Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (37^{th}cycle)

Optimization methods for dynamical system learning

By

Simone Pirrera

******

Supervisor(s):
Prof. Diego Regruto, Supervisor
Prof. Sophie M. Fosson, Co-Supervisor

Doctoral Examination Committee:
Ph.D. Daniele Astolfi, Referee, CNRS Researcher, LAGEPP, University of Lyon 1
Prof. Emiliano Dall'Anese, Referee, Boston University (College of Engineering)
Prof. Andrea Calimera, Politecnico di Torino, DAUIN
Prof. Antonello Giannitrapani, Università di Siena, DIISM
Prof. Carlo Novara, Politecnico di Torino, DET

Politecnico di Torino
2025

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Simone Pirrera
2025

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Dedicated to my loving parents and Jennifer

Acknowledgements

I express my deepest gratitude to my supervisors, Prof. Diego Regruto and Prof. Sophie Fosson, for their guidance and support throughout these three years. I also thank our research group leader, Prof. Vito Cerone, for his careful advice.
Special thanks to my family and my girlfriend, Jennifer Fera, for their unwavering encouragement.
I am also grateful to all Laboratory 11 mates for their collaboration and friendship.

Abstract

System identification (SI) is the discipline of learning dynamical systems from experimental data. This thesis considers the simulation error minimization (SEM) approach to SI and aims to develop algorithms for its solution. In this context, standard solutions are based on applying gradient-based optimization methods. However, these methods may fail due to phenomena known as the vanishing and exploding gradients issues, which arise due to the peculiar structure of the optimization problem. In this thesis, we aim to address these issues by proposing a constrained optimization approach.

To solve the formulated constrained optimization problem, we concentrate on developing novel, efficient, first-order algorithms. To this end, we introduce the controlled multipliers optimization (CMO) framework, which reformulates the optimization problems as control problems, allowing us to utilize controller design to derive solutions. Within this framework, we present three distinct optimization algorithms for equality and inequality constrained optimization, we analyze their convergence, and validate their effectiveness on numerical examples.

Ultimately, we apply one of the proposed algorithms to SEM-based identification. Our approach's efficacy is validated through various benchmark tests and simulation experiments, revealing significant improvements in SI tasks, including black-box SI through artificial neural networks and gray-box problems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

System identification (SI) is the science of building mathematical models that describe physical or artificial systems and phenomena.
The general paradigm of SI is as follows. Firstly, we collect experimental data on the system; usually, these data are affected by noise. Secondly, we select a model class, i.e., a family of models to which we assume the system belongs. Then, we estimate the model's parameters by solving a suitably defined optimization problem. Finally, we validate the model using a data set not used during the estimation.

SI is crucial in various engineering applications. Depending on the context, the identified model is used for prediction, simulation [1, 2], controller design [3, 4], decision-making, and data filtering and denoising [5], among many others. Another increasingly widespread application of SI is direct data-driven control (DDDC) design. A popular approach to DDDC is recasting the controller design problem into the problem of identifying the controller directly from data, given performance specifications. Examples of DDDC methods following this approach are correlation-based tuning [6], virtual reference feedback tuning [7, 8], and methods based on set-membership identification [9, 10].

Most approaches to SI are based on the solution to a suitable optimization problem aiming to minimize prediction or simulation error. Mathematically, this consists of the solution to the optimization problem

$$\min_{\theta \in \mathbb{R}^{n_\theta}} \mathscr{L}(\theta | u, \tilde{y}), \tag{1.1}$$

where $u_t \in \mathbb{R}^q, \tilde{y}_t \in \mathbb{R}^p$ are the available input and output data, respectively, and $\mathscr{L} : \mathbb{R}^{n_\theta} \to \mathbb{R}$ accounts for the error. The definition of the cost function $\mathscr{L}$ depends on several aspects. Firstly, it depends on the considered error model and the norm used to assess it. Secondly, it depends on the considered model class. Finally, we can use different regularization terms to penalize the model's complexity.

There are several options available when it comes to selecting a model class. First, we can classify models as either state-space or input-output models. Next, we can differentiate models based on their structure, i.e., the functional relationship they define between input and output. The simplest structure is that of linearly parameterized models, where the model is a linear combination of specified functions. Alternatively, the model structure can be determined from physical equations; in this case, we deal with gray-box models. Another popular approach to defining the model structure is using flexible function approximators, such as neural networks. The latter approach strongly connects classical SI and modern machine learning. Indeed, objects like recurrent neural networks are particular nonlinear state-space models. In this context, the terms "learn" and "identify" are synonymous, and we will use them interchangeably in this thesis.

Most approaches to SI rely on prediction error minimization (PEM). According to PEM, we identify the model parameters by minimizing the difference between the measured output and the one-step-ahead prediction provided by the model. Considering linearly parametrized models, the PEM problem is convex and can be solved efficiently. Instances of problems of this kind are, e.g., ARX identification [1] and kernel methods [11]. Instead, when the model structure is nonlinear in the parameters, the identification problem (1.1) is non-convex, and a tractable solution often requires looking for local minimizers. Moreover, PEM methods enjoy consistency (i.e., convergence of the estimate to the actual parameter value as the number of data tends to infinity) under the crucial assumption that the selected system model class and the noise model are correct. See, e.g., [1]. However, if these assumptions are not satisfied, PEM methods may yield models that exhibit low accuracy when used to simulate the system. See [12] and [13] for a detailed discussion.

A viable alternative is to formulate the parameter estimation problem as the minimization of the simulation error, i.e., we define $\mathscr{L}$ in (1.1) as

$$\mathscr{L}(\theta) = \sum_{t=1}^{N} \|y_t(\theta|u) - \tilde{y}_t\|_p \tag{1.2}$$

where $y_t(\theta|u)$ is the output of the model when the measured input u is provided, and $p = 1, 2$, or ∞. This approach is known as *simulation error minimization* (SEM) and leads to consistent estimates regardless of the measurement noise model; see [14] and [15]. A noteworthy case is the SEM problem for linear systems. In such a scenario, the optimization problem is polynomial, and semidefinite relaxation techniques are adopted to compute the global solution; see, e.g., [16] for theoretical details and [17] for an illustrative example. Conversely, in the general case of nonlinear systems, the SEM problem is a generic differentiable non-convex minimization problem for which we can compute local solutions only.

Several possibilities to minimize (1.2) are available. The most commonly considered approach is applying gradient-based optimization algorithms. Common choices include first-order gradient descent (GD) algorithms such as stochastic and mini-batch GD, Nesterov's, RMSprop, and Adam algorithms. See, e.g., [18] and [19]. Alternatively to GD, we have second-order methods. These methods present a theoretically guaranteed improvement in the convergence rate compared to first-order algorithms. Nevertheless, they are computationally expensive, and their complexity does not scale well for large problems because they require computing large Hessian matrices. A typical example is the Newton's method. To address these limitations, methods utilizing Hessian approximation have emerged. Among them, we mention Gauss-Newton, Levenberg-Marquardt (LM) [20], and Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithms [21], [22], and [23]. We refer to these methods as quasi-second order.

When using gradient-based optimization algorithms, the problem of evaluating the gradients $\nabla_\theta y_t(\theta)$ emerges. The problem's dynamic structure generates a dynamic relation between the variables $\nabla_\theta y_t(\theta)$. The technique known as dynamic backpropagation, proposed in [24], leverages this dependence to evaluate the gradients as the output of a dynamical system called the sensitivity model.
A more popular alternative is the backpropagation through time (BPTT) algorithm; see, e.g., [25] and [26]. In BPTT, the dynamic dependencies are utilized implicitly to compute partial derivatives recursively. Specifically, BPTT relies on unrolling, i.e., expanding the recurrent model over time into a series of feedforward models. In this framework, the standard choice for the computation of the derivatives is using automatic differentiation (see, e.g., [27] for a recent survey).

Solving SEM problems using gradient-based algorithms is generally associated with slow convergence, numerical convergence to non-optimal solutions, or even instability. The root of these issues lies in phenomena known as vanishing and exploding gradient issues; see, e.g., [28] and [29]. These issues arise independently of the method used to compute the gradient. Indeed, the dependence of $\nabla_\theta y_t$ from $\nabla_\theta y_{t-1}$, for all $t = n+1,\dots,N$, implicitly defines a dynamical system whose outputs are the required gradients. If such a system is a contraction, i.e., it causes a magnitude decrease along time, $\nabla_\theta y_t$ becomes vanishingly small for large t. On the other hand, if it is expansive, the norm of $\nabla_\theta y_t$ diverges. Through Example 1.0.1, we highlight the phenomenon's origin to illustrate this central issue.

Example 1.0.1. (Illustrative example) *Consider the first-order linear dynamical system*

$$y_t = \theta_1 y_{t-1} + \theta_2 u_{t-1}, \qquad y_1 = 0. \tag{1.3}$$

Then, the generic sample at time t is given by

$$y_t = \sum_{k=1}^{t-1} \theta_1^{k-1} \theta_2 u_{t-k}. \tag{1.4}$$

The minimization of the simulation error's ℓ_2 norm (i.e., energy) leads to defining the cost function

$$\mathscr{L}(\theta) = \sum_{t=2}^{N} (y_t - \tilde{y}_t)^2 = \sum_{t=2}^{N} \left(-\tilde{y}_t + \sum_{k=1}^{t-1} \theta_1^{k-1} \theta_2 u_{t-k} \right)^2 \tag{1.5}$$

which have gradient

$$\nabla_\theta \mathscr{L} = 2 \sum_{t=2}^{N} (y_t - \tilde{y}_t) \nabla_\theta y_t \tag{1.6}$$

where

$$\nabla_\theta y_t = \begin{bmatrix} \frac{dy_t}{d\theta_1} \\ \frac{dy_t}{d\theta_2} \end{bmatrix} = \sum_{k=1}^{t-1} \begin{bmatrix} (k-1)\theta_1^{k-2} \theta_2 u_{t-k} \\ \theta_1^{k-1} u_{t-k} \end{bmatrix}. \tag{1.7}$$

Notice that if, at a generic iteration, the optimization variable θ_1 is such that $|\theta_1| < 1$, then the summation terms corresponding to large k are small. We notice that for each time t, only a few terms of the sum (1.7) contribute significantly.

