$

Hence, for all initial conditions x_0, Problem (B.1) can be compactly written as

$$\min_K \ \tfrac{1}{2}\,\mathbb{E}\Big[x_0^\top \Big(\sum_{t=0}^{\infty}(A+BK)^{t\top}(Q + K^\top RK)(A+BK)^t\Big)x_0.\Big]$$

By averaging on the initial condition, we obtain

$$\min_K \ \tfrac{1}{2}\,\mathrm{Tr}\Big(\sum_{t=0}^{\infty}(A+BK)^{t\top}(Q + K^\top RK)(A+BK)^t\Sigma_0\Big)$$

where $\Sigma_0 := \mathbb{E}[x_0 x_0^\top]$. Without loss of generality, we consider p_{x_0} to be a uniform distribution about the unit sphere. Therefore, we can finally write

$$\min_{K \in \mathcal{D}} \ J(K), \tag{B.3}$$

where the cost function $J : \mathcal{D} \to \mathbb{R}$ is given by

$$J(K) := \tfrac{1}{2}\,\mathrm{Tr}\sum_{t=0}^{\infty}(A+BK)^{t,\top}(Q + K^\top RK)(A+BK)^t, \tag{B.4}$$

and the set $\mathcal{D} \subset \mathbb{R}^{m\times n} := \{K \in \mathbb{R}^{m\times n} \mid J(K) < \infty\}$ is the domain of J, i.e., the set over which J is well-defined.

Remark B.1. It is possible to show that the set of stabilizing gains $\mathcal{K} := \{K \in \mathbb{R}^{m\times n} \mid A + BK \text{ is Schur}\} \subseteq \mathbb{R}^{m\times n}$ coincides with the interior of $\mathcal{D}$, see [48, Lemma 3.2]. Therefore, it must be that the optimal gain $K_\star$ belongs to $\mathcal{D}$. $\triangle$

B.1.1 Gradient of the reduced cost function J

As for the gradient $\nabla J(K)$ of the reduced cost function J the following holds.

Lemma B.1. *The gradient of the reduced cost function $J(K)$ evaluated at $K \in \mathbb{R}^{m\times n}$ can be operatively computed as*

$$\nabla J(K) = \Big(RK + B_\star^\top P A_{\mathrm{cl}}\Big)W \tag{B.5}$$

where $P \in \mathbb{R}^{n\times n}$ and $W \in \mathbb{R}^{n\times n}$ are the solutions of

$$(A+BK)\,W(A+BK)^\top - W = -I \tag{B.6a}$$

$$(A+BK)^\top P(A+BK) - P = -(Q + K^\top RK) \tag{B.6b}$$

which, for a stabilizing K, admit unique symmetric positive-definite solutions. △

Proof. Consider a perturbation $\Delta K \in \mathbb{R}^{m\times n}$. The first order approximation of J, with and $\epsilon > 0 \in \mathbb{R}$, reads as

$$J(K+\epsilon\Delta K) = J(K) + \epsilon DJ(K)\cdot\Delta K + o(\epsilon|\Delta K|) \tag{B.7}$$

Let us write

$$\begin{aligned}
J(K+\epsilon\Delta K) &= \tfrac{1}{2}\operatorname{Tr}\sum_{t=0}^{\infty}\Big((A+BK+\epsilon B\Delta K)^{t\top} \\
&\qquad\cdot(Q+(K+\epsilon\Delta K)^\top R(K+\epsilon\Delta K))(A+BK+\epsilon B\Delta K)^{t}\Big) \\
&= \tfrac{1}{2}\operatorname{Tr}\sum_{t=0}^{\infty}\Big(\Big((A+BK)^t+\epsilon\sum_{\tau=0}^{t-1}(A+BK)^{t-1-\tau}B\Delta K(A+BK)^{\tau}\Big)^\top \\
&\qquad\cdot(Q+(K+\epsilon\Delta K)^\top R(K+\epsilon\Delta K)) \\
&\qquad\quad\cdot\Big((A+BK)^t+\epsilon\sum_{\tau=0}^{t-1}(A+BK)^{t-1-\tau}B\Delta K(A+BK)^{\tau}\Big)\Big) \\
&\qquad\qquad + o(\epsilon|\Delta K|)
\end{aligned}$$

We start by observing that the following holds

$$Q+(K+\epsilon\Delta K)^\top R(K+\epsilon\Delta K) = (Q+K^\top RK)+\epsilon K^\top R\Delta K+\epsilon\Delta K^\top RK + o(\epsilon|\Delta K|)$$

Computing the products and highlighting the first order terms only, we have

$$\begin{aligned}
J(K+\epsilon\Delta K) = \tfrac{1}{2}\operatorname{Tr}\sum_{t=0}^{\infty}\Big(&(A+BK)^{t,\top}(Q+K^\top RK)(A+BK)^t \\
&+2\epsilon(A+BK)^{t,\top}(K^\top R\Delta K)(A+BK)^t \\
&+\epsilon\Big(\sum_{\tau=0}^{t-1}(A+BK)^{t-1-\tau}B\Delta K(A+BK)^{\tau}\Big)^\top(Q+K^\top RK)(A+BK)^t \\
&+\epsilon(A+BK)^{t,\top}(Q+K^\top RK)\Big(\sum_{\tau=0}^{t-1}(A+BK)^{t-1-\tau}B\Delta K(A+BK)^{\tau}\Big)\Big) \\
&\quad+o(\epsilon|\Delta K|)
\end{aligned}$$

Notice that $\operatorname{Tr}(M)=\operatorname{Tr}(M^\top)$, hence we can write

$$J(K+\epsilon\Delta K)-J(K) = \epsilon\tfrac{1}{2}\operatorname{Tr}\sum_{t=0}^{\infty}\Big(2(A+BK)^t(A+BK)^{t,\top}K^\top R\Delta K$$

$$
\begin{aligned}
&+ 2\sum_{\tau=0}^{t-1}(A+BK)^{\tau}(A+BK)^{t,\top}(Q+K^\top RK) \\
&\quad \cdot (A+BK)^{t-1-\tau}B\Delta K\Big) \\
&\quad + o(\epsilon|\Delta K|) \\
&= \epsilon \operatorname{Tr}\Big(M(K)\Delta K\Big) + o(\epsilon|\Delta K|)
\end{aligned}
$$

with the matrix $M(K) \in \mathbb{R}^{n\times m}$ defined as

$$
\begin{aligned}
M(K) := \sum_{t=0}^{\infty}\Big(&(A+BK)^t(A+BK)^{t,\top}K^\top R \\
&+ \sum_{\tau=0}^{t-1}(A+BK)^{\tau}(A+BK)^{t,\top}(Q+K^\top RK)(A+BK)^{t-1-\tau}B\Big)
\end{aligned}
$$

Invoking the Kleinman's lemma (see Section B.1.5), we can write

$$
\nabla J(K) = M(K)^\top
$$

having the same dimension of the matrix $K \in \mathbb{R}^{m\times n}$.

Remark B.2. We can write a representation of the Frèchet derivative as the inner product of $\nabla J(K)$ and ΔK as

$$
DJ(K)\cdot\Delta K = \langle \nabla J(K), \Delta K\rangle = \operatorname{Tr}\Big(\nabla J(K)^\top \Delta K\Big) \qquad \triangle
$$

We are now ready to provide an explicit calculation of $\nabla J(K)$. Specifically, it holds

$$
\begin{aligned}
\nabla J(K) = &\underbrace{\sum_{t=0}^{\infty}\Big(RK(A+BK)^t(A+BK)^{t,\top}\Big)}_{(a)} \\
&+ \underbrace{\sum_{t=0}^{\infty}\sum_{\tau=0}^{t-1}\Big(B^\top(A+BK)^{t-1-\tau,\top}(Q+K^\top RK)(A+BK)^t(A+BK)^{\tau,\top}\Big)}_{(b)}
\end{aligned}
$$

The term (a) is equivalent to

$$
\sum_{t=0}^{\infty}\Big(RK(A+BK)^t(A+BK)^{t,\top}\Big) = RK\underbrace{\sum_{t=0}^{\infty}(A+BK)^t(A+BK)^{t,\top}}_{=\text{(B.6a) (if } K \text{ stabilizing)}}
$$

As for the term (b), by exchanging the summations it can be written as

$$\sum_{\tau=0}^{\infty}\sum_{t=\tau+1}^{\infty}\left(B^\top(A+BK)^{t-1-\tau,\top}(Q+K^\top RK)(A+BK)^t(A+BK)^{\tau,\top}\right)$$

and changing summation variable $t \mapsto \xi = t-\tau-1$ we have

$$\begin{aligned}
&\sum_{\tau=0}^{\infty}\sum_{t=\tau+1}^{\infty}\left(B^\top(A+BK)^{t-1-\tau,\top}(Q+K^\top RK)(A+BK)^t(A+BK)^{\tau,\top}\right)\\
&=\sum_{\tau=0}^{\infty}\sum_{\xi=0}^{\infty}\left(B^\top(A+BK)^{\xi,\top}(Q+K^\top RK)(A+BK)^{\xi+\tau+1}(A+BK)^{\tau,\top}\right)\\
&=B^\top\sum_{\xi=0}^{\infty}(A+BK)^{\xi,\top}(Q+K^\top RK)(A+BK)^{\xi+1}\underbrace{\sum_{\tau=0}^{\infty}(A+BK)^{\tau}(A+BK)^{\tau,\top}}_{=\text{(B.6a) (if } K \text{ stabilizing)}}\\
&=B^\top\underbrace{\sum_{\xi=0}^{\infty}(A+BK)^{\xi,\top}(Q+K^\top RK)(A+BK)^{\xi}}_{=\text{(B.6b) (if } K \text{ stabilizing)}}(A+BK)\underbrace{\sum_{\tau=0}^{\infty}(A+BK)^{\tau}(A+BK)^{\tau,\top}}_{=\text{(B.6a) (if } K \text{ stabilizing)}}
\end{aligned}$$

where we set $\xi = t-\tau-1$ (and, hence, $t = \xi+\tau+1$). The highlighted terms, for K stabilizing, can be computed as solutions of the Lyapunov equations in (B.6)[1]. More in detail, definying $P \in \mathbb{R}^{n\times n}$ and $W \in \mathbb{R}^{n\times n}$ as the solutions of the lyapunov equations

$$\begin{aligned}
&(A+BK)\,W(A+BK)^\top - W = -I,\\
&(A+BK)^\top P(A+BK) - P = -(Q+K^\top RK),
\end{aligned}$$

we can write

$$\nabla J(K) = RKW + B^\top P(A+BK)W.$$

The proof follows. ∎

B.1.2 First-order Necessary Conditions for Optimality of Problem (B.3)