Conversely, if $|\theta_1| > 1$*, the magnitude of the terms in* (1.7) *increases exponentially with* k*.*

Exploding gradient issues may be handled effectively by techniques such as gradient clipping [30]. On the other hand, effectively and efficiently learning arbitrary nonlinear systems without encountering vanishing gradient issues is an open and challenging problem. In particular, since the vanishing gradient leads to slow convergence and inaccurate estimation, alternative methods avoiding the vanishing gradient issue are required to improve the time and energy needed to identify an accurate model, thus increasing the SI approach's scope of applicability.

A relevant sub-class of SI problems that has garnered recent interest is related to recurrent neural networks (RNNs); see, e.g., [31] and [19]. Known for their capability to approximate any dynamical system, RNNs are now widely employed to perform nonlinear black-box identification [32, 33]. Early works on RNN models led to the formulation of the Elman RNNs and neural state-space models, that are classically identified using BPTT. In this context, the classical approach to attenuate the vanishing gradient problem is to include a direct gradient propagation path in the network definition. This idea led to the definition of different RNN structures like long-short-term memory (LSTM) and gated recurrent units (GRU) networks. See, e.g., [19, 34] for a detailed discussion. However, such an approach leads to the definition of large networks that are prone to overfitting and less data-efficient than classical RNNs. Moreover, common regularization strategies that mitigate overfitting may fail when applied to such networks [35].

Despite the practical advantages of LSTM and GRU against classical RNNs, the main idea behind how they attenuate the vanishing gradient issue is based on modifying the system's model; therefore, we cannot directly employ this approach to learn general dynamical models, e.g., gray-box and physics-informed models. This observation motivates the need for alternative approaches to tackle the vanishing gradient problem. Recent contributions in this direction explore defining an identification algorithm that does not require the computation of the gradients $\nabla_\theta y_t$. This approach can avoid vanishing and exploding gradient problems instead of attenuating them. Recent works along this direction include:

- application of meta-heuristic global optimization, e.g., particle swarm optimization or genetic algorithms. See, e.g., [36] and [37]. Although these methods completely avoid any gradient computation, they are generally very

slow when estimating many parameters. Moreover, they lack theoretical convergence guarantees.

- [38], where the authors propose considering a constrained optimization formulation of the problem and applying approximated sequential quadratic programming (SQP) to solve it. This approach effectively reduces the number of iterations required for convergence while the cost of each iteration increases.

In this thesis, we consider SEM identification of nonlinear input-output models. Specifically, we formulate the identification problem by defining a constrained optimization problem in which optimization variables represent model parameters and noise-free outputs, while constraints define the relation between them; i.e., we consider

$$\begin{aligned} &\underset{\theta\in\mathbb{R}^{n_\theta},Y\in\mathbb{R}^{pN}}{\arg\min} \quad \mathscr{L}(\theta,y) \\ &\text{s.t.} \\ &h_{t-n}(\theta,y) = -y_t + \mathscr{M}(\theta,y|u) = 0, \\ &\quad t = n+1,\dots,N, \end{aligned} \tag{1.8}$$

where $Y = [y_1^\top,\dots,y_N^\top]^\top \in \mathbb{R}^{pN}$, $\mathscr{M} : \mathbb{R}^{n_\theta} \times \mathbb{R}^{pN} \to \mathbb{R}^p$ represents the model of the system and $\mathscr{L} : \mathbb{R}^{n_\theta} \times \mathbb{R}^{pN} \to \mathbb{R}$ accounts for the simulation error. A similar mathematical formulation is considered in [38], where the authors develop an inexact SQP method for training neural state-space models. Moreover, constrained optimization problems of this kind are also formulated in the context of set-membership identification, in which we typically assume that noise is bounded and incorporate additional inequality constraints accordingly. See, e.g., [39, 16].

Let us denote $x = [\theta^\top, y_1^\top,\dots,y_N^\top]^\top \in \mathbb{R}^{n_\theta+pN}$ the optimization variables of Problem (1.8). This problem is characterized by the first-order optimality conditions

$$\begin{aligned} &\nabla_x \mathscr{L}(x) + \sum_{t=1}^{N-n} \lambda_t \nabla_x h_t(x) = 0 \\ &h_t(x) = 0, \qquad t = 1,\dots,N-n. \end{aligned} \tag{1.9}$$

Equation (1.9) defines the stationary points of the problem and represents a set of necessary (but generally not sufficient) conditions for the local optimality of x. Notice that this condition only requires computing gradients of simple functions that do not recursively depend on expressions evaluated at previous time instants.

Consequently, the numerical difficulties related to the vanishing gradient are absent if we use this formulation. In Example 1.0.2, we illustrate how this casts to the same identification problem we considered in Example 1.0.1.

Example 1.0.2 (Illustrative example, cont.)**.** *Consider the system* (1.3)*. According to* (1.8) *and* (1.9)*, KKT conditions for the constrained formulation of the identification problem are*

$$\nabla_\theta \mathscr{L} + \sum_{t=n+1}^{N} \lambda_{t-n} \nabla_\theta h_{t-n} = \begin{bmatrix} \sum_{t=2}^{N} \lambda_{t-1} y_{t-1} \\ \sum_{t=2}^{N} \lambda_{t-1} u_{t-1} \end{bmatrix} = 0 \tag{1.10}$$

$$\begin{aligned} &\nabla_y \mathscr{L} + \sum_{t=n+1}^{N} \lambda_{t-n} \nabla_y h_{t-n} = 0 \iff \\ &2(y_t - \tilde{y}_t) + \lambda_{t-1} - \lambda_t \theta_1 = 0, \quad \forall t = 1, \dots, N, \end{aligned} \tag{1.11}$$

where we defined $\lambda_N = 0$ *to ease the notation and keep it consistent, and*

$$h_{t-n} = y_t - \theta_1 y_{t-1} - \theta_2 u_t = 0, \quad \forall t = 2, \dots, N. \tag{1.12}$$

Compared with Example (1.0.1)*, the number of optimization variables increases from* 2 *to* $2+N$*, and the number of equations to be solved increases from* 2 *to* $2N+1$*. Nonetheless, all the equations are now simple bilinear functions of the optimization variables. This formulation does not require computing the powers of* θ_1*, which causes the vanishing gradient issue.*

The illustrative Example 1.0.2 highlights that we can avoid vanishing gradient issues by solving a constrained optimization problem. Nevertheless, this task is challenging from a computational viewpoint.

Standard approaches to constrained optimization include interior-point methods, sequential quadratic programming, and Lagrangian methods; see [40] and [41]. All these approaches require solving large systems of linear equations, which may become computationally- and memory-prohibitive when the number of data (and consequently, constraints) increases.

This consideration motivates the need for an efficient equality-constrained optimization algorithm. For this purpose, this thesis aims to develop novel first-order, memory-efficient algorithms for constrained optimization. Specifically, we propose

an original approach based on feedback control theory, which we call *controlled multipliers optimization* (CMO) framework.

CMO involves reformulating an assigned constrained optimization problem into an equivalent stabilization and output regulation control problem. We define a fictitious plant where first-order optimality conditions define the state equations, the constraints define the output, and the Lagrange multipliers are interpreted as the control input. We demonstrate that by finding appropriate control laws, we can define a feedback control system that drives the plant's state trajectories to converge toward the optimization problem solution.

We focus on equality-constrained optimization problems and present two solutions: one based on proportional-integral (PI) control [42], and the other based on feedback linearization (FL) [43]. We refer to the resulting algorithms as PI-CMO and FL-CMO, respectively. Our theoretical analysis demonstrates that both algorithms achieve global exponential convergence to the global optimal solution to the optimization problem when the objective function is convex and the constraints are linear. Additionally, we conduct a local analysis of FL-CMO to establish the asymptotic stability of isolated minima for non-convex problems.
Next, we address optimization problems that are subject to linear inequality constraints. These problems often arise in set-membership identification, particularly when the noise is assumed to be bounded in magnitude. We propose a solution based on the CMO framework, considering a plant defined using the augmented Lagrangian and a modified PI controller; we call this algorithm the modified PI-CMO (M-PI-CMO). We show that this algorithm is globally exponentially convergent for convex optimization problems.
Finally, we compare our approaches with similar ones in the literature. Specifically, we compare PI-CMO and M-PI-CMO against primal-dual gradient dynamics (PDGD). Both theoretical and simulation results indicate that PI-CMO and M-PI-CMO converge faster than PDGD. Furthermore, FL-CMO is compared to null-space gradient dynamics, demonstrating that FL-CMO generalizes the latter.

In the second part of the thesis, we apply the proposed FL-CMO optimization algorithm to the problem of identifying nonlinear input-output (NIO) models. Specifically, we consider a rather general class of linear or nonlinear models in the form

$$y_t = \mathscr{M}(y_{t-1}, \ldots, y_{t-n}, u_t, \ldots, u_{t-n}|\theta), \tag{1.13}$$

where $u_t \in \mathbb{R}^q$ is the input, $y_t \in \mathbb{R}^p$ is the output, and

$$\mathscr{M} : \underbrace{\mathbb{R}^p \times \cdots \times \mathbb{R}^p}_{n\ times} \times \underbrace{\mathbb{R}^q \times \cdots \times \mathbb{R}^q}_{n+1\ times} \to \mathbb{R}^p \tag{1.14}$$

is any differentiable function. The considered model class can account for a large array of models, including linear, block-structured, gray-box, physics-informed, and neural networks models.