Formally, a given $K^\star$ satisfies the FNC of problem (B.3) if $\nabla J(K^\star) = 0$, that is

$$RK^\star W_c(K^\star) + B^\top P(K^\star)(A+BK^\star)W_c(K^\star) = 0 \tag{B.15a}$$

[1]Consider the autonomous system $x_{t+1} = Ax_t$ and the Lyapunov equation $A^\top XA - X + Q = 0$. If A is stable, X can be computed explicitly as $X = \sum_{t=0}^{\infty} A^{t,\top}QA^t$.

with $W_c(K^\star)$ and $P(K^\star)$ obtained, respectively, as solutions of

$$(A + BK^\star)W_c(A + BK^\star)^\top - W_c = -I \tag{B.15b}$$

$$(A + BK^\star)^\top P(A + BK^\star) - P = -(Q + K^{\star\top} RK^\star) \tag{B.15c}$$

B.1.3 Gradient method for LQR

Being the set of stabilizing gains $\mathcal{K}$ open [49, Lemma IV.3] and connected [49, Lemma IV.6], the gradient descent method could be used to solve Problem (B.3) (see, e.g., [48]). Namely, at each iteration $k \in \mathbb{N}$, an estimate K_k of $K_\star$ is maintained and iteratively updated according to

$$K_{k+1} = K_k - \gamma \nabla J(K_k), \tag{B.16}$$

where $\gamma > 0$ is the stepsize, while $\nabla J : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is the gradient of J with respect to K evaluated at K_k, when $\mathbb{R}^{m\times n}$ is equipped with the Frobenius inner product. It is possible to show that, by initializing $K_0 \in \mathcal{K}$ and selecting a proper stepsize γ, the optimal gain $K_\star$ is an exponentially stable equilibrium of the dynamical system (B.16), see [48, Theorem 4.6].

B.1.4 Anderson-Moore Algorithm

The Anderson-Moore Algorithm, cf [8] is is a fixed point iteration applied to the first order necessary conditions (B.15) of problem (B.3) Given a stabilizing K^0, the Anderson-Moore algorithm reads for all iterations $k \geq 0$

Algorithm 14 Anderson-Moore Algorithm

```
Require: Stabilizing gain $K^0$
  for $k = 0, 1, 2 \dots$ do
      Compute $W^k$ solution of
          $(A + BK^k)W(A + BK^k)^\top - W = -I$
      Compute $P^k$ solution of
          $(A + BK^k)^\top P(A + BK^k) - P = -(Q + K^{k\top} RK^k)$
      Find $K^{k+1}$ such that $RKW^k + B^\top P^k(A + BK)W^k = 0$, i.e., set
          $K^{k+1} = -(R + B^\top P^k B)^{-1} B^\top P^k A$
```

Remark B.3. One can also introduce a globalization enhancement of the method by replacing the last step with

$$K^{k+1} = K^k - \gamma^k \big(K^k + (R + B^\top P^k B)^{-1} B^\top P^k A \big) \qquad \triangle$$

B.1.5 Kleinman Lemma

Definition B.1 (Trace function). *A scalar function $f : \mathbb{R}^{m \times p} \to \mathbb{R}$ is called a* trace function *of the matrix $L \in \mathbb{R}^{m \times p}$ if it is defined as the following composition*

$$f(L) = \operatorname{Tr}\big(g(L)\big) \tag{B.17}$$

where $g : \mathbb{R}^{m \times p} \to \mathbb{R}^{r \times r}$ is continuously differentiable. $\triangle$

The following lemma holds.

Lemma B.2 (Kleinman). *Let $f(K)$ be a trace function of $K \in \mathbb{R}^{m \times p}$. If one can write*

$$f(K + \epsilon \Delta K) - f(K) = \epsilon \operatorname{Tr}(M(K) \Delta K) + \epsilon o(|\Delta K|) \tag{B.18}$$

where $M(K) \in \mathbb{R}^{p \times m}$, then

$$\frac{\mathrm{d} f(K)}{\mathrm{d} K} = \nabla f(K) = M(K)^\top \tag{B.19}$$

with $\nabla f(K) \in \mathbb{R}^{m \times p}$. $\triangle$

B.2 Nonlinear Optimal Control

In the following, we review two numerical methods for the resolution of nonlienare optimal control problems in the form

$$\min_{\boldsymbol{x} \in \mathbb{R}^{nT}, \boldsymbol{u} \in \mathbb{R}^{mT}} \sum_{t=0}^{T-1} \ell_t(x_t, u_t) + \ell_T(x_T) \tag{B.20a}$$

$$\text{subj.to } x_{t+1} = f(x_t, u_t), \quad t \in [0, T-1] \tag{B.20b}$$

with initial condition $x_0 = x_{\text{init}} \in \mathbb{R}^n$, stage cost $\ell_t : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and terminal cost $\ell_T : \mathbb{R}^n \to \mathbb{R}$.

B.2.1 Gradient Method for Optimal Control

In this subsection we recall a numerical strategy proposed, e.g., in [27, Section 1.9] to solve a discrete-time optimal control problem as in (B.20) based on the gradient method.

The leading idea is to express the state x_t at each $t \in [0, T-1]$ as a function of the input sequence $\boldsymbol{u}$ only. Formally, for all t we can introduce a map $\phi_t : \mathbb{R}^{mT} \to \mathbb{R}^n$ such that

$$x_t := \phi_t(\boldsymbol{u}), \tag{B.21}$$

so that problem (B.20) can be recast into the reduced version

$$\min_{\boldsymbol{u}} \sum_{t=0}^{T-1} \ell_t(\phi_t(\boldsymbol{u}), u_t) + \ell_T(\phi_T(\boldsymbol{u})) = \min_{\boldsymbol{u}} J(\boldsymbol{u}) \tag{B.22}$$

where the optimization variable is only the input sequence $\boldsymbol{u} \in \mathbb{R}^{mT}$. Problem (B.22) is an unconstrained optimization problem in $\boldsymbol{u}$ with a $\mathcal{C}^2$ cost function. Notice that the cost function $J(\boldsymbol{u})$ inherits from (B.20) its smoothness properties, but also its nonconvexity. Hence, problem (B.22) can be addressed via a gradient descent method in which a tentative solution $\boldsymbol{u}^k \in \mathbb{R}^{mT}$ is iteratively updated as

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k - \gamma^k \nabla J(\boldsymbol{u}^k), \tag{B.23}$$

where $k > 0$ denotes the iteration counter, while the parameter $\gamma^k > 0$ is the so-called step-size.

Denoting $\Delta u_t^k = -\nabla_{u_t} J(\boldsymbol{u}^k)$, the previous update can be also written in a component-wise fashion for $t \in [0, T-1]$ as

$$u_t^{k+1} = u_t^k + \gamma^k \Delta u_t^k. \tag{B.24}$$

The gradient of $J(\cdot)$ at every $\boldsymbol{u}^k$ can be efficiently computed by properly exploiting a costate difference equation (to be simulated backward in time) based on the linearization of the cost and the system dynamics at a given trajectory $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ according to (3.11). Algorithm 15 summarizes the overall procedure, with system state initialized at $x_0^k = x_{\text{init}}$ for all k.

Algorithm 15 Gradient Method for Optimal Control

```
for $k = 0, 1, 2 \ldots$ do
    set $\lambda_T^k = \nabla \ell_T(x_T^k)$
    for $t = T-1, \ldots, 0$ do
        Step 1: compute descent direction
            $\lambda_t^k = A_t^{k\top} \lambda_{t+1}^k + a_t^k$    (B.25a)
            $\Delta u_t^k = -B_t^{k\top} \lambda_{t+1}^k - b_t^k$    (B.25b)
    for $t = 0, \ldots, T-1$ do
        Step 2: compute new input sequence
            $u_t^{k+1} = u_t^k + \gamma^k \Delta u_t^k$
        Step 3: (open-loop) update new (feasible) trajectory
            $x_{t+1}^{k+1} = f(x_t^{k+1}, u_t^{k+1})$    (B.26)
```

As mentioned above, the update of the costate $\boldsymbol{\lambda}^k = \mathrm{col}(\lambda_1^k, \ldots, \lambda_T^k)$ involves the linearization of both the cost and the dynamics at the current input estimate $\boldsymbol{u}^k$ and corresponding state $\boldsymbol{x}^k$ (cf. (3.11)). Then, the component $\Delta u_t^k \in \mathbb{R}^m$ of the update (descent) direction in (B.24) is obtained via (B.25). Thus, the algorithm makes explicit use of the state sequence $\boldsymbol{x}^k$ (associated to the current input estimate $\boldsymbol{u}^k$), which is obtained by forward simulation of the dynamics (B.20b) over the horizon $[0, T-1]$ so that $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ is a trajectory (cf. (B.26)).

Remark B.4. We stress that, as it results from (B.26), each state trajectory $\boldsymbol{x}^{k+1}$ is generated by an open-loop simulation of the dynamics, so that the method is not practically implementable for systems exhibiting unstable behaviors. △

Since Algorithm 15 generates a sequence of inputs $\{\boldsymbol{u}^k\}_{k\geq 0}$ associated to a gradient method applied to (B.22), it inherits its convergence results. Notice that the presented backward-forward sweep algorithm could be seen as the reverse mode of the so-called Algorithmic Differentiation, see [104] for details.

B.2.2 Discrete-time PRONTO

In this section, along the lines of [17], we present a discrete-time version of the optimal control algorithm PRONTO proposed in [113] in a continuous-time framework.