We formulate the identification problem as a constrained optimization of the kind (1.8) and demonstrate that the structure of the problem is compatible with the assumptions required to apply FL-CMO. Next, we prove that it converges to a stationary point of the original unconstrained formulation (1.1). To cope with the main computational bottleneck of the algorithm, we leverage the sparsity of the constraints' Jacobian to propose a Q-less QR factorization procedure that allows us to reduce the computational complexity from cubic to quadratic in the number of identification data samples. Moreover, we optimize the amount of required iterations by using adaptive step size to integrate the original, continuous-time formulation of FL-CMO.

In addition to introducing the algorithm and performing its theoretical analysis, we demonstrate the effectiveness of the proposed approach on several SI problems. We consider four key problems: established black-box SI benchmarks and a realistic gray-box identification problem. Concerning the black-box problems, we consider the *nonlinear neural output error* (NNOE) model, which is a particular kind of RNN in input-output form. The results indicate that, on the one hand, our approach significantly outperforms standard methods for training NNOE models. On the other hand, a properly trained NNOE model can provide superior performances compared to prevalent RNN models, including Elman RNN, LSTM, and GRU. We explain the latter by noting that NIO models are intrinsically more data-efficient than their state-space counterparts. Concerning gray-box identification, we show that the proposed method significantly improves the parameter estimates compared to standard gradient approaches.

Finally, as a side project, we also devote our attention to the identification of linear-time-invariant systems. We formulate the SEM problem for this class of models as constrained optimization, and we notice that the formulated problem is described by polynomials. This observation motivates the need to develop novel and

fast algorithms for polynomial optimization. We address this problem by proposing a solution based on the alternating direction method of multipliers (ADMM) algorithm, which we call ADMM4POP. We theoretically analyze the algorithm's convergence and validate the effectiveness of the proposed approach through numerical examples.

1.1 Outline of the thesis

This thesis is organized into three parts.

1.1.1 Part I: Background

Part I provides a background of the required preliminary results needed in the subsequent chapters.

In Chapter 2, we deal with the *nonlinear SI problem*. First, we introduce the general SI problem, review the relationship between state-space and input-output models, and discuss the differences between commonly considered loss functions used to define the problem. Then, we provide an overview of SI methods, focusing on linear, gray-box, block-oriented, and neural network models.

In Chapter 3, we recall definitions and standard results on *nonlinear systems* stability and control. Specifically, we first deal with notions of stability and review Lyapunov's stability criteria; then, we review feedback linearization controller design for nonlinear systems.

Chapter 4 reviews basic notions and fundamental results related to *optimization problems*. We consider constrained and unconstrained optimization problems and review results on optimality conditions. Next, we review some optimization algorithms with a particular emphasis on gradient and Lagrangian methods. Finally, we review some results related to dynamical systems approaches to the analysis of optimization algorithms' convergence.

1.1.2 Part II: Constrained optimization through control

Part II is the core of the thesis. It introduces the proposed *controlled multipliers optimization* (CMO) approach to design optimization algorithms through control theory.

In Chapter 5, we introduce the main proposed *framework* by defining the fictitious plant to be controlled and proving two fundamental lemmas: one for equality-constrained problems and another for inequality-constrained ones.

Chapter 6 proposes an optimization algorithm based on *PI control* (PI-CMO) for equality-constrained optimization. We prove its convergence for strongly convex optimization problems and compare it to primal-dual gradient dynamics (PDGD).

In Chapter 7, we consider optimization problems with *linear inequality* constraints and propose a solution based on a modification of the PI control law (modified PI-CMO). We theoretically analyze its convergence for convex problems and compare it with the augmented Lagrangian PDGD (Aug-PDGD).

In Chapter 8, we develop a solution based on *feedback linearization* controller design (FL-CMO). We prove convergence for both convex and non-convex optimization problems and demonstrate the practical effectiveness of the approach through numerical experiments.

This part is partially based on the papers:

- [44]. V. Cerone, S. M. Fosson, S. Pirrera, D. Regruto, "A new framework for constrained optimization via feedback control of Lagrange multipliers.", submitted to IEEE Transactions on Automatic Control, 2024, available at https://arxiv.org/abs/2403.12738.
- [45]. V. Cerone, S. M. Fosson, S. Pirrera, D. Regruto, "A feedback control approach to convex optimization with inequality constraints," In 63rd IEEE Conference on Decision and Control (CDC), 2024, available at https://arxiv.org/abs/2409.07168.

1.1.3 Part III: Constrained optimization for system identification

Part III deals with applying algorithms for constrained optimization to identify dynamical systems.

In Chapter 9, we consider *nonlinear input-output models* and formulate the identification problem as constrained optimization. Then, we propose a learning algorithm defined starting from FL-CMO, as proposed in Chapter 8. We introduce novel strategies to improve the algorithm's computational cost, and we theoretically study the computational complexity of the iterations, proving that it is quadratic in the number of data. Finally, we present numerical results on benchmarks and selected problems concerning both black-box and gray-box SI.

Chapter 10 is devoted to the *linear SI* problem. In particular, we formulate the linear SI problem as a polynomial optimization problem and propose an algorithm based on ADMM to solve it efficiently. We analyze the convergence of the proposed algorithm and provide numerical examples.

This part is partially based on the papers:

- [46]. V. Cerone, S. M. Fosson, S. Pirrera, D. Regruto, "A constrained optimization approach to system identification of nonlinear input-output models," manuscript in preparation, 2024.
- [17]. V. Cerone, S. M. Fosson, S. Pirrera, D. Regruto, "Alternating direction method of multipliers for polynomial optimization," In European Control Conference (ECC), 2023, available at https://ieeexplore.ieee.org/document/10178190.

Chapter 11 draws the conclusions of the thesis.

1.2 Notation

Throughout the thesis, we use the following notation.
For a symmetric matrix A, $A \succ (\succeq) 0$ indicates that A is positive (semi)definite, while $A \prec (\preceq) 0$ denotes negative (semi)definiteness.
For a function $x(t) : \mathbb{R} \to \mathbb{R}^n$, we define its component-wise derivative with respect

to t as $\dot{x} = \frac{\mathrm{d}}{\mathrm{d}t}x(t) : \mathbb{R} \to \mathbb{R}^n$. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, its gradient is $\nabla_x f(x) : \mathbb{R}^n \to \mathbb{R}^n$, while its Hessian is $\nabla_x^2 f(x) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ or equivalently $H_{xx} f(x) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$. The subscript "$x$" in ∇ and the argument of f may be omitted when clear from context. For a function $g : \mathbb{R}^n \to \mathbb{R}^m$, we denote its Jacobian as $J_g(x) = [\nabla g_1(x), \ldots, \nabla g_m(x)]^\top : \mathbb{R}^n \to \mathbb{R}^{m\times n}$.

Part I

Background

Chapter 2

Introduction to system identification

This chapter introduces the general nonlinear system identification (SI) problem and overview methods in this context. The content of this chapter is mainly based on the books "System Identification: Theory for the User" by L. Ljung [1], "Block-oriented Nonlinear System Identification" by F. Giri & E. W. Bai [47], "Recurrent Neural Networks" by F. M. Salem [19], and on referenced papers.

In SI, the model structure, i.e., the relationship the model defines between input and output, is often selected based on some available physical knowledge of the system to be identified. Depending on the amount of a-priori knowledge, we distinguish between gray-box and black-box SI. If we aim to estimate physically meaningful parameters of a system, we refer to the identification procedure as gray-box SI. If we assume that no physical knowledge is available, we deal with black-box

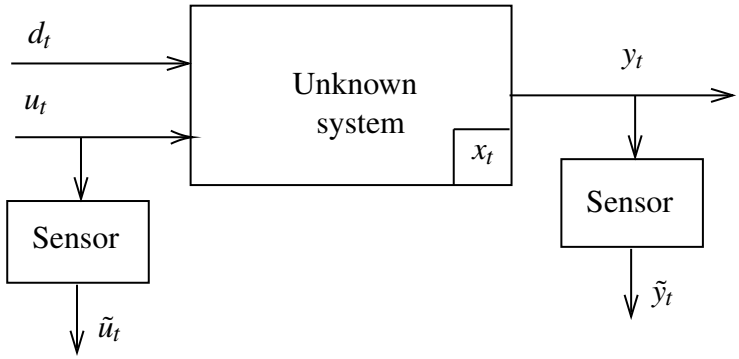

Fig. 2.1 Generic description of an experimental setup

SI. In the latter case, a common choice is to look for universal approximators to define the model structure.

In the following section, we formally define the SI problem and review the results of the relation between state-space and input-output models. Afterward, we review SI methods, from linear SI to block-oriented, gray-box, and neural network models. Other approaches to nonlinear SI that we will not cover include, e.g., Gaussian processes [48] and kernel methods [11].

2.1 Problem formulation

Let P be an unknown dynamical system that we want to identify, and let us assume that P allows for a state-space representation as

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) \\ y_t &= h(x_t, u_t), \end{aligned} \tag{2.1}$$

where $x_t \in \mathbb{R}^n$, n the dynamical order of the system, and $u_t \in \mathbb{R}^q$, $y_t \in \mathbb{R}^p$ denote the system's input and output at time t, respectively.