The key idea of PRONTO is to use a stabilizing feedback in an optimal control method to gain numerical stability, and to interpret such (tracking) controller as a projection operator that maps (state-input) curves into system trajectories. Given a state-input curve $(\boldsymbol{\alpha}, \boldsymbol{\mu})$, let us formally consider a nonlinear tracking system given by

$$\begin{aligned} u_t &= \mu_t + K_t(\alpha_t - x_t), \\ x_{t+1} &= f(x_t, u_t), \end{aligned} \tag{B.27}$$

where $K_t \in \mathbb{R}^{n \times m}$ is a properly selected gain matrix. Thanks to the feedback policy (B.27), the optimal control problem (B.20) can be written as

$$\min_{\boldsymbol{\alpha},\boldsymbol{\mu}} \ \ell(\phi(\boldsymbol{\alpha}, \boldsymbol{\mu}), \psi(\boldsymbol{\alpha}, \boldsymbol{\mu})). \tag{B.28}$$

Figure B.1 also provides a graphical interpretation of PRONTO. At each iteration of the algorithm an update direction (in blue) is sought onto the tangent space to the trajectory manifold (in green). Then, the updated (infeasible) curve is projected back onto the trajectory manifold (in black) by the projection operator. Specifically, the PRONTO algorithm iteratively refines, for all $k > 0$, a tentative solution of problem (B.28) according to the update

$$\begin{bmatrix} \boldsymbol{x}^{k+1} \\ \boldsymbol{u}^{k+1} \end{bmatrix} = \mathcal{P}\Bigg(\underbrace{\begin{bmatrix} \boldsymbol{x}^{k} \\ \boldsymbol{u}^{k} \end{bmatrix} + \gamma^k \begin{bmatrix} \boldsymbol{\Delta}\mathbf{x}^{k} \\ \boldsymbol{\Delta}\mathbf{u}^{k} \end{bmatrix}}_{(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})} \Bigg), \tag{B.29}$$

where $\gamma^k \in (0, 1]$ is the step-size, while the update direction $(\boldsymbol{\Delta}\mathbf{x}^k, \boldsymbol{\Delta}\mathbf{u}^k) \in \mathbb{R}^{nT} \times \mathbb{R}^{mT}$ is obtained by minimizing a quadratic approximation of the cost in (B.28) over the tangent space $T^k_{(\boldsymbol{x},\boldsymbol{u})}\mathcal{T}$ at the current trajectory $(\boldsymbol{x}^k, \boldsymbol{u}^k)$.

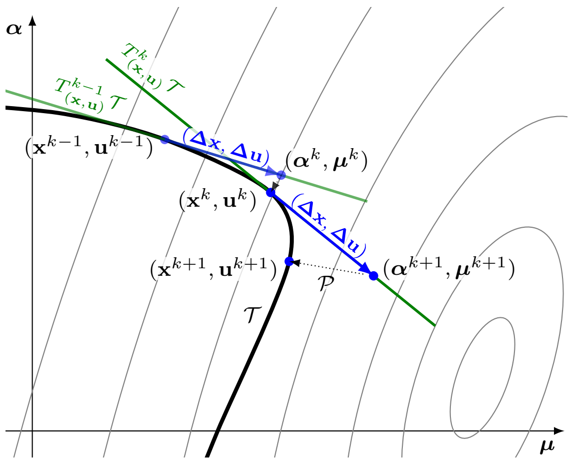
The update direction $(\boldsymbol{\Delta}\mathbf{x}^k, \boldsymbol{\Delta}\mathbf{u}^k)$ is obtained as the minimizer of the following problem

$$\min_{(\boldsymbol{\Delta}\mathbf{x}, \boldsymbol{\Delta}\mathbf{u}) \in T^k_{(\boldsymbol{x},\boldsymbol{u})}\mathcal{T}} \nabla \ell(\boldsymbol{x}^k, \boldsymbol{u}^k)^\top \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix}^\top W(\boldsymbol{x}^k, \boldsymbol{u}^k) \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix},$$

where $W(\boldsymbol{x}^k, \boldsymbol{u}^k)$ is a square matrix. In the pure Newton version of PRONTO, $W(\boldsymbol{x}^k, \boldsymbol{u}^k)$ is the second order derivative of the reduced problem (B.28), including also second order derivatives of the projection operator, i.e.,

$$W(\boldsymbol{x}^k, \boldsymbol{u}^k) := \nabla^2 \ell(\boldsymbol{x}^k, \boldsymbol{u}^k) + \nabla^2 \mathcal{P}(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k) \nabla \ell(\boldsymbol{x}^k, \boldsymbol{u}^k)$$

We refer to [113] for a detailed discussion.



$\boldsymbol{\alpha}$
$\boldsymbol{\mu}$
$T^k_{(\mathbf{x},\mathbf{u})}\mathcal{T}$
$T^{k-1}_{(\mathbf{x},\mathbf{u})}\mathcal{T}$
$(\mathbf{x}^{k-1}, \mathbf{u}^{k-1})$
$(\boldsymbol{\Delta}\mathbf{x}, \boldsymbol{\Delta}\mathbf{u})$
$(\boldsymbol{\alpha}^k, \boldsymbol{\mu}^k)$
$(\mathbf{x}^k, \mathbf{u}^k)$
$(\boldsymbol{\Delta}\mathbf{x}, \boldsymbol{\Delta}\mathbf{u})$
$(\mathbf{x}^{k+1}, \mathbf{u}^{k+1})$
$\mathcal{P}$
$(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})$
$\mathcal{T}$

Figure B.1: Representation of PRONTO approach: in gray the level curves of the cost function $\ell(\cdot,\cdot)$, in black the trajectory manifold $\mathcal{T}$, in green its tangent space at various trajectories. At each iteration k, the update direction $(\boldsymbol{\Delta}\mathbf{x}, \boldsymbol{\Delta}\mathbf{u})$ in blue is sought on the tangent space at the current trajectory $(\boldsymbol{x}^k, \boldsymbol{u}^k)$. The updated curve $(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\mu}^{k+1})$ is then projected onto $\mathcal{T}$ by the projection operator $\mathcal{P}$ (dotted line).

Remark B.5. Depending on the choice of $W(\boldsymbol{x}^k, \boldsymbol{u}^k)$ some lower-order versions of PRONTO are possible, e.g., setting $W(\boldsymbol{x}^k, \boldsymbol{u}^k) = I$, with I being the identity matrix, we obtain a first-order method. Another possibility is to chose $W(\boldsymbol{x}^k, \boldsymbol{u}^k)$ as the second-order derivatives of the cost only. △

It can be shown that the update direction $(\boldsymbol{\Delta}\mathbf{x}^k, \boldsymbol{\Delta}\mathbf{u}^k)$ is obtained solving the **Linear Quadratic (LQ) problem**

$$\begin{aligned}\min_{\boldsymbol{\Delta}\mathbf{x},\boldsymbol{\Delta}\mathbf{u}} \; & \sum_{t=0}^{T-1} \left(\begin{bmatrix} a_t^k \\ b_t^k \end{bmatrix}^\top \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix}^\top \begin{bmatrix} Q_t^k & S_t^k \\ S_t^{k\top} & R_t^k \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} \right) \\ & \quad + a_T^{k\top} \Delta x_T + \Delta x_T^\top Q_T^k \Delta x_T \\ \text{subj.to} \; & \Delta x_{t+1} = A_t^k \Delta x_t + B_t^k \Delta u_t, \;\; t \in [0, T-1] \\ & \Delta x_0 = 0, \end{aligned} \tag{B.30}$$

where $Q_t^k \in \mathbb{R}^{n\times m}$, $S_t^k \in \mathbb{R}^{n\times m}$ and $R_t^k \in \mathbb{R}^{m\times m}$ are proper weight matrices, components of $W(\boldsymbol{x}^k, \boldsymbol{u}^k)$, while $A_t^k, B_t^k, a_t^k, b_t^k$ follow the shorthand notation in (3.11). Algorithm 16 recaps the procedure described so far.

Algorithm 16 PRONTO

```
for $k = 0, 1, 2 \dots$ do
    Step 1: compute descent direction $(\mathbf{\Delta x}^k, \mathbf{\Delta u}^k)$ by solving the LQ problem
        $$\begin{aligned} \min_{\mathbf{\Delta x}, \mathbf{\Delta u}} \; & \sum_{t=0}^{T-1} \left( \begin{bmatrix} a_t^k \\ b_t^k \end{bmatrix}^\top \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix}^\top \begin{bmatrix} Q_t^k & S_t^k \\ S_t^{k\top} & R_t^k \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix} \right) \\ & + a_T^{k\top} \Delta x_T + \Delta x_T^\top Q_T^k \Delta x_T \\ \text{subj.to} \; & \Delta x_{t+1} = A_t^k \Delta x_t + B_t^k \Delta u_t, \;\; t \in [0, T-1] \\ & \Delta x_0 = 0, \end{aligned} \tag{B.31}$$
    for $t = 0, \dots, T-1$ do
        Step 2: update (unfeasible) curve
            $$\begin{aligned} \alpha_t^{k+1} &= x_t^k + \gamma^k \Delta x_t^k \\ \mu_t^{k+1} &= u_t^k + \gamma^k \Delta u_t^k \end{aligned} \tag{B.32}$$
        Step 3: compute new (feasible) trajectory
            $$\begin{aligned} u_t^{k+1} &= \mu_t^{k+1} + K_t(\alpha_t^{k+1} - x_t^{k+1}) \\ x_{t+1}^{k+1} &= f(x_t^{k+1}, u_t^{k+1}) \end{aligned}$$
```

Appendix C

Averaging Theory for Two-time-scale Systems

We report [13, Theorem 2.2.4], which is a useful result in the context of averaging theory for two-time-scale systems. Consider the time-varying system

$$\chi_{k+1} = \mathcal{A}(z_t)\chi_k + h(z_t, k) + \epsilon g(\chi_k, z_t, k), \tag{C.1a}$$

$$z_{t+1} = z_t + \epsilon f(\chi_k, z_t, k), \tag{C.1b}$$

with $\chi_k \in \mathbb{R}^{n_\chi}$, $z_t \in \mathbb{R}^{n_z}$, $g : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_z} \times \mathbb{N} \to \mathbb{R}^{n_\chi}$, $f : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_z} \times \mathbb{N} \to \mathbb{R}^{n_\chi}$, and $A : \mathbb{R}^{n_z} \to \mathbb{R}^{n_\chi \times n}$. We enforce the following assumptions.

Assumption C.1. *There exists r such that f, g, and h are Lipschitz continuous into* $\mathcal{B}_r(0_{n_\chi + n_z})$. $\triangle$

Assumption C.2. *It holds* $h(0,k) = 0$, $g(0,0,k) = 0$, *and* $f(0,0,k) = 0$ *for all* $k \in \mathbb{N}$. $\triangle$

Assumption C.3. *There exist* $r, m_1, m_2 > 0$ *and* $a_1, a_2 \in (0,1)$ *such that*

$$m_1 a_1^k \leq \left\| \mathcal{A}(z)^k \right\| \leq m_2 a_2^k,$$

for all $z \in \mathcal{B}_r(0_{n_z})$ *and* $k \in \mathbb{N}$. *Moreover, there exists* $k_a > 0$ *such that*

$$\left\| \frac{\partial \mathcal{A}(z)}{\partial z_i} \right\| \leq k_a,$$

for all $i \in \{1, \dots, m\}$ *and* $z \in \mathcal{B}_r(0_{n_z})$. $\triangle$

Assumption C.4. *The function f is piecewise continuous in k with the limit*

$$f^{av}(z) := \lim_{T \to \infty} \frac{1}{T} \sum_{i=\bar{t}+1}^{\bar{t}+T} f(0, z, i) \tag{C.2}$$

existing uniformly in $\bar{k} \in \mathbb{N}$ *and for all* $z \in \mathcal{B}_r(0_{n_z})$. △

We associate a so-called *averaged system* to (C.1) given by

$$z_{k+1}^{\text{av}} = z_k^{\text{av}} + \epsilon f^{\text{av}}(z_k^{\text{av}}). \tag{C.3}$$

Assumption C.5. *Consider* f^{av} *as defined in* (C.2) *and let* $\Delta f : \mathbb{R}^{n_z} \times \mathbb{N} \to \mathbb{R}^{n_z}$ *be defined as*

$$\Delta f(z,k) := f(0,z,k) - f^{av}(z).$$

Then, there exists a nonnegative strictly decreasing function $\nu(k)$ *such that*

$$\lim_{k\to\infty} \nu(k) = 0,$$

and

$$\left\| \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} \tilde{f}(z,i) \right\| \leq \nu(T) \|z\|, \qquad \left\| \frac{1}{T} \sum_{i=\bar{k}+1}^{\bar{k}+T} \frac{\partial \tilde{f}(z,i)}{\partial z} \right\| \leq \nu(T),$$

uniformly in $\bar{k} \in \mathbb{N}$ *and for all* $z \in \mathcal{B}_r(0_{n_z})$. △

Theorem C.1. *[13, Theorem 2.2.4] Consider System* (C.1) *and let Assumptions C.1, C.2, C.3, C.4 and C.5 hold. If there exists* $\epsilon_0 > 0$ *such that, for all* $\epsilon \in (0, \epsilon_0)$*, the origin is exponentially stable for System* (C.3)*, then there exists* $\epsilon_1 > 0$ *such that, for all* $\epsilon \in (0, \epsilon_1)$*, the origin is an exponentially stable equilibrium of System* (C.1). △

Ringraziamenti

Quello che si conclude con queste pagine è stato, per me, uno dei viaggi più sfidanti, duri e formativi della mia vita. Al contempo, è stato anche un percorso incredibile, al quale guardando indietro, faccio a volte fatica a credere.

È inevitabile quindi rivolgere il ringraziamento più grande a Giuseppe, che ha reso tutto questo possibile. Grazie per avermi offerto questa opportunità, grazie per aver riposto in me fiducia anche quando nemmeno io riuscivo ad averne, grazie per essere stato capace di accettare e gestire questa mia adolescenza scientifica, grazie per tutto quello che mi hai insegnato, non solo sul piano scientifico, ma anche umano, comunicativo e lavorativo. Spero di essere stato, almeno in parte, all'altezza. Grazie anche per aver creato un gruppo in cui il valore umano ed il rispetto sono sempre stati al primo posto. È evidente per tutti noi che quello che viviamo quotidianamente è prima di tutto un gruppo di persone, per quanto particolari, dal valore umano impareggiabile, in cui la metrica di valore non è solamente scientifica, ma anche, e soprattutto, personale. Questo è sicuramente merito tuo. Grazie di tutto, di cuore.

Un grazie enorme va poi ad Ivano, un mentore ed un amico, a cui devo tantissimo, che mi ha cresciuto quasi quotidianamente dai tempi della tesi magistrale. Grazie per tutte le chiacchierate, scientifiche e filosofiche, che abbiamo avuto in questi anni, sento che buona parte della mia formazione umana e professionale sia passata anche da lì. Grazie anche per tutte le lezioni, dal come fare il caffè all'ADMM, dal principio del modello interno alle regole del basket durante le partite della Virtus. Sei *prof* nell'animo e ti stimo profondamente. Grazie per avermi insegnato a non aver paura dell'ignoto.

Thanks to Professor Ames, who gave me the opportunity to spend a remarkable semester at Caltech, my months in California were an incredible opportunity where I learned a lot from every point of view. Thank you so much.

Thanks to John (Hauser), an incredible genius whom I had the opportunity to meet, who illuminated few weeks in November with functional analysis and delightful wine-based dinners. I now completely understand why Giuseppe admires him so much.

Grazie poi a Lorenzo (Pichierri), amico prima e collega poi, compagno di interminabili giornate ed infinite serate scientifiche e non, conclusesi sempre nella fedelissima Piazza San Francesco. Grazie per aver reso il lavoro meno lavoro e grazie per

avermi sempre spronato, con le buone o con le cattive, nei momenti più bui. Dalle salite in bicicletta alle notti americane, le bestie erano sempre lì a proteggermi.

Grazie al CASY e *¡hasta l'otimization siempre!* Grazie a tutti i fantastici colleghi che compongono la nostra Comune: grazie al nostro illuminato Presidente Guido, grazie a Marco (scheggia impazzita), allo zio Testa, a Simone, Trama, Alice, Riccardino, il ministro Cecconi e Biagio(ne). Grazie per esserci sempre stati, dalle discussioni alla lavagna ai più eleganti Pizza CASY. E grazie anche ai colleghi passati Lolgent, Marione e Andrea (Camisa), anche lui mentore sotto tantissimi aspetti. Grazie.

Grazie a tutti i miei amici. Grazie alla nostra semi-disfunzionale famiglia bolognese e grazie agli amici di sempre di Verona. Vi ringrazio veramente dal profondo del cuore, sono veramente fortunato ad avervi accanto.

Grazie a Mamma e Papà, sempre presenti, per il supporto quotidiano, per l'amore e l'affetto che mi avete sempre dato e per i valori che mi avete insegnato. Vi devo tutto.

Grazie a mio fratello, Pietro, esempio di passione, entusiasmo ed umanità. Ti ammiro moltissimo, ho molto da imparare da te.

Grazie alle mie nonne, al loro carattere straordinario e al loro caldo affetto, sono quello che sono anche grazie a voi. Mi avete insegnato tanto.

Grazie a mio zio Gianfranco, mio riferimento ingegneristico, grazie per le lunghe chiacchierate e i buonissimi spunti, letterari e non solo.

Grazie a Martina, alla tua famiglia per avermi accolto con amore e, soprattutto, a te. Non ho parole per ringraziarti, sei stata complice e faro nella notte di questa strana avventura. E lo sei stata, letteralmente, dal primo giorno. Non posso fare altro che ringraziarti, per tutto, per l'amore e per ogni singolo istante passato insieme. Sei un fiore unico dal valore inestimabile.

Queste poche righe non possono certo esaurire tutte le straordinarie persone che hanno condiviso con me qualche passo di questo incredibile percorso. Grazie a tutti.

Lorenzo

Bibliography

[1] Y. Abbasi-Yadkori and C. Szepesvári, *Regret bounds for the adaptive control of linear quadratic systems*, Proceedings of the 24th annual conference on learning theory, 2011, pp. 1–26.

[2] D. R Agrawal, H. Parwana, R. K Cosner, U. Rosolia, A. D Ames, and D. Panagou, *A constructive method for designing safe multirate controllers for differentially-flat systems*, IEEE Control Sys. Letters **6** (2021), 2138–2143.

[3] A P. Aguiar, F. A Bayer, J. Hauser, A. J Häusler, G. Notarstefano, A. M Pascoal, A. Rucco, and A. Saccon, *Constrained optimal motion planning for autonomous vehicles using PRONTO*, Sensing and control for autonomous vehicles, 2017, pp. 207–226.

[4] M. Akbari, B. Gharesifard, and T. Linder, *Achieving logarithmic regret via hints in online learning of noisy LQR systems*, IEEE 61st conference on decision and control (cdc), 2022, pp. 4700–4705.

[5] H. Almubarak, N. Sadegh, and E. A Theodorou, *Safety embedded control of nonlinear systems via barrier states*, IEEE Control Systems Letters **6** (2021), 1328–1333.

[6] H. Almubarak, E. A Theodorou, and N. Sadegh, *Hjb based optimal safe control using control barrier functions*, 2021 60th IEEE conf. on decision and control, 2021, pp. 6829–6834.

[7] A. D Ames, X. Xu, J. W Grizzle, and P. Tabuada, *Control barrier function based quadratic programs for safety critical systems*, IEEE Trans. on Automatic Control **62** (2016), no. 8, 3861–3876.

[8] B. D. O. Anderson and J. B. Moore, *Optimal Control - Linear Quadratic Methods*, Prentice-Hall International, Inc., 1989.

[9] B. D. Anderson and J. B Moore, *Optimal control: linear quadratic methods*, Courier Corporation, 2007.

[10] A. M Annaswamy, K. H Johansson, G. J Pappas, et al., *Control for societal-scale challenges: Road map 2030*, IEEE Control Systems Society (2023).

[11] N. Aronszajn, *Theory of reproducing kernels*, Transactions of the American mathematical society **68** (1950), no. 3, 337–404.

[12] A. Aswani, H. Gonzalez, S S. Sastry, and C. Tomlin, *Provably safe and robust learning-based model predictive control*, Automatica **49** (2013), no. 5, 1216–1226.

[13] E-W Bai, L-C Fu, and S. S. Sastry, *Averaging analysis for discrete time and sampled data adaptive systems*, IEEE Transactions on Circuits and Systems **35** (1988), no. 2, 137–148.

[14] E.-W. Bai and S. S. Sastry, *Persistency of excitation, sufficient richness and parameter convergence in discrete time adaptive control*, Systems & control letters **6** (1985), no. 3, 153–163.

[15] S. Barratt and S. Boyd, *Stochastic control with affine dynamics and extended quadratic costs*, IEEE Transactions on Automatic Control **67** (2021), no. 1, 320–335.

[16] N. Bastianello, R. Carli, L. Schenato, and M. Todescato, *Asynchronous distributed optimization over lossy networks via relaxed ADMM: Stability and linear convergence*, IEEE TAC **66** (2020), no. 6, 2620–2635.

[17] F. A Bayer, G. Notarstefano, and F. Allgöwer, *A projected SQP method for nonlinear optimal control with quadratic convergence*, IEEE conference on decision and control CDC, 2013, pp. 6463–6468.

[18] T. Beckers, D. Kulić, and S. Hirche, *Stable Gaussian process based tracking control of Euler–Lagrange systems*, Automatica **103** (2019), 390–397.

[19] T. Beckers, J. Umlauft, and S. Hirche, *Stable model-based control with Gaussian process regression for robot manipulators*, IFAC-PapersOnLine **50** (2017), no. 1, 3877–3884.