We assume the available data is a set of input-output measurements $\{u_t, \tilde{y}_t\}$, for $t = 1, \dots, N$. In a general setting, data are collected according to the diagram in Figure 2.1: both input and output are measured by means of sensors, which potentially introduce noise and uncertainty. Moreover, some unmeasurable input disturbances d_t may act on the system: it is usually assumed that the effect of d_t on the outputs is small.

Throughout this thesis, unless otherwise stated, we assume that the input samples u_t are unaffected by noise, while the output samples are corrupted by a noise sequence η_t according to $\tilde{y}_t = y_t + \eta_t$. In this case, η_t models the joint effect of measurement uncertainty and unmeasured input disturbances. This is referred to as the *output error* noise structure.

2.1.1 State-space and input-output models

This section investigates the differences and similarities between state-space models and input-output ones. We review under what conditions we can equivalently write a system in the state-space form (2.1) as a system in the regressor form representation

$$y_t = \phi(y_{t-1}, \dots, y_{t-n}, u_t, \dots, u_{t-n}), \tag{2.2}$$

where $\phi(\cdot)$ is a generic differentiable function.

To obtain the input-output behavior of (2.1), we resort to the following procedure, which is the discrete-time counterpart of the development in, e.g., [49] and [50, Chapter 2].

First, we iteratively derive the dependency of y_{t+j} from initial state x_0 and the input, for $j = 1, \dots, n$ as

$$\begin{aligned}
y_t &= h(x_t, u_t) && \text{(2.3a)}\\
y_{t+1} &= h(x_{t+1}, u_{t+1}) = h(f(x_t, u_t), u_{t+1}) \doteq h_1(x_t, u_t, u_{t+1}) && \text{(2.3b)}\\
y_{t+2} &= h(x_{t+2}, u_{t+2}) = \cdots \doteq h_2(x_t, u_t, u_{t+1}, u_{t+2}) && \text{(2.3c)}\\
&\vdots \\
y_{t+n-1} &= \cdots = h_{n-1}(x_t, u_t, \dots, u_{t+n-1}) && \text{(2.3d)}\\
y_{t+n} &= \cdots = h_n(x_t, u_t, u_{t+1}, \dots, u_{t+n}). && \text{(2.3e)}
\end{aligned}$$

Equations (2.3) is a set of n equations which express the output samples $y_t, \dots, y_{t+n-1}$ as a function of the input samples $u_t, \dots, u_{t+n-1}$ and the initial state x_t only.

Recall now the following definitions of observability.

Definition 2.1.1. (Distinguishable, [50, Chapter 4])

Given a system (2.1), two states $x_1, x_2 \in \mathbb{R}^n$ are said to be indistinguishable, if for all $T \geq 0$ and any sequence of control inputs $\bar{u}_T = u_t, \dots, u_{t+T}$, we have

$$y_{t+T}(x_1, \bar{u}_T) = y_{t+T}(x_2, \bar{u}_T). \tag{2.4}$$

If x_1, x_2 are indistinguishable, we write $x_1 I x_2$.

Definition 2.1.2. (Local weak observability, [50, Chapter 4])

Given a system (2.1), a state $x_1 \in \mathbb{R}^n$ is said to be locally weak observable if there exists a neighbor V of x_1, such that for any $x_2 \in U$, with $U \subseteq V$ open neighbor of x_1, it holds

$$x_1 I x_2 \Rightarrow x_1 = x_2 \qquad \forall x_1, x_2 \in N(x_0). \tag{2.5}$$

The following discussion will refer to local weak observability simply as "observability." Observability can be intuitively understood as the property of a system that enables the reconstruction of its state based on knowledge of the model, the input, and the output. Several control and identification problems strongly rely on the assumption of observability. Among them, we mention the observer design problem [51] and the problem of characterizing the identifiability of state-space models [52].

A classical result on observability is the following.

Theorem 2.1.1. (Characterization of observability, [50, Chapter 4] and [53])

The system (2.1) is observable if and only if

$$rank\left(\frac{\partial [h_1, \dots, h_{n-1}]}{\partial x}\right) = n. \tag{2.6}$$

Notice that, in the case of linear systems, this condition reduces to the classical rank condition by Kalman [51].

As a consequence of Theorem 2.1.1, if the system is observable, we can solve the system of equations (2.3a)-(2.3d) for x_t, i.e., we can define the map Φ^{-1} such that

$$x_t = \Phi^{-1}(y_t, \dots, y_{t+n-1}, u_t, \dots, u_{t+n-1}). \tag{2.7}$$

The existence of this function allows us to rewrite Equation (2.3e) as

$$\begin{aligned} y_{t+n} &= h_n(x_t, u_t, u_{t+1}, \dots, u_{t+n}) = \\ &= h_n(\Phi^{-1}(y_t, \dots, y_{t+n-1}, u_t, \dots, u_{t+n-1}), u_t, \dots, u_{t+n}) = \\ &= \phi(y_t, \dots, y_{t+n-1}, u_t, \dots, u_{t+n}). \end{aligned} \tag{2.8}$$

This expression is formally the same as (2.2). Thus, we conclude the following theorem (which is the discrete-time counterpart of [50, Theorem 2.2.1]).

Theorem 2.1.2. (State elimination)

Let S be a system of the kind (2.1). If S is locally observable, then there exists an open subset V where it admits an input-output representation (2.2).

As a consequence of this result, under the mild assumption of observability, state-space models and input-output ones are equally powerful in representing dynamical systems. Moreover, the procedure developed in this section allows us to derive an input-output model for any given state-space model explicitly.

2.1.2 Cost functions

This section reviews the most commonly considered loss functions used in SI. Specifically, we consider input-output models described by

$$y_t = \mathscr{M}(\phi_t, \theta), \quad \phi_t = [y_{t-1}^\top, \ldots, y_{t-n}^\top, u_t^\top, \ldots, u_{t-n}^\top]^\top, \tag{2.9}$$

and review the definitions for one-step-ahead prediction and simulation error.

We define the one-step-ahead prediction error as

$$e_t = \mathscr{M}(\tilde{\phi}_t, \theta) - \tilde{y}_t, \tag{2.10}$$

where $\tilde{\phi}_t = [\tilde{y}_{t-1}^\top, \ldots, \tilde{y}_{t-n}^\top, u_t^\top, \ldots, u_{t-n}^\top]^\top$ is the regressor built using the collected noisy data. If $\mathscr{M}$ is linear in the parameters θ, then minimizing any norm of the one-step-ahead prediction error is a convex optimization problem and can be solved efficiently. We can interpret the one-step-ahead prediction error as equation error noise. Such noise structure is often introduced to model the effect of unmeasured input disturbances or the effect of unmodeled dynamics.

The simulation error is defined as

$$e_t = y_t - \tilde{y}_t, \tag{2.11}$$

where

$$y_t = \tilde{y}_t, \quad t = 1, \ldots, n \tag{2.12a}$$

$$y_t = \mathscr{M}(\phi_t, \theta), \quad t = n+1, \ldots, N, \tag{2.12b}$$

N is the number of available data, and ϕ_t is defined according to Equation (2.9). Alternatively, we can consider $y_1, \dots, y_n$ as additional optimization variables.
The simulation error is the difference between the measured output and the output variables produced by the model; thus, it can account for unmeasured input disturbances, unmodeled dynamics, and measurement noise.

2.2 Overview of SI methods

2.2.1 Linear SI

In linear SI, the model is a linear time-invariant (LTI) dynamical system, i.e.,

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t, \end{aligned} \tag{2.13}$$

or equivalently

$$y_t + \sum_{j=1}^{n} \alpha_j y_{t-j} = \sum_{j=0}^{n} \beta_j u_{t-j} \tag{2.14}$$

for some $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times q}, C \in \mathbb{R}^{p\times n}, D \in \mathbb{R}^{p\times q}$ and $\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^{n+1}$.

Among methods for linear SI, we mention prediction error methods (PEM) [1], set-membership (SM) identification [54], and subspace identification [55].

2.2.2 Gray-box identification

In a gray-box SI problem, the goal is to estimate the physical parameters of a system. Gray-box models may be given in continuous- or discrete-time, state-space, or input-output form.

Unlike black-box SI, this problem may be challenging even if the considered system is LTI and we consider one-step-ahead prediction error minimization. The difficulty arises from the fact that the prediction model may depend on the physical parameters through arbitrarily complicated relations, thus leading, in general, to non-convex optimization problems. See, e.g., [56] and [57].

According to [58], when the selected model structure is nonlinear, the separation between black-box and gray-box models is less sharp. Depending on the amount of physical knowledge used to construct the model of the system, the procedure may be a combination of the gray- and black-box approaches. When the parameters have physical meaning, we deal with physics-based models, and the identification is a pure gray-box problem. If the physical information only provides a set of meaningful nonlinearities, then the identification procedure is considered partially gray-box and partially black-box. In this case, block-oriented models and basis function expansion (which we shall review in the following sections) are viable approaches.

2.2.3 Block-oriented models

Block-oriented models are nonlinear systems described as the interconnection of LTI and static nonlinear blocks. The blocks can be combined according to several possible structures: Wiener and Hammerstein models belong to this category. Several approaches are available to perform their identification. The best linear approximation paradigm [59] proposes starting by identifying a linear system approximation and then introducing the nonlinearities. In the set-membership approach [60, 61], we characterize the set of all systems compatible with the system structure and the data. Other approaches include iterative and frequency methods; see, e.g., [47].