[20] G. Belgioioso, A. Nedić, and S. Grammatico, *Distributed generalized Nash equilibrium seeking in aggregative games on time-varying networks*, IEEE TAC **66** (2020), no. 5, 2061–2075.

[21] G. Belgioioso, P. Yi, S. Grammatico, and L. Pavel, *Distributed generalized Nash equilibrium seeking: An operator-theoretic perspective*, IEEE CSM **42** (2022), no. 4, 87–102.

[22] R. Bellman, *Dynamic programming*, Princeton University Press, 1957.

[23] J. Berberich, A. Koch, C. W Scherer, and F. Allgöwer, *Robust data-driven state-feedback design*, 2020 american control conference (acc), 2020, pp. 1532–1538.

[24] F. Berkenkamp, A. P Schoellig, and A. Krause, *Safe controller optimization for quadrotors with Gaussian processes*, International conference on robotics and automation (icra), 2016, pp. 491–496.

[25] D. Bertsekas and S. E Shreve, *Stochastic optimal control: the discrete-time case*, Vol. 5, Athena Scientific, 1996.

[26] D. P Bertsekas, *Constrained optimization and lagrange multiplier methods*, Athena Scientific, 1982.

[27] ______, *Nonlinear programming* (1999).

[28] ______, *Nonlinear programming*, Athena Scientific, 1999.

[29] ______, *Dynamic programming and optimal control*, 3rd ed., Vol. 2, Athena Scientific, Belmont (MA), 2011.

[30] ______, *Nonlinear programming*, Athena Scientific, 2016.

[31] D. P. Bertsekas, *Dynamic programming and optimal control*, 2nd ed., Vol. 1, Athena Scientific, 2005.

[32] J. T. Betts, *Practical methods for optimal control using nonlinear programming*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.

[33] T. Bian and Z.-P. Jiang, *Value iteration and adaptive dynamic programming for data-driven adaptive optimal control design*, Automatica **71** (2016), 348–360.

[34] L. T Biegler, *Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation*, Computers & chemical engineering **8** (1984), no. 3-4, 243–247.

[35] L. T Biegler and V. M Zavala, *Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization*, Computers & Chemical Engineering **33** (2009), no. 3, 575–582.

[36] M. Binder, G. Darivianakis, A. Eichler, and J. Lygeros, *Approximate explicit model predictive controller using Gaussian processes*, IEEE conference on decision and control (CDC), 2019, pp. 841–846.

[37] F. Blanchini et al., *Set-theoretic methods in control*, Springer, 2008.

[38] H. G. Bock and K. J. Plitt, *A multiple shooting algorithm for direct solution of optimal control problems*, Proceedings of the 9th ifac world congress budapest (hungary), 1984July, pp. 242 –247.

[39] M. Borghesi, A. Bosso, and G. Notarstefano, *On-policy data-driven linear quadratic regulator via model reference adaptive reinforcement learning*, 2023 62nd IEEE conference on decision and control (CDC), 2023, pp. 32–37.

[40] F. Borrelli and T. Keviczky, *Distributed LQR design for identical dynamically decoupled systems*, IEEE Transactions on Automatic Control **53** (2008), no. 8, 1901–1912.

[41] L. Bottou, F. E Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, Siam Review **60** (2018), no. 2, 223–311.

[42] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, SIAM, 1994.

[43] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[44] S. J Bradtke, B E. Ydstie, and A. G Barto, *Adaptive linear quadratic control using policy iteration*, Proceedings of 1994 american control conference-acc'94, 1994, pp. 3475–3479.

[45] L. Brunke, M. Greeff, A. W Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P Schoellig, *Safe learning in robotics: From learning-based control to safe reinforcement learning*, Annual Review of Control, Robotics, and Autonomous Systems **5** (2022), 411–444.

[46] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control - Optimization, Estimation, and Control*, Hemisphere Publishing Cooperation, 1975.

[47] A. E. Bryson and Y.-C. Ho, *Applied optimal control: Opt., estimation and control*, Routledge, 1975.

[48] J. Bu, A. Mesbahi, M. Fazel, and M. Mesbahi, *LQR through the lens of first order methods: Discrete-time case*, arXiv preprint arXiv:1907.08921 (2019).

[49] J. Bu, A. Mesbahi, and M. Mesbahi, *On topological properties of the set of stabilizing feedback gains*, IEEE Transactions on Automatic Control **66** (2020), no. 2, 730–744.

[50] F. Bullo, *Lectures on network systems*, Vol. 1, Kindle Direct Publishing Seattle, DC, USA, 2020.

[51] D. Burk, A. Völz, and K. Graichen, *A modular framework for distributed model predictive control of nonlinear continuous-time systems (graMPC-d)*, Optimization and Engineering (2021), 1–25.

[52] M. C Campi and P. Kumar, *Adaptive linear quadratic gaussian control: the cost-biased approach revisited*, SIAM Journal on Control and Optimization **36** (1998), no. 6, 1890–1907.

[53] A. Carè, R. Carli, A. Dalla Libera, D. Romeres, and G. Pillonetto, *Kernel methods and gaussian processes for system identification and control: A road map on regularized kernel-based learning for control*, IEEE Control Systems Magazine **43** (2023), no. 5, 69–110.

[54] G. Carnevale, A. Camisa, and G. Notarstefano, *Distributed online aggregative optimization for dynamic multi-robot coordination*, IEEE TAC (2022).

[55] G. Carnevale, F. Fabiani, F. Fele, K. Margellos, and G. Notarstefano, *Tracking-based distributed equilibrium seeking for aggregative games*, preprint arXiv:2210.14547 (2022).

[56] G. Carnevale, N. Mimmo, and G. Notarstefano, *Aggregative feedback optimization for distributed cooperative robotics*, IFAC-PapersOnLine **55** (2022), no. 13, 7–12.

[57] G. Carnevale and G. Notarstefano, *A learning-based distributed algorithm for personalized aggregative optimization*, Cdc, 2022, pp. 1576–1581.

[58] A. Cassel, A. Cohen, and T. Koren, *Logarithmic regret for learning linear quadratic regulators efficiently*, International conference on machine learning, 2020, pp. 1328–1337.

[59] F. Celi, G. Baggio, and F. Pasqualetti, *Data-driven eigenstructure assignment for sparse feedback design*, 2023 62nd IEEE conference on decision and control (cdc), 2023, pp. 618–623.

[60] M. Chen, D. Wang, X. Wang, Z.-G. Wu, and W. Wang, *Distributed aggregative optimization via finite-time dynamic average consensus*, IEEE Transactions on Network Science and Engineering (2023).

[61] Y. Chen, M. Ahmadi, and A. D Ames, *Optimal safe controller synthesis: A density function approach*, 2020 american control conf., 2020, pp. 5407–5412.

[62] Z. Chen and S. Liang, *Distributed aggregative optimization with quantized communication*, Kyber. **58** (2022), no. 1, 123–144.

[63] A. Chiuso and G. Pillonetto, *System identification: A machine learning perspective*, Annual Review of Control, Robotics, and Autonomous Systems **2** (2019), 281–304.

[64] J. J Choi, D. Lee, K. Sreenath, C. J Tomlin, and S. L Herbert, *Robust control barrier–value functions for safety-critical control*, 2021 60th IEEE conf. on decision and control, 2021, pp. 6814–6821.

[65] G. Chowdhary, H. A Kingravi, J. P How, and P. A Vela, *Bayesian nonparametric adaptive control using Gaussian processes*, IEEE Transactions on Neural Networks and Learning Systems **26** (2014), no. 3, 537–550.

[66] M. H Cohen and C. Belta, *Approximate optimal control for safety-critical systems with control barrier functions*, 2020 59th IEEE conf. on decision and control, 2020, pp. 2062–2067.

[67] E. Crisostomi, B. Ghaddar, F. Häusler, J. Naoum-Sawaya, G. Russo, R. Shorten, et al., *Analytics for the sharing economy: Mathematics, engineering and business perspectives*, Springer, 2020.

[68] N. Csomay-Shanklin, A. J Taylor, U. Rosolia, and A. D Ames, *Multi-rate planning and control of uncertain nonlinear systems: Model predictive control and control lyapunov functions*, 2022 IEEE 61st conf. on decision and control, 2022, pp. 3732–3739.

[69] J. De O. Pantoja, *Differential dynamic programming and newton's method*, International Journal of Control **47** (1988), no. 5, 1539–1553.

[70] C. De Persis and P. Tesi, *Formulas for data-driven control: Stabilization, optimality, and robustness*, IEEE Transactions on Automatic Control **65** (2019), no. 3, 909–924.

[71] ———, *Low-complexity learning of linear quadratic regulators from noisy data*, Automatica **128** (2021), 109548.

[72] ———, *Learning controllers for nonlinear systems from data*, Annual Reviews in Control (2023), 100915.

[73] ———, *Learning controllers for nonlinear systems from data*, Annual Reviews in Control (2023), 100915.

[74] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, *On the sample complexity of the linear quadratic regulator*, Foundations of Computational Mathematics **20** (2020), no. 4, 633–679.

[75] S. Dean, S. Tu, N. Matni, and B. Recht, *Safely learning to control the constrained linear quadratic regulator*, IEEE american control conference (ACC), 2019, pp. 5582–5588.

[76] M. P Deisenroth, J. Peters, and C. E Rasmussen, *Approximate dynamic programming with Gaussian processes*, American control conference (ACC), 2008, pp. 4480–4485.

[77] M. Diehl, H. J. Ferreau, and N. Haverbeke, *Efficient numerical methods for nonlinear MPC and moving horizon estimation*, Nonlinear model predictive control, 2009, pp. 391–417.

[78] A. Dontchev and W. Hager, *Lipschitzian stability for state constrained nonlinear optimal control*, SIAM Jour. on Control and Opt. **36** (1998), no. 2, 698–718.

[79] F. Dörfler, M. R Jovanović, M. Chertkov, and F. Bullo, *Sparsity-promoting optimal wide-area control of power networks*, IEEE Transactions on Power Systems **29** (2014), no. 5, 2281–2291.

[80] F. Dörfler, P. Tesi, and C. De Persis, *On the role of regularization in direct data-driven LQR control*, IEEE conference on decision and control (CDC), 2022, pp. 1091–1098.

[81] ———, *On the certainty-equivalence approach to direct data-driven LQR design*, IEEE Transactions on Automatic Control (2023).

[82] J. C Dunn and D. P Bertsekas, *Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems*, Journal of Optimization Theory and Applications **63** (1989), no. 1, 23–38.

[83] A. Engelmann et al., *Decomposition of nonconvex optimization via bi-level distributed aladin*, IEEE Transactions on Control of Network Systems **7** (2020), no. 4, 1848–1858.

[84] F. Fabiani, M. A. Tajeddini, H. Kebriaei, and S. Grammatico, *Local Stackelberg equilibrium seeking in generalized aggregative games*, IEEE TAC **67** (2021), no. 2, 965–970.

[85] A. Falsone, I. Notarnicola, G. Notarstefano, and M. Prandini, *Tracking-ADMM for distributed constraint-coupled optimization*, Aut. **117** (2020), 108962.

[86] Y. Fanger, J. Umlauft, and S. Hirche, *Gaussian processes for dynamic movement primitives with application in knowledge-based cooperation*, International conference on intelligent robots and systems (IROS), 2016, pp. 3913–3919.

[87] M. Fardad and M. R Jovanović, *On the design of optimal structured and sparse feedback gains via sequential convex programming*, 2014 american control conference, 2014, pp. 2426–2431.

[88] F. Farina, A. Camisa, A. Testa, I. Notarnicola, and G. Notarstefano, *Disropt: a python framework for distributed optimization*, IFAC-PapersOnLine **53** (2020), no. 2, 2666–2671.

[89] S. Fattahi, G. Fazelnia, J. Lavaei, and M. Arcak, *Transformation of optimal centralized controllers into near-globally optimal static distributed controllers*, IEEE Transactions on Automatic Control **64** (2018), no. 1, 66–80.

[90] T. Faulwasser, L. Grüne, M. A Müller, et al., *Economic nonlinear model predictive control*, Now Foundations and Trends, 2018.

[91] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, *Global convergence of policy gradient methods for the linear quadratic regulator*, International conference on machine learning, 2018, pp. 1467–1476.

[92] L. Ferrarotti and A. Bemporad, *Synthesis of optimal feedback controllers from data via stochastic gradient descent*, 2019 18th european control conference (ecc), 2019, pp. 2486–2491.

[93] M. Filo and B. Bamieh, *Function space approach for gradient descent in optimal control*, IEEE american control conference (ACC), 2018, pp. 3447–3453.

[94] J. F Fisac, A. K Akametalu, M. N Zeilinger, S. Kaynama, J. Gillula, and C. J Tomlin, *A general safety framework for learning-based control in uncertain robotic systems*, IEEE Transactions on Automatic Control **64** (2018), no. 7, 2737–2752.

[95] G. Foderaro, S. Ferrari, and T. A Wettergren, *Distributed optimal control for multi-agent trajectory optimization*, Aut. **50** (2014), no. 1, 149–154.

[96] G. Folland, *Higher-order derivatives and Taylor's formula in several variables*, Preprint (2005), 1–4.

[97] H. Frankowska and S. Plaskacz, *Semicontinuous solutions of hamilton–jacobi–bellman equations with degenerate state constraints*, Jour. of math. analysis and appl. **251** (2000), no. 2, 818–838.

[98] L. Furieri, Y. Zheng, and M. Kamgarpour, *Learning the globally optimal distributed LQ regulator*, Conference on learning for dynamics and control, 2020, pp. 287–297.

[99] D. Gadjov and L. Pavel, *Single-timescale distributed gne seeking for aggregative games over networks via forward–backward operator splitting*, IEEE TAC **66** (2020), no. 7, 3259–3266.

[100] T. Ge, W. Lin, and J. Feng, *Invariance principles allowing of non-lyapunov functions for estimating attractor of discrete dynamical systems*, IEEE TAC **57** (2011), no. 2, 500–505.

[101] P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer, *Accelerated gradient methods and dual decomposition in distributed model predictive control*, Aut. **49** (2013), no. 3, 829–833.

[102] P. Giselsson and A. Rantzer, *Large-scale and distributed optimization: An introduction*, Springer, 2018.

[103] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[104] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.

[105] P. D Grontas, M. W Fisher, and F. Dörfler, *Distributed and constrained h 2 control design via system level synthesis and dual consensus ADMM*, Cdc, 2022, pp. 301–307.

[106] S. Gros and M. Zanon, *Data-driven economic nMPC using reinforcement learning*, IEEE Transactions on Automatic Control **65** (2019), no. 2, 636–648.

[107] L. Grüne, E. D Sontag, and F. R Wirth, *Asymptotic stability equals exponential stability, and iss equals finite energy gain—if you twist your eyes*, Systems & Control Letters **38** (1999), no. 2, 127–134.

[108] X. Guo, D. Keivan, G. Dullerud, P. Seiler, and B. Hu, *Complexity of derivative-free policy optimization for structured H_∞ control*, Advances in Neural Information Processing Systems **36** (2024).

[109] W. M Haddad and V. Chellaboina, *Nonlinear dynamical systems and control*, Nonlinear dynamical systems and control, 2011.

[110] W. W Hager and H. Zhang, *A survey of nonlinear conjugate gradient methods*, Pacific journal of Optimization **2** (2006), no. 1, 35–58.

[111] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, *A survey of distributed optimization methods for multi-robot systems*, preprint arXiv:2103.12840 (2021).

[112] S. Hassan-Moghaddam, M. R Jovanović, and S. Meyn, *Data-driven proximal algorithms for the design of structured optimal feedback gains*, IEEE american control conference (ACC), 2019, pp. 5846–5850.

[113] J. Hauser, *A projection operator approach to the optimization of trajectory functionals*, IFAC Proceedings Volumes **35** (2002), no. 1, 377–382.

[114] J. Hauser and A. Saccon, *A barrier function method for the optimization of trajectory functionals with constraints*, Cdc, 2006, pp. 864–869.

[115] M. R Hestenes, E. Stiefel, et al., *Methods of conjugate gradients for solving linear systems*, Journal of research of the National Bureau of Standards **49** (1952), no. 6, 409–436.

[116] L. Hewing, J. Kabzan, and M. N Zeilinger, *Cautious model predictive control using Gaussian process regression*, IEEE Transactions on Control Systems Technology **28** (2019), no. 6, 2736–2743.

[117] L. Hewing, K. P Wabersich, M. Menner, and M. N Zeilinger, *Learning-based Model Predictive Control: Toward safe learning in control*, Annual Review of Control, Robotics, and Autonomous Systems **3** (2020), 269–296.

[118] B. Houska and J. Shi, *Distributed MPC with aladin—a tutorial*, Acc, 2022, pp. 358–363.

[119] R. Hult, M. Zanon, S. Gros, and P. Falcone, *Optimal coordination of automated vehicles at intersections: Theory and experiments*, IEEE Transactions on Control Systems Technology **27** (2018), no. 6, 2510–2525.

[120] A. Isidori, *Lectures in feedback design for multivariable systems*, Springer, 2017.

[121] A. Jain, T. Nghiem, M. Morari, and R. Mangharam, *Learning and control using Gaussian processes*, Acm/IEEE international conference on cyber-physical systems (iccps), 2018, pp. 140–149.

[122] M. K. Jensen, *Aggregative games and best-reply potentials*, Economic theory **43** (2010), no. 1, 45–66.

[123] J. L Jerez, P. J Goulart, S. Richter, G. A Constantinides, E. C Kerrigan, and M. Morari, *Embedded online optimization for model predictive control at megahertz rates*, IEEE Transactions on Automatic Control **59** (2014), no. 12, 3238–3251.

[124] Y. Jiang and Z.-P. Jiang, *Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics*, Automatica **48** (2012), no. 10, 2699–2704.

[125] J. Jiao, H. L Trentelman, and M K. Camlibel, *A suboptimality approach to distributed linear quadratic optimal control*, IEEE Transactions on Automatic Control **65** (2019), no. 3, 1218–1225.

[126] G. Jing, H. Bai, J. George, and A. Chakrabortty, *Model-free optimal control of linear multiagent systems via decomposition and hierarchical approximation*, IEEE Transactions on Control of Network Systems **8** (2021), no. 3, 1069–1081.

[127] G. Jing, H. Bai, J. George, A. Chakrabortty, and P. K Sharma, *Learning distributed stabilizing controllers for multi-agent systems*, IEEE Control Systems Letters **6** (2021), 301–306.

[128] R. M Johnstone, C R. Johnson Jr, R. R Bitmead, and B. D. Anderson, *Exponential convergence of recursive least squares with exponential forgetting factor*, Systems & Control Letters **2** (1982), no. 2, 77–82.

[129] J. Kabzan, L. Hewing, A. Liniger, and M. N Zeilinger, *Learning-based model predictive control for autonomous racing*, IEEE Robotics and Automation Letters **4** (2019), no. 4, 3363–3370.

[130] P. Kapasouris, M. Athans, and G. Stein, *Design of feedback control systems for unstable plants with saturating actuators*, Proc. IFAC symp. on nonlinear control system design, 1990, pp. 302–307.

[131] V. Katewa and F. Pasqualetti, *Minimum-gain pole placement with sparse static feedback*, IEEE Transactions on Automatic Control **66** (2020), no. 8, 3445–3459.

[132] T. Kavuncu, A. Yaraneri, and N. Mehr, *Potential iLQR: A potential-minimizing controller for planning multi-agent interactive trajectories*, preprint arXiv:2107.04926 (2021).

[133] S. S Kia, B. Van Scoy, J. Cortes, R. A Freeman, K. M Lynch, and S. Martinez, *Tutorial on dynamic average consensus: The problem, its applications, and the algorithms*, IEEE Control Systems Magazine **39** (2019), no. 3, 40–72.

[134] D. E. Kirk, *Optimal control theory*, Prentice-Hall Inc., 1970.

[135] B. Kiumarsi, F. L Lewis, and Z.-P. Jiang, *H_∞ control of linear discrete-time systems: Off-policy reinforcement learning*, Automatica **78** (2017), 144–152.

[136] B. Kiumarsi, F. L Lewis, M.-B. Naghibi-Sistani, and A. Karimpour, *Optimal tracking control of unknown discrete-time linear systems using input-output measured data*, IEEE transactions on cybernetics **45** (2015), no. 12, 2770–2779.

[137] D. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Transactions on Automatic Control **13** (1968), no. 1, 114–115.

[138] J. Kocijan, *Modelling and control of dynamic systems using Gaussian process models*, Springer, 2016.

[139] P. N Köhler, M. A Müller, J. Pannek, and F. Allgöwer, *Distributed economic model predictive control for cooperative supply chain management using customer forecast information*, IFAC Journal of Systems and Control **15** (2021), 100125.

[140] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, *Learning-based model predictive control for safe exploration*, IEEE conference on decision and control (CDC), 2018, pp. 6059–6066.