2.2.4 Basis function expansion

Given a nonlinear model in input-output form

$$y_t = f(\phi_t), \qquad \phi_t \doteq [y_{t-1}, \dots, y_{t-n}, u_t, \dots, u_{t-n}]^\top, \tag{2.15}$$

the basis function expansion approach consists of representing the function $f(\cdot)$ as a linear combination of known basis functions $\beta_j(\phi_t)$, i.e,

$$y_t = \sum_{j=1}^{B} \theta_j \beta_j(\phi_t). \tag{2.16}$$

Usually, the functions β_j are taken as orthogonal elements of some Hilbert functions space. More specifically, the following are common choices for the selection of the functions β_j:

- Physical insight on the system. In this case, the identification is a gray-box problem.
- Polynomial functions. A possibility to select β_j is using multivariate polynomials; this choice is motivated by the Weierstrass theorem, which states that in any compact set, polynomials are universal approximators for continuous functions.
- Radial basis functions. Radial basis functions are functions whose value depends only on the distance between the input and some fixed point. A typical example is the Gaussian radial function

$$\beta_j(\phi_t) = e^{-c_j \left\| \phi_t - \bar{\phi}_j \right\|^2}, \tag{2.17}$$

  where $c_j, \bar{\phi}_j$ are fixed choices.

When minimizing the one-step-ahead prediction error, the identification problem

$$\underset{\theta \in \mathbb{R}^{n_\theta}}{\arg\min} \sum_{t=n+1}^{N} \left(\tilde{y}_t - \sum_{j=1}^{B} \theta_j \beta_j(\tilde{\phi}_t) \right)^2, \tag{2.18}$$

where $\tilde{\phi}_t \doteq [\tilde{y}_{t-1}, \ldots, \tilde{y}_{t-n}, u_t, \ldots, u_{t-n}]^\top$, is a standard least squares problem and can be solved efficiently. Instead, if we aim to minimize the simulation error, the problem will be non-convex.

Whatever the choice of β_j, this approach suffers from the so-called cruise of dimensionality, i.e., the number of model parameters grows combinatorially with the dimension on the regression vector ϕ. This fact generally leads to overfitting issues.

Sparsification-based methods were suggested to address overfitting and reduce the model complexity. Such approaches include:

- least absolute selection and shrinkage operator (LASSO) [62], which consists of including ℓ_1-norm regularization to the optimization problem;

- sequential thresholded least-squares, used in SINDy [63, 64].
- set-membership approaches, e.g., [65].

2.2.5 Neural networks for SI

Recurrent neural networks (RNNs) are a viable approach to parameterize the system in pure black-box SI. In fact, RNNs are, by definition, instances of state-space models. Moreover, as the number of neurons converges to infinity, they can approximate any system. This potential of RNNs to approximate any system makes them a compelling choice for performing nonlinear SI. Several families of RNNs exist; in the following, we provide a brief overview of methods in this context.

When considering state-space models, we have

- Elmann RNNs. Originally proposed by Elman in [66], Elmann RNNs are defined as the cascade interconnection of simple nonlinear systems. Letting $\sigma(\cdot)$ denote any nonlinear activation function, the L-layer Elman RNN is

$$\begin{aligned} x_t^{(1)} &= \sigma(W_{ux}^{(1)} u_t + W_{xx}^{(1)} x_{t-1}^{(1)} + b^{(1)}) && \text{(2.19a)} \\ x_t^{(2)} &= \sigma(W_{ux}^{(2)} x_t^{(1)} + W_{xx}^{(2)} x_{t-1}^{(2)} + b^{(2)}) && \text{(2.19b)} \\ &\vdots \\ x_t^{(L)} &= \sigma(W_{ux}^{(L)} x_t^{(L-1)} + W_{xx}^{(L)} x_{t-1}^{(L)} + b^{(L)}) && \text{(2.19c)} \\ y_t &= \mathcal{N}(x_t^{(L)}, \theta), && \text{(2.19d)} \end{aligned}$$

  where $W_{ux}^{(\ell)}, W_{xx}^{(\ell)}, b^{(\ell)}$ for $\ell = 1, \dots, L$ and θ are parameters to be estimated and $\mathcal{N}$ is a multi-layer perception (MLP), i.e., $\mathcal{N}(x) = l_1(x) \circ \cdots \circ l_L(x)$ with $l_\ell(x) = \sigma(A_\ell x + b_\ell)$ for $\ell = 1, \dots, L$ and $\theta = [\mathrm{vec}(A_1)^\top, b_1^\top, \dots, \mathrm{vec}(A_L)^\top, b_L^\top]^\top$.

- Nonlinear neural state-space models (NNSS) [67]. A NNSS model is defined by

$$\begin{aligned} x_{t+1} &= \mathcal{N}_x(x_t, u_t, \theta_x) && \text{(2.20a)} \\ y_t &= \mathcal{N}_y(x_t, u_t, \theta_y), && \text{(2.20b)} \end{aligned}$$

where $\mathscr{N}_x, \mathscr{N}_y$ are MLPs and $\theta \doteq [\theta_x^\top, \theta_y^\top]^\top$ is a vector of parameters to be estimated. The rationale behind this definition is straightforward: MLPs are used to approximate both the state and the output equation.

- Long-short-term memory (LSTM) and gated recurrent unit networks (GRU). Originally proposed as a modification of the Elmann RNN to handle the vanishing gradient issue, network structures like LSTM [68, 69] and GRU [70] are now prevalent models in machine learning and SI. These networks were designed to reduce the effect of the vanishing gradient problem. In particular, the main idea is to include a direct propagation path for the gradient in the state equation. In both cases, the network is an extension of the Elmann RNNs where the state propagation equation (called cells) is more involved, i.e., composed of more than one matrix multiplication and nonlinearity. The structure of each cell bases itself on nonlinearities, called gates, that drive the information flow depending on the current state and input.

  Applications of those kinds of networks include, but are not limited to, SI, natural language processing, speech recognition, and economics; see, e.g., [71, 32]. The main drawbacks of such models are the required computational effort for training and the reduced data efficiency (i.e., they are prone to overfitting).

The most popular approach to identify any of those models is to apply gradient-based optimization while computing the derivatives thanks to the *backpropagation through time* (BPTT) algorithm. It is well-known that this approach leads to difficulties in learning from long sequences (long-term dependencies) due to vanishing and exploding gradient issues. See, e.g., [28] and [29] for a detailed discussion.

Advantages of NNSS models, when compared with RNNs, LSTM, and GRU, include the possibility of selecting the order of the system to be identified independently from the number of layers: this allows exploiting a-priori information when available and eases the cross-validation procedure. In fact, NNSS allows us to define the multi-layer architectures to capture strongly nonlinear dynamics and, at the same time, to retain the information that the order of the system is low.

Considering input-output models, the main idea is to approximate the function $f(\cdot)$ defining the unknown input-output mapping

$$y_t = f(y_{t-1}, \dots, y_{t-n}, u_t, u_{t-1}, \dots, u_{t-n}) \tag{2.21}$$

by a MLP. Depending on the considered cost function, we have two main alternatives to estimate the parameters of the network:

1. in case of one-step-ahead prediction error, i.e., we consider the optimization problem:

$$\arg\min_{\theta\in\mathbb{R}^{n_\theta}} \|\tilde{y}_t - \mathscr{N}(\tilde{y}_{t-1},\dots,\tilde{y}_{t-n},u_t,\dots,u_{t-n},\theta)\|_2^2 + \rho(\theta), \tag{2.22}$$

   we refer to the model as to *nonlinear neural autoregressive exogenous* (NNARX) [72, 73]. Identification of NNARX is computationally efficient and classically performed through standard backpropagation. The main disadvantage of NNARX identification is that the identified model often exhibits poor simulation performances.

2. if we minimize the simulation error, i.e., we solve the optimization problem

$$\arg\min_{\theta\in\mathbb{R}^{n_\theta}} \|\tilde{y}_t - y_t(u,\theta)\|_2^2 + \rho(\theta). \tag{2.23}$$

   where $y_k(u,\theta)$ is the output of the model defined recursively as

$$y_t = \tilde{y}_t, \qquad k = 1,\dots,n \tag{2.24}$$

$$y_t = \mathscr{N}(y_{t-1},\dots,y_{t-n},u_t,\dots,u_{t-n},\theta), \qquad k = n+1,\dots,N, \tag{2.25}$$

   we refer to the model as to *output error neural network* (NNOE) [73]. As noted in [12, 74], NNOE can be considered a special kind of RNN, and as shown in [75], NNOEs are computationally as strong as fully connected RNNs. Identification of NNOE is usually performed using BPTT and suffers from the vanishing and exploding gradient phenomena described in Chapter 1.

Chapter 3

Stability and control of nonlinear systems

This chapter reviews basic concepts about nonlinear dynamical systems. Specifically, we consider stability notions and results for autonomous systems and provide an overview of feedback linearization controller design. The material of this chapter is based on the books "Nonlinear Systems" by H. K. Khalil [76], "Nonlinear Control Systems I" and "Nonlinear Control Systems II" by A. Isidori [43],[77], "Applied Nonlinear Control" by J. E. Slotine and W. Li [78] and "Mathematical Control Theory: Deterministic Finite Dimensional Systems" by E. Sontag [79].

3.1 Stability Notions

We consider an autonomous system of the kind

$$\dot{x}(t) = f(x(t)) \tag{3.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is assumed to be continuously differentiable.

Definition 3.1.1. (Equilibrium point, [76],[78, Chapter 3]).

A point $x^* \in \mathbb{R}^n$ such that $f(x^*) = 0$ is said to be an equilibrium point.

In the following, we denote $\mathscr{B}_d(\mathscr{A})$ the open ball of radius d centered in the set $\mathscr{A} \subset \mathbb{R}^n$, i.e., $\mathscr{B}_d(\mathscr{A}) \doteq \{x \in \mathbb{R}^n : \exists y \in \mathscr{A} \text{ s.t. } \|x-y\|_2 \leq d\}$. Moreover, we denote as $\|x\|_{\mathscr{A}}$ the distance of x from the set $\mathscr{A}$, i.e., $\inf_{y\in\mathbb{R}^n} \sup_{x\in\mathscr{A}} \|x-y\|_2$. Notice that, despite what the symbol suggests, $\|x\|_{\mathscr{A}}$ is not a norm.

Definition 3.1.2. ($\varepsilon/\delta_\varepsilon$ stability, [76], [78, Chapter 3]).

The set $\mathscr{A}$ is stable if for all $\varepsilon > 0, \exists \delta_\varepsilon > 0$ s.t. for all $x_0 \in \mathbb{R}^n : \|x_0\|_{\mathscr{A}} \leq \delta_\varepsilon$, $\|x(t,x_0)\|_{\mathscr{A}} \leq \varepsilon$ for all $t \geq 0$.

A system is said to be unstable if it is not stable.

Definition 3.1.3. (Asymptotic stability, [76], [78, Chapter 3]).

The set $\mathscr{A}$ is asymptotically stable (AS) if $\mathscr{A}$ is stable in the $\varepsilon/\delta_\varepsilon$ sense, and there exists a set $\mathscr{D} \subseteq \mathbb{R}^n$ s.t. for all $x_0 \in \mathscr{D}$

$$\lim_{t\to\infty} \|x(t,x_0)\|_{\mathscr{A}} = 0.$$

This property is referred to as attractivity. $\mathscr{D}$ is called domain of attraction. If $D = \mathbb{R}^n$, we have global asymptotic stability (GAS). Otherwise, we have local asymptotic stability (LAS) in $\mathscr{D}$.

Definition 3.1.4. (Exponential stability, [76],[78, Chapter 3]).

The set $\mathscr{A}$ is locally (resp. globally) exponentially stable if $\mathscr{A}$ is LAS (resp. GAS) and exists $c_1, c_2, c_3 \in \mathbb{R}$, $c_3 > 0$ s.t. for all $\|x_0\|_{\mathscr{A}} \leq c_1$

$$\|x(t,x_0)\|_{\mathscr{A}} \leq c_2 e^{-c_3 t} \|x_0\|_{\mathscr{A}}.$$

Next, we provide theorems characterizing the above-defined stability notions. We start with the so-called Lyapunov's linearization method.

Theorem 3.1.1. (Lyapunov's linearization method, [78, Section 3.3]).

Consider the linearized system at the equilibrium point $x^* \in \mathbb{R}^n$, defined as

$$\dot{x} = Ax, \qquad A \doteq \left.\frac{\partial f}{\partial x}\right|_{x^*} \tag{3.2}$$

If the linearized system is strictly stable (i.e., all eigenvalues of A are strictly in the left-half complex plane), the equilibrium point x^* is locally exponentially stable for the actual nonlinear system. Conversely, if the linearized system is unstable (i.e. if at least one eigenvalue of A is strictly in the right-half complex plane), the equilibrium point is unstable.

Notice that if the linearized system is marginally stable (i.e., all eigenvalues of A are in the left-half complex plane, but at least one of them is on the imaginary axis), then one cannot conclude anything about the nonlinear system.

The main drawback of the linearization method is that it is local, and thus, we cannot use it to draw global results. To achieve this goal, we need to consider Lyapunov's direct method. To define Lyapunov functions and the corresponding theorems, we rely on the concepts of $\mathcal{K}$ and $\mathcal{K}_\infty$ functions.

Definition 3.1.5. (Class $\mathcal{K}$ function, [77, Chapter 10]).

A function $\alpha : [0,d) \to \mathbb{R}$, $d > 0$, is a class $\mathcal{K}$ function if it is continuous, $\alpha(0) = 0$, and is strictly increasing.

Definition 3.1.6. (Class $\mathcal{K}_\infty$ function, [77, Chapter 10]).

A function $\alpha : [0,\infty) \to \mathbb{R}$ is a class $\mathcal{K}_\infty$ function if it is class $\mathcal{K}$ with $d = \infty$ and is radially unbounded (i.e., $\lim_{r\to\infty} \alpha(r) = \infty$).

In the remainder of this section, we present the main results on the stability of autonomous dynamical systems.

Theorem 3.1.2. (Lyapunov's theorem for local stability, [77, Chapter 10]).

Suppose there exists a function $V(x) : \mathcal{B}_d(\mathcal{A}) \to \mathbb{R}$ such that there exists $\alpha_l, \alpha_u, \alpha$ class $\mathcal{K}$ functions with domain $[0,d)$ such that

$$\alpha_l(\|x\|_{\mathcal{A}}) \leq V(x) \leq \alpha_u(\|x\|_{\mathcal{A}}) \tag{3.3}$$

$$\dot{V}(x) = \frac{\partial V}{\partial f} f(x) \leq \alpha(\|x\|_{\mathcal{A}}). \tag{3.4}$$

Then, $\mathcal{A}$ is LAS (with a domain of attraction that is not better specified).

We remark that condition (3.3), combined with sufficient regularity of $V(x)$, is equivalent to the positive definiteness of $V(x)$ conditional to the set $\mathscr{A}$, i.e., $V(x) = 0$ for all $x \in \mathscr{A}$ and $V(x) > 0$ for all $x \notin \mathscr{A}$.

Theorem 3.1.3. (Lyapunov's theorem for global stability, [77, Chapter 10]).

Assume $\alpha_l, \alpha, \alpha_u$ are class $\mathscr{K}_\infty$ functions and let the assumptions of Theorem 3.1.2 hold with $d \to \infty$. Then, $\mathscr{A}$ is globally asymptotically stable.

Lemma 3.1.1. (Comparison Lemma, [76, Chapter 3]).

Consider the scalar differential equation

$$\dot{y} = \phi(y), \qquad y(t_0) = y_0 \tag{3.5}$$

where $y \in \mathbb{R}$ and $\alpha : \mathbb{R} \to \mathbb{R}$ is a locally Lipschitz function. Let $[t_0, T)$ be the maximal interval of definition of $y(t)$.

Let $v(t)$ be a continuous function whose time derivative $\dot{v}(t)$ satisfies

$$\dot{v}(t) \leq \phi(v(t)), \qquad v(t_0) \leq y_0. \tag{3.6}$$

Then,

$$v(t) \leq y(t), \quad \forall t \in [t_0, T). \tag{3.7}$$

The following theorem provides conditions for global exponential stability. Due to its central role in Chapters 6, 7 and 8, we also provide a proof for the result.

Theorem 3.1.4. (Global exponential stability of an equilibrium point).

Assume there exists a function $V : \mathbb{R}^n \to \mathbb{R}$ such that there are real constant $a_l, a_u, c > 0$ such that

$$a_l \|x - x^*\| \leq V(x) \leq a_u \|x - x^*\| \tag{3.8}$$

$$\dot{V}(x) \leq -cV(x). \tag{3.9}$$

Then, for all $x(0) = x_0 \in \mathbb{R}^n$,

$$\|x(t) - x^*\| \leq \sqrt{\frac{a_u}{a_l}} e^{-\frac{1}{2}ct} \|x_0 - x^*\|, \tag{3.10}$$

i.e., the point x^* is globally exponentially stable with convergence rate $c/2$.

Proof. The proof is based on the application of Lemma 3.1.1. Choose $\phi(y) = -cy$ and $v(t) = V(x(t))$ in Lemma 3.1.1; then we have $y(t) = V(x_0)e^{-ct}$ solution to the scalar differential equation $\dot{y} = -cy$, and

$$V(x(t)) \leq \underbrace{e^{-ct}V(x_0)}_{y(t)}.$$

Consequently,

$$\|x - x^*\|^2 \leq \frac{1}{a_l} V(x(t)) \leq \frac{1}{a_l} e^{-ct} V(x_0) \leq \frac{a_u}{a_l} e^{-ct} \|x_0 - x^*\|^2,$$

The result follows by taking the square root on both sides. □

We remark that if $V = x^\top Px$, then $a_l = \lambda_{\min}(P)$, $a_u = \lambda_{\max}(P)$, and thus the coefficient of the exponential term is proportional to the condition number of P.

3.2 Feedback linearization controller design

This section reviews basic concepts related to feedback linearization controller design for multiple-input multiple-output systems.
The content of this section is mainly based on the book "Nonlinear Control Systems" by A. Isidori [43].

3.2.1 Geometric theory of dynamical systems

Definition 3.2.1. (Lie derivative, [43, Section 1.2], [76, Section 13.2]).

Let $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ be a vector field and $H_i : \mathbb{R}^n \mapsto \mathbb{R}$. The Lie derivative of H_i along F is

$$L_F H_i(x) = \nabla H_i(x)^\top F(x) \in \mathbb{R}. \tag{3.11}$$

By defining $L_F^0 H_i(x) = H_i(x)$, for $k = 1, 2, \ldots$, we have

$$L_F^k H_i(x) = (\nabla L_F^{k-1} H_i(x))^\top F(x). \tag{3.12}$$

As illustrated, e.g., in [43, 76], we can apply feedback linearization to input-affine nonlinear dynamical systems of the form

$$\begin{aligned} \dot{x} &= F(x) + G(x)u \\ y &= H(x) \end{aligned} \tag{3.13}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $y(t) \in \mathbb{R}^m$, $F : \mathbb{R}^n \mapsto \mathbb{R}^n$, $G : \mathbb{R}^n \mapsto \mathbb{R}^{n,m}$, $H : \mathbb{R}^n \mapsto \mathbb{R}^m$. For dynamical systems of this kind, we define the concept of vector relative degree.

Let $G_j(x)$ be the j-th column of $G(x)$.

Definition 3.2.2. (Relative degree, [43, Section 5.1], [76, Section 13.2]).