[141] D. Kouzoupis, H. J. Ferreau, H. Peyrl, and M. Diehl, *First-order methods in embedded nonlinear model predictive control*, 2015 european control conference (ecc), 2015, pp. 2617–2622.

[142] K. Krauth, S. Tu, and B. Recht, *Finite-time analysis of approximate policy iteration for the linear quadratic regulator*, Advances in Neural Information Processing Systems **32** (2019).

[143] W. Levine and M. Athans, *On the determination of the optimal constant output feedback gains for linear multivariable systems*, IEEE Transactions on Automatic control **15** (1970), no. 1, 44–48.

[144] W. Li and E. Todorov, *Iterative linear quadratic regulator design for nonlinear biological movement systems.*, International conference on informatics in control, automation and robotics, 2004, pp. 222–229.

[145] X. Li, X. Yi, and L. Xie, *Distributed online convex optimization with an aggregative variable*, IEEE Transactions on Control of Network Systems **9** (2021), no. 1, 438–449.

[146] Y. Li, Y. Tang, R. Zhang, and N. Li, *Distributed reinforcement learning for decentralized linear quadratic control: A derivative-free policy optimization approach*, IEEE Transactions on Automatic Control (2021).

[147] L-Z Liao and C. A Shoemaker, *Convergence in unconstrained discrete-time differential dynamic programming*, IEEE Transactions on Automatic Control **36** (1991), no. 6, 692–706.

[148] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*, Princeton university press, 2011.

[149] F. Lin, M. Fardad, and M. R Jovanovic, *Augmented Lagrangian approach to design of structured optimal state feedback gains*, IEEE Transactions on Automatic Control **56** (2011), no. 12, 2923–2929.

[150] ———, *Optimal control of vehicular formations with nearest neighbor interactions*, IEEE Transactions on Automatic Control **57** (2011), no. 9, 2203–2218.

[151] F. Lin, M. Fardad, and M. R Jovanović, *Design of optimal sparse feedback gains via the alternating direction method of multipliers*, IEEE Transactions on Automatic Control **58** (2013), no. 9, 2426–2431.

[152] M. Liu, G. Chowdhary, B. C. Da Silva, S.-Y. Liu, and J. P How, *Gaussian processes for learning and control: A tutorial with examples*, IEEE Control Systems Magazine **38** (2018), no. 5, 53–86.

[153] L. Ljung, *System identification*, Wiley encyclopedia of electrical and electronics engineering (1999), 1–19.

[154] V. G Lopez, M. Alsalti, and M. A Müller, *Efficient off-policy Q-learning for data-based discrete-time LQR problems*, arXiv preprint arXiv:2105.07761 (2021).

[155] ———, *Efficient off-policy Q-learning for data-based discrete-time LQR problems*, IEEE Transactions on Automatic Control (2023).

[156] J. Ma, Z. Cheng, X. Zhang, M. Tomizuka, and T. H. Lee, *Optimal decentralized control for uncertain systems by symmetric gauss–seidel semi-proximal alm*, IEEE Transactions on Automatic Control **66** (2021), no. 11, 5554–5560.

[157] M. Maiworm, D. Limon, J. M. Manzano, and R. Findeisen, *Stability of Gaussian process learning based output feedback model predictive control*, IFAC-PapersOnLine **51** (2018), no. 20, 455–461.

[158] M. Manetti, M. Morandini, and P. Mantegazza, *Servo-fluid-elastic modeling of contactless levitated adaptive secondary mirrors*, Computational Mechanics **50** (2012), 85–98.

[159] I. Markovsky and F. Dörfler, *Behavioral systems theory in data-driven analysis, signal processing, and control*, Annual Reviews in Control **52** (2021), 42–64.

[160] H. Modares and F. L Lewis, *Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning*, IEEE Transactions on Automatic Control **59** (2014), no. 11, 3051–3056.

[161] H. Modares, F. L Lewis, and Z.-P. Jiang, *Optimal output-feedback control of unknown continuous-time linear systems using off-policy reinforcement learning*, IEEE Transactions on Cybernetics **46** (2016), no. 11, 2401–2410.

[162] H. Mohammadi, A. Zare, M. Soltanolkotabi, and M. R Jovanović, *Convergence and sample complexity of gradient methods for the model-free linear–quadratic regulator problem*, IEEE Transactions on Automatic Control **67** (2021), no. 5, 2435–2450.

[163] T. G Molnar, R. K Cosner, A. W Singletary, W. Ubellacker, and A. D Ames, *Model-free safety-critical control for robotic systems*, IEEE Rob. and Aut. Letters **7** (2021), no. 2, 944–951.

[164] M. A Müller and F. Allgöwer, *Economic and distributed model predictive control: Recent developments in optimization-based control*, SICE Jour. of Control, Measurement, and Sys. Integration **10** (2017), no. 2, 39–52.

[165] A. Nedić and J. Liu, *Distributed optimization for control*, Ann. Rev. of Contr., Rob., and Aut. Sys. **1** (2018), 77–103.

[166] T. X Nghiem and C. N Jones, *Data-driven demand response modeling and control of buildings with gaussian processes*, 2017 american control conference (acc), 2017, pp. 2919–2924.

[167] D. Nguyen-Tuong and J. Peters, *Model learning for robot control: a survey*, Cognitive processing **12** (2011), no. 4, 319–340.

[168] D. Nguyen-Tuong, M. Seeger, and J. Peters, *Model learning with local Gaussian process regression*, Advanced Robotics **23** (2009), no. 15, 2015–2034.

[169] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer, 1999.

[170] G. Notarstefano, I. Notarnicola, and A. Camisa, *Distributed optimization for smart cyber-physical networks*, Found. and Trends® in Systems and Control **7** (2019), no. 3, 253–383.

[171] A. Padoan, G. Scarciotti, and A. Astolfi, *A geometric characterization of the persistence of excitation condition for the solutions of autonomous systems*, IEEE Transactions on Automatic Control **62** (2017), no. 11, 5666–5677.

[172] B. Pang, T. Bian, and Z.-P. Jiang, *Data-driven finite-horizon optimal control for linear time-varying discrete-time systems*, 2018 IEEE conference on decision and control (cdc), 2018, pp. 861–866.

[173] ______, *Robust policy iteration for continuous-time linear quadratic regulation*, IEEE Transactions on Automatic Control **67** (2021), no. 1, 504–511.

[174] J. D. O Pantoja and D. Mayne, *Sequential quadratic programming algorithm for discrete optimal control problems with control inequality constraints*, International Journal of Control **53** (1991), no. 4, 823–836.

[175] F. Parise, S. Grammatico, B. Gentile, and J. Lygeros, *Distributed convergence to Nash equilibria in network and average aggregative games*, Aut. **117** (2020), 108959.

[176] F. Parise and A. Ozdaglar, *Analysis and interventions in large network games*, Ann. Rev. of Control, Robotics, and Autonomous Sys. **4** (2021), 455–486.

[177] S. Park, D. Lee, H. J. Ahn, C. Tomlin, and S. Moura, *Optimal control of battery fast charging based-on Pontryagin's minimum principle*, Cdc, 2020, pp. 3506–3513.

[178] L. Pichierri, G. Carnevale, L. Sforni, A. Testa, and G. Notarstefano, *A distributed online optimization strategy for cooperative robotic surveillance*, 2023 IEEE icra, 2023, pp. 5537–5543.

[179] B. Polyak, M. Khlebnikov, and P. Shcherbakov, *An lmi approach to structured sparse feedback design in linear control systems*, 2013 european control conference (ecc), 2013, pp. 833–838.

[180] B. T Polyak, *The conjugate gradient method in extremal problems*, USSR Computational Mathematics and Mathematical Physics **9** (1969), no. 4, 94–112.

[181] L. S. Pontryagin, V. G. Boltyanskii, R. Gamkrelidze, and E. F. Mishchenko, *The Mathematical Theory of Optimal Processes*, Wiley (NY), 1962.

[182] C. Possieri and M. Sassano, *Q-learning for continuous-time linear systems: A data-driven implementation of the kleinman algorithm*, IEEE Transactions on Systems, Man, and Cybernetics: Systems **52** (2022), no. 10, 6487–6497.

[183] ______, *Value iteration for continuous-time linear time-invariant systems*, IEEE Transactions on Automatic Control (2022).

[184] C. Qin, H. Zhang, and Y. Luo, *Online optimal tracking control of continuous-time linear systems with unknown dynamics by using adaptive dynamic programming*, International Journal of Control **87** (2014), no. 5, 1000–1009.

[185] C. E. Rasmussen, C. K. Williams, and F. Bach, *Gaussian processes for machine learning*, MIT Press, 2006.

[186] T. Rautert and E. W Sachs, *Computational design of optimal output feedback controllers*, SIAM Journal on Optimization **7** (1997), no. 3, 837–852.

[187] J. B. Rawlings, D. Q Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*, Vol. 2, Nob Hill Publishing Madison, WI, 2017.

[188] B. Recht, *A tour of reinforcement learning: The view from continuous control*, Annual Review of Control, Robotics, and Autonomous Systems **2** (2019), 253–279.

[189] M. F Reis, A P. Aguiar, and P. Tabuada, *Control barrier function-based quadratic programs introduce undesirable asymptotically stable equilibria*, IEEE Control Sys. Letters **5** (2020), no. 2, 731–736.

[190] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, *Fully decentralized admm for coordination and collision avoidance*, 2018 european control conference (ecc), 2018, pp. 825–830.

[191] S. A. A. Rizvi and Z. Lin, *Output feedback reinforcement q-learning control for the discrete-time linear quadratic regulator problem*, 2017 IEEE 56th annual conference on decision and control (cdc), 2017, pp. 1311–1316.

[192] ______, *Reinforcement learning-based linear quadratic regulation of continuous-time systems using dynamic output feedback*, IEEE transactions on cybernetics **50** (2019), no. 11, 4670–4679.

[193] U. Rosolia and A. D Ames, *Multi-rate control design leveraging control barrier functions and model predictive control policies*, IEEE Control Sys. Letters **5** (2020), no. 3, 1007–1012.

[194] U. Rosolia and F. Borrelli, *Learning model predictive control for iterative tasks. a data-driven control framework*, IEEE Transactions on Automatic Control **63** (2017), no. 7, 1883–1896.

[195] U. Rosolia, A. Singletary, and A. D Ames, *Unified multirate control: From low-level actuation to high-level planning*, IEEE Trans. on Automatic Control **67** (2022), no. 12, 6627–6640.