The system (3.13) has a (local) vector relative degree $r = (r_1, \ldots, r_m)^\top$ at $\bar{x} \in \mathbb{R}^n$ if

(a) there exists an $\varepsilon > 0$ such that

$$L_{G_j} L_F^k H_i(x) = 0, \quad \forall x \in \mathscr{B}_\varepsilon(\bar{x}) \tag{3.14}$$

for each $1 \leq i, j \leq m$, for each $k \in \mathbb{N}$ such that $k < r_i - 1$, where $\mathscr{B}_\varepsilon(\bar{x})$ is a ball of radius ε centered in $\bar{x}$.

(b) the matrix $\left[L_{G_j}L_F^{r_i-1}H_i(x)\right]_{1\le i,j\le m} \in \mathbb{R}^{m,m}$ *is nonsingular at* $x=\bar{x}$*, i.e.,*

$$rank\left(\left[L_{G_j}L_F^{r_i-1}H_i(\bar{x})\right]_{1\le i,j\le m}\right)=m. \tag{3.15}$$

For a single-input, single-output system, if the relative degree is r, the output y and its derivatives up to $(r-1)$-th order do not depend on the input, while the r-th order derivative does.

For linear systems, the concept is equivalent to the relative degree of a transfer function, i.e., the difference between the degree of the denominator and the degree of the numerator. The physical meaning of this is related to the delay (phase shift) between input and output at high frequency, which influences the response speed. Similarly, for nonlinear systems, the physical meaning of the relative degree is associated with a delay between input and output.

Now, we recall the concept of normal form and zero dynamics.

Given a nonlinear dynamical system of the kind (3.13) with relative degree r at x_0, there exists a (local) diffeomorphism (i.e., a smooth change of coordinates) $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$z=\begin{bmatrix}\zeta\\ \eta\end{bmatrix}=\Phi(x), \tag{3.16}$$

where

$$\zeta_1=\phi_1(x)=H_1(x), \tag{3.17a}$$
$$\vdots$$
$$\zeta_{r_1}=\phi_{r_1}(x)=L_F^{r_1-1}H_1(x), \tag{3.17b}$$
$$\vdots$$
$$\zeta_{r-r_m}=\phi_{r-r_m}(x)=H_m(x), \tag{3.17c}$$
$$\vdots$$
$$\zeta_r=\phi_r(x)=L_F^{r_m-1}H_m(x) \tag{3.17d}$$
$$\eta_1=\phi_{r+1}(x), \tag{3.17e}$$
$$\vdots$$
$$\eta_{n-r}=\phi_n(x) \tag{3.17f}$$

and $\phi_i(x)$ for $i \in \{r+1,\dots,m\}$ are such that $L_{G_j}\phi_i(x) = 0 \quad \forall j \in \{1,\dots,m\}$.

Definition 3.2.3. (Normal form, [43, Section 5.1])

The dynamical system (3.13) described in the new coordinates (3.16)-(3.17) is said to be the normal form of the system.

We can write the normal form explicitly as

$$\dot{\zeta}_1 = \zeta_2, \quad \dots, \quad \dot{\zeta}_{r_1-1} = \zeta_{r_1}, \tag{3.18a}$$

$$\dot{\zeta}_{r_1} = L_F^{r_1} H_1(\Phi^{-1}(z)) + \sum_{j=1}^{m} L_{G_j} L_F^{r_1-1} H_1(\Phi^{-1}(z)) u_j \tag{3.18b}$$

$$\vdots$$

$$\dot{\zeta}_{r-r_m} = \zeta_{r-r_m+1} \tag{3.18c}$$

$$\dot{\zeta}_r = L_f^{r_m} H_m(\Phi^{-1}(z)) + \sum_{j=1}^{m} L_{G_j} L_F^{r_m-1} H_m(\Phi^{-1}(z)) u_j \tag{3.18d}$$

$$\dot{\eta} = q(\zeta,\eta). \tag{3.18e}$$

Definition 3.2.4. (Zero dynamics, [80],[43, Section 4.3]).

The zero dynamics of a system $\mathscr{S}$ is the dynamical system that describes the internal behavior of $\mathscr{S}$ if we set the initial conditions and the inputs to constrain the output of $\mathscr{S}$ to be identically zero. For a system of the kind (3.13), this is represented by $\dot{\eta} = q(0,\eta)$, where $q : \mathbb{R}^r \times \mathbb{R}^{n-r} \to \mathbb{R}^{n-r}$ is defined in (3.18).

In other words, the zero dynamics accounts for the modes that are not observable from the system's output, whose stability is of concern. When dealing with linear systems $\dot{x} = Ax + Bu, y = Cx$, it is possible to show that the zero dynamics is of the form $\dot{\eta} = Q\eta$, with $\eta(t) \in \mathbb{R}^{n-r}, Q \in \mathbb{R}^{n-r \times n-r}$, and with eigenvalues of the matrix Q exactly equal to the zeros of the transfer function of the system $H(s) = C(sI_n - A)^{-1}B$. We refer the reader to [81] and [43, Section 6.1] for more insight on this notion.

3.2.2 Nonlinear feedback control of multivariable systems

Next, we state the non-interacting control problem and its solution.

Definition 3.2.5. (Non-interacting control problem, [43, Section 5.3])

Let us consider system (3.13). We say that a controller of the form

$$u = \alpha(x) + \beta(x)v \tag{3.19}$$

with $\alpha(x) \in \mathbb{R}^m$, $\beta(x) \in \mathbb{R}^{m,m}$ and $v(t) \in \mathbb{R}^m$ solves the non-interacting control problem if in the system

$$\begin{cases} \dot{x} = F(x) + G(x)\alpha(x) + G(x)\beta(x)v \\ y = H(x) \end{cases} \tag{3.20}$$

each input $v_i(t)$ affects only output $y_i(t)$, for each $i = 1, \ldots, m$.

The non-interacting control problem admits a solution if the plant has a finite relative degree. Moreover, the zero dynamics behavior determines the closed-loop system's stability. The following results detail these facts.

Theorem 3.2.1. (Non-interacting control solution, [43, Section 5.3])

The non-interacting control problem admits a solution if and only if system (3.13) has some vector relative degree r. In that case, the solution is given by

$$\alpha(x) = -A^{-1}(x)b(x), \qquad \beta(x) = A^{-1}(x) \tag{3.21}$$

where

$$A(x) = \left[L_{G_j} L_F^{r_i - 1} H_i(x)\right]_{1 \le i,j \le m} \in \mathbb{R}^{m,m} \tag{3.22a}$$

$$b(x) = [L_F^{r_i} H_i(x)]_{i=1,\ldots,m} \in \mathbb{R}^m. \tag{3.22b}$$

Moreover, for each $i = 1, 2, \ldots, m$, the input-output behavior between $v_i(t)$ and $y_i(t)$ is linear and described in the $s-$domain by the transfer function

$$\frac{1}{s^{r_i}}. \tag{3.23}$$

Theorem 3.2.1 provides an input-output global linearization of system (3.13), which is decoupled in each component $i = 1, \ldots, m$.

Next, in Theorem 3.2.2, we relate the stability of the controlled system to the stability of the zero dynamics of the plant.

Theorem 3.2.2. (Non-interacting control stability, [43, Section 5.3] [78, Section 6.5])

Assume that system (3.13) *has vector relative degree* r *and that its zero dynamics is asymptotically stable at* $\eta = \bar{\eta}$. *Next, select*

$$v_i(t) = \mathscr{G}_i(y_i), \quad i = 1, \ldots, m \tag{3.24}$$

such that the closed loop system (3.23)-(3.24) *is asymptotically stable at* $y_i = 0$ *for all* $i = 1, \ldots, m$.

Then the feedback control system defined by (3.20) *and* (3.24) *(with* $\alpha(x)$ *and* $\beta(x)$ *given in Theorem 3.2.1), is asymptotically stable at* $(\eta, \xi) = (\bar{\eta}, 0)$.

We remark that in [43, Section 5.3] and [78, Section 6.5] the proof of this theorem is given assuming equilibrium point $\bar{\eta} = 0$ for the zero dynamics, but the argument is without loss of generality for any desired equilibrium point.

Chapter 4

Optimization problems

This chapter reviews the fundamental results of mathematical optimization theory. The material in this chapter is mainly based on the books "Nonlinear Programming" by D. P. Bertsekas [40], "Numerical Optimization" by J. Nocedal & S. J. Wright [41], and "Convex Optimization" by S. Boyd & L. Vandenberghe [82], and on referred papers.

We formally define an optimization problem as

Definition 4.0.1. (Optimization problem and feasible set, [82, Chapter 4]), [40, Chapter 3].

An optimization problem is a problem of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ s.t. \quad & g_i(x) \leq 0 \quad i = 1,\dots,p \\ & h_i(x) = 0 \quad i = 1,\dots,m. \end{aligned} \tag{4.1}$$

The set $\mathscr{X}$ of x such that the constraints are satisfied is called the feasible set.

For an optimization problem, we define global and local solutions according to the following definitions:

Definition 4.0.2. (Global minimum, [82, Chapter 4]).

A point x^* is said to be the global minimum of the problem (4.1) if

$$f(x^*) \leq f(x) \quad \forall x \in \mathscr{X}. \tag{4.2}$$

Definition 4.0.3. (Local minimum, [82, Chapter 4]).

A point x^* is said to be a local minimum of the problem (4.1) if

$$\exists \varepsilon > 0 \text{ s.t. } f(x^*) \leq f(x) \quad \forall x \in \mathscr{X} \text{ with } \|x - x^*\| < \varepsilon. \tag{4.3}$$

A property of paramount importance for an optimization problem is convexity.

Definition 4.0.4. (Convex function, [82, Chapter 3]).

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be convex if satisfies

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \qquad \forall x, y \in \mathbb{R}^n \quad t \in [0,1]. \tag{4.4}$$

An optimization problem is convex if f, g_i are convex and h_i are affine in the optimization variables x.

The following theorem motivates distinguishing between convex and non-convex optimization problems.

Theorem 4.0.1. (Optimality for convex problems, [82, Section 4.2])

If the optimization problem (4.1) is convex, all local minima are also global minima.

As a consequence of this result, convex problems are tractable, while non-convex ones are considered hard.

4.1 Optimality conditions

This section reviews the necessary and sufficient optimality conditions for unconstrained, equality-constrained, and inequality-constrained optimization problems.

4.1.1 Unconstrained optimization

Theorem 4.1.1. (Unconstrained problems, [40, Chapter 1]).

Consider an unconstrained optimization problem, i.e., $\min_{x\in\mathbb{R}^n} f(x)$. *If* $f(x)$ *is twice differentiable on an open set* S *containing* x^*, *then* x^* *is a local minimum if and only if*

$$\nabla f(x^*) = 0 \quad \textit{(first order condition)} \tag{4.5}$$

and

$$\nabla^2 f(x^*) \succeq 0 \quad \textit{(second order condition)}. \tag{4.6}$$

Theorem 4.1.1 establishes necessary and sufficient conditions for local optimality. If the problem is convex, these conditions become necessary and sufficient for global optimality. Notice that the condition $\nabla f(x^*) = 0$ is only necessary. Points x^* which satisfy this equation are called stationary points.

4.1.2 Equality constrained optimization

Next, we consider problems with equality constraints only, i.e.,

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \quad h_i(x) = 0, \quad i = 1,\dots,m. \end{aligned} \tag{4.7}$$

For this kind of problem, we define the Lagrangian function as

$$\mathscr{L}(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x). \tag{4.8}$$

Theorem 4.1.2. (Lagrange multipliers Theorem - Necessary condition, [40, Section 4.1]).

Consider an equality-constrained optimization problem of the kind (4.7). Let x^* *be a local minimum of the problem. Assume that* $f(x), h_i(x)$ *are twice differentiable on an open set* S *containing* x^* *and that* x^* *is regular, i.e.,* $\nabla_x h_1(x^*)$,

..., $\nabla_x h_m(x^*)$ are linearly independent. Then there exists a unique vector $\lambda^* \in \mathbb{R}^m$, called Lagrange multipliers vector, such that

$$\nabla_x \mathscr{L}(x^*, \lambda^*) = 0 \quad \textit{(first order condition)}, \tag{4.9}$$

$$\nabla_\lambda \mathscr{L}(x^*, \lambda^*) = h(x^*) = 0_m \quad \textit{(primal feasibility)}, \tag{4.10}$$

and $y^\top \nabla_x^2 \mathscr{L}(x^*, \lambda^*) y \succeq 0$ for all feasible variations $y \in V(x^*) \doteq \{y \in \mathbb{R}^n : \nabla_x h_i(x^*)^\top y = 0, i = 1, \ldots, m\}$ (second order condition).

A pair $(x^\star, \lambda^\star)$ that satisfies both (4.9) and (4.10) is said to be a *stationary point* for the problem (4.7).

Theorem 4.1.3. (Lagrange multipliers Theorem - Sufficient conditions, [40, Section 4.2]).

Consider an equality-constrained optimization problem of the kind (4.7). Assume $f(x), h_i(x)$ are twice differentiable on an open set S containing $x^* \in \mathbb{R}^n$. If there exists a vector $\lambda^* \in \mathbb{R}^m$ such that (4.10), (4.9), and $y^\top \nabla^2 \mathscr{L}(x^*, \lambda^*) y \succeq 0$ for all feasible variations $y \in V(x^*) \doteq \{y \in \mathbb{R}^n : \nabla_x h_i(x^*)^\top y = 0, i = 1, \ldots, m\}$ (second order condition), then x^* is a strict local minimum of the problem, i.e., $\exists \gamma > 0, \varepsilon > 0$ such that $f(x) \geq f(x^*) + \gamma \|x - x^*\|^2$ for all x such that $h(x) = 0$ and $\|x - x^*\| > \varepsilon$.

4.1.3 Inequality constrained optimization

Finally, we consider problems with inequality constraints of the kind (4.1). For such problems, we distinguish between active inequality constraints, i.e., constraints that hold with equality, and inactive ones, i.e., that hold with strict inequality. Formally, the set of active constraints is defined for any feasible point x as

$$A(x) = \{j \text{ s.t. } g_j(x) = 0\}. \tag{4.11}$$

A feasible point x is said to be regular if $\nabla_x h_i(x), i = 1, \ldots, m$ and $g_j(x), j \in A(x)$ are linearly independent.

For such kind of problems, we define the Lagrangian as

$$\mathscr{L}(x,\lambda,\mu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{p} \mu_j g_j(x). \tag{4.12}$$

Theorem 4.1.4. (Karush-Kuhn-Tucker - Necessary condition, [40, Section 4.3]).

Consider an optimization problem of the kind (4.1). Let x^* be a local minimum of the problem. Assume that $f(x)$, $h_i(x)$, and $g_i(x)$ are twice differentiable on an open set S containing x^* and that x^* is regular. Then there exists unique vectors $\lambda^* \in \mathbb{R}^m, \mu^* \in \mathbb{R}^p$, called Lagrange multipliers vectors, such that

$$\nabla_x \mathscr{L}(x^*,\lambda^*,\mu^*) = 0, \quad \textit{(first order condition)}, \tag{4.13a}$$
$$h(x^*) = 0_m, g(x^*) \leq 0, \quad \textit{(primal feasibility)}, \tag{4.13b}$$
$$\mu_j^* \geq 0, \forall j = 1,\ldots,p, \quad \textit{(dual feasibility)}, \tag{4.13c}$$
$$\mu_j^* g_j(x^\star) = 0, \forall j = 1,\ldots,p, \quad \textit{(complementary slackness)}, \tag{4.13d}$$

and

$$y^\top \nabla_x^2 \mathscr{L}(x^*,\lambda^*,\mu^*) y \succeq 0 \tag{4.14}$$

for all feasible variations $y \in V(x^*) \doteq \{y \in \mathbb{R}^n : \nabla_x h_i(x^*)^\top y = 0, i = 1,\ldots,m, \nabla_x g_j(x^*)^\top y = 0, j \in A(x^*)\}$ (second order condition).

Theorem 4.1.5. (General sufficiency conditions, [40, Section 4.3]).

Consider an optimization problem of the kind (4.1). Assume that $f(x)$, $h_i(x)$, and $g_i(x)$ are twice differentiable on an open set S containing $x^* \in \mathbb{R}^n$. If there exists vectors $\lambda^* \in \mathbb{R}^m, \mu^* \in \mathbb{R}^p$ such that $\nabla_x \mathscr{L}(x^*,\lambda^*,\mu^*) = 0$ (first order condition), $h(x^*) = 0_m, g(x^*) \leq 0_p$ (primal feasibility), $\mu_j > 0$ for all $j \in A(x^*)$, $\mu_j = 0$ for $j \notin A(x^*)$ (dual feasibility), and $y^\top \nabla_x^2 \mathscr{L}(x^*,\lambda^*,\mu^*) y \succeq 0$ for all feasible variations $y \in V(x^*) \doteq \{y \in \mathbb{R}^n : \nabla_x h_i(x^*)^\top y = 0, i = 1,\ldots,m, \nabla_x g_j(x^*)^\top y = 0, j \in A(x^*)\}$ (second order condition), then x^* is a strict local minimum of the problem.

4.2 Optimization algorithms

In this section, we review optimization algorithms for unconstrained and equality-constrained optimization. The list of considered algorithms is limited to those useful in the following chapter's discussion and should not be considered exhaustive.

4.2.1 Unconstrained problems

When dealing with (differentiable) unconstrained optimization, the most commonly considered algorithm is *gradient descent*. It consists of iterating the update rule

$$x \leftarrow x - \alpha P \nabla_x f(x) \tag{4.15}$$

where $P \succ 0$ and $\alpha > 0$.

Different choices of P give rise to different optimization algorithms. If $P = I$, then we deal with the steepest descent algorithm. If $P = (\nabla_x^2 f(x))^{-1}$, we deal with Newton's method. For a detailed analysis of these methods, the reader is referred to [40, Chapter 1] and [41, Chapter 3].

For nonlinear least-squares problems, i.e., $f(x) = (1/2)\sum_{i=1}^{N} g_i(x)^2$, it is possible to derive an approximation of Newton's method that does not require computing the Hessians of g_i. In particular, we take the approximation

$$\nabla_x^2 f(x) = \nabla_x g(x) \nabla_x g(x)^\top + \sum_{i=1}^{N} \nabla_x^2 g_i(x) g_i(x) \approx \nabla_x g(x) \nabla_x g(x)^\top. \tag{4.16}$$

This approximation leads to the update rule:

$$x \leftarrow x - \alpha (\nabla_x g(x) \nabla_x g(x)^\top)^{-1} \underbrace{\nabla_x g(x) g(x)}_{\nabla_x f(x)}. \tag{4.17}$$

This algorithm is called the Gauss-Newton method, and it can be shown to be equivalent to the algorithm obtained by globally optimizing a linearized cost at each iteration. A modification of this algorithm is the Levenberg–Marquardt algorithm, which replaces $P = \nabla_x g(x) \nabla_x g(x)^\top$ with $P = \nabla_x g(x) \nabla_x g(x)^\top + \gamma \mathrm{diag}(\nabla_x g(x) \nabla