[196] U. Rosolia, X. Zhang, and F. Borrelli, *Data-driven predictive control for autonomous systems*, Annual Review of Control, Robotics, and Autonomous Systems **1** (2018), 259–286.

[197] M. Rotulo, C. De Persis, and P. Tesi, *Data-driven linear quadratic regulation via semidefinite programming*, IFAC-PapersOnLine **53** (2020), no. 2, 3995–4000.

[198] ______, *Online learning of data-driven controllers for unknown switched linear systems*, Automatica **145** (2022), 110519.

[199] A. Saccon, J. Hauser, and A P. Aguiar, *Optimal control on lie groups: The projection operator approach*, IEEE TAC **58** (2013), no. 9, 2230–2245.

[200] D. Sadigh, S. Sastry, S. A Seshia, and A. D Dragan, *Planning for autonomous cars that leverage effects on human actions.*, Robotics: Science and systems, 2016, pp. 1–9.

[201] A. P. Sage, *Optimum systems control*, Prentice-Hall, 1968.

[202] A. D Saravanos, A. Tsolovikos, E. Bakolas, and E. A Theodorou, *Distributed covariance steering with consensus ADMM for stochastic multi-agent systems.*, Robotics: Science and systems, 2021.

[203] M. Sassano and A. Astolfi, *Combining pontryagin's principle and dynamic programming for linear and nonlinear systems*, IEEE TAC **65** (2020), no. 12, 5312–5327.

[204] R. Sengupta, S. Amin, A. Annaswamy, S. Moura, and V. Bulusu, *Smart cities and control*, IEEE Control Systems Magazine **35** (2015), no. 6, 20–21.

[205] L. Sforni, A. Camisa, and G. Notarstefano, *Structured-policy Q-learning: an LMI-based design strategy for distributed reinforcement learning*, IEEE conference on decision and control (CDC), 2022, pp. 4059–4064.

[206] L. Sforni, G. Carnevale, I. Notarnicola, and G. Notarstefano, *On-policy data-driven linear quadratic regulator via combined policy iteration and recursive least squares*, IEEE 62nd conference on decision and control (CDC), 2023, pp. 5047–5052.

[207] ———, *Stability-certified on-policy data-driven lqr via recursive learning and policy gradient*, arXiv preprint arXiv:2403.05367 (2024).

[208] L. Sforni, G. Carnevale, and G. Notarstefano, *A distributed feedback-based framework for nonlinear aggregative optimal control*, IEEE Transactions on Automatic Control (2023). (submitted).

[209] L. Sforni, I. Notarnicola, and G. Notarstefano, *Learning-driven nonlinear optimal control via gaussian process regression*, 2021 60th IEEE conference on decision and control (CDC), 2021, pp. 4412–4417.

[210] ———, *Sparse data-driven LQR via augmented lagrangian-based policy search*, IEEE Transactions on Control of Network Systems (2024). (submitted).

[211] L. Sforni, G. Notarstefano, and A. D. Ames, *Receding horizon CBF-based multi-layer controllers for safe trajectory generation*, 2024 american control conference (ACC), 2024. (accepted).

[212] L. Sforni, S. Spedicato, I. Notarnicola, and G. Notarstefano, *Gopronto: a feedback-based framework for nonlinear optimal control*, arXiv preprint arXiv:2108.13308 (2021).

[213] S. Shin, T. Faulwasser, M. Zanon, and V. M Zavala, *A parallel decomposition scheme for solving long-horizon optimal control problems*, IEEE conference on decision and control (CDC), 2019, pp. 5264–5271.

[214] O. Shorinwa, T. Halsted, and M. Schwager, *Scalable distributed optimization with separable variables in multi-agent networks*, Acc, 2020, pp. 3619–3626.

[215] O. So, K. Stachowicz, and E. A Theodorou, *Multimodal maximum entropy dynamic games*, preprint arXiv:2201.12925 (2022).

[216] S. Spedicato, S. Mahesh, and G. Notarstefano, *A sparse polytopic LPV controller for fully-distributed nonlinear optimal control*, IEEE european control conference (ECC), 2019, pp. 554–559.

[217] S. Spedicato and G. Notarstefano, *Cloud-assisted distributed nonlinear optimal control for dynamics over graph*, IFAC-PapersOnLine **51** (2018), no. 23, 361–366.

[218] N. Srinivas, A. Krause, S. M Kakade, and M. W Seeger, *Information-theoretic regret bounds for Gaussian process optimization in the bandit setting*, IEEE Transactions on Information Theory **58** (2012), no. 5, 3250–3265.

[219] G. Stomberg, A. Engelmann, and T. Faulwasser, *Decentralized non-convex optimization via bi-level SQP and ADMM*, Cdc, 2022.

[220] G. Stomberg et al., *Cooperative distributed MPC via decentralized real-time optimization: Implementation results for robot formations*, Control Engineering Practice **138** (2023), 105579.

[221] R. S Sutton and A. G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[222] M. J. Tenny, S. J. Wright, and J. B. Rawlings, *Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming*, J. Comp. Optim. Appl. **28** (2004Apr.), no. 1, 87–121.

[223] A. Testa, G. Carnevale, and G. Notarstefano, *A tutorial on distributed optimization for cooperative robotics: from setups and algorithms to toolboxes and research directions*, preprint arXiv:2309.04257 (2023).

[224] E. Theodorou, Y. Tassa, and E. Todorov, *Stochastic differential dynamic programming*, Proceedings of the 2010 american control conference, 2010, pp. 1125–1132.

[225] E. Todorov and W. Li, *A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems*, Proceedings of the 2005, american control conference, 2005., 2005, pp. 300–306.

[226] S. Tonkens and S. Herbert, *Refining control barrier functions through hamilton-jacobi reachability*, 2022 IEEE/rsj int. conf. on intel. robots and sys., 2022, pp. 13355–13362.

[227] T. Tsang, D. Himmelblau, and T. F Edgar, *Optimal control via collocation and non-linear programming*, International Journal of Control **21** (1975), no. 5, 763–768.

[228] C. S Turner, *Recursive discrete-time sinusoidal oscillators*, IEEE Signal Processing Magazine **20** (2003), no. 3, 103–111.

[229] J. Umlauft, T. Beckers, and S. Hirche, *Scenario-based optimal control for Gaussian process state space models*, European control conference (ECC), 2018, pp. 1386–1392.

[230] J. Umlauft and S. Hirche, *Feedback linearization based on Gaussian processes with event-triggered online learning*, IEEE Transactions on Automatic Control **65** (2019), no. 10, 4154–4169.

[231] K. G Vamvoudakis, *Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach*, Systems & Control Letters **100** (2017), 14–20.

[232] R. Van Parys and G. Pipeleers, *Distributed MPC for multi-vehicle systems moving in formation*, Robotics and Autonomous Systems **97** (2017), 144–152.

[233] H. J Van Waarde, J. Eising, H. L Trentelman, and M K. Camlibel, *Data informativity: a new perspective on data-driven analysis and control*, IEEE Transactions on Automatic Control **65** (2020), no. 11, 4753–4768.

[234] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, *Acados–a modular open-source framework for fast embedded optimal control*, Mathematical Programming Computation **14** (2022), no. 1, 147–183.

[235] E. E Vlahakis and G. D Halikias, *Distributed LQR methods for networks of non-identical plants*, 2018 IEEE conference on decision and control (cdc), 2018, pp. 6145–6150.

[236] D. Vrabie, O Pastravanu, M. Abu-Khalaf, and F. L Lewis, *Adaptive optimal control for continuous-time linear systems based on policy iteration*, Automatica **45** (2009), no. 2, 477–484.

[237] T. Wang and P. Yi, *Distributed projection-free algorithm for constrained aggregative optimization*, preprint arXiv:2207.11885 (2022).

[238] J. C Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, *A note on persistency of excitation*, Systems & Control Letters **54** (2005), no. 4, 325–329.

[239] C. Wilson and H. Dowlatabadi, *Models of decision making and residential energy use*, Annu. Rev. Environ. Resour. **32** (2007), 169–203.

[240] S. J Wright, *Primal-dual interior-point methods*, SIAM, 1997.

[241] W. Xiao, C. A Belta, and C. G Cassandras, *Sufficient conditions for feasibility of optimal control problems using control barrier functions*, Automatica **135** (2022), 109960.

[242] W. Xiao, C. G Cassandras, and C. A Belta, *Bridging the gap between optimal trajectory planning and safety-critical control with applications to autonomous vehicles*, Automatica **129** (2021), 109592.

[243] L. Xu, B. Guo, C. Galimberti, M. Farina, R. Carli, and G. F. Trecate, *Suboptimal distributed LQR design for physically coupled systems*, IFAC-PapersOnLine **53** (2020), no. 2, 11032–11037.

[244] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H Johansson, *A survey of distributed optimization*, Annual Reviews in Control **47** (2019), 278–305.

[245] L. Yingzhao and C. Jones, *On Gaussian process based Koopman operator*, Ifac world congress, 2020.

[246] A. Zanelli, A. Domahidi, J Jerez, and M. Morari, *Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs*, International Journal of Control **93** (2020), no. 1, 13–29.

[247] A. Zanelli, R. Quirynen, and M. Diehl, *Efficient zero-order NMPC with feasibility and stability guarantees*, IEEE european control conference (ECC), 2019, pp. 2769–2775.

[248] M. Zanon, S. Gros, H. Wymeersch, and P. Falcone, *An asynchronous algorithm for optimal vehicle coordination at traffic intersections*, IFAC-PapersOnLine **50** (2017), no. 1, 12008–12014.

[249] A. Zare, H. Mohammadi, N. K Dhingra, T. T Georgiou, and M. R Jovanović, *Proximal algorithms for large-scale statistical modeling and sensor/actuator selection*, IEEE Transactions on Automatic Control **65** (2019), no. 8, 3441–3456.

[250] K. Zhang, B. Hu, and T. Basar, *Policy optimization for H_2 linear control with H_∞ robustness guarantee: Implicit regularization and global convergence*, Learning for dynamics and control, 2020, pp. 179–190.

[251] Y. Zhang and M. M Zavlanos, *Distributed off-policy actor-critic reinforcement learning with policy consensus*, IEEE conference on decision and control (cdc), 2019, pp. 4674–4679.

[252] M. Zhu and S. Martínez, *Discrete-time dynamic average consensus*, Aut. **46** (2010), no. 2, 322–329.

[253] I. Ziemann, A. Tsiamis, H. Sandberg, and N. Matni, *How are policy gradient methods affected by the limits of control?*, 2022 IEEE 61st conference on decision and control (cdc), 2022, pp. 5992–5999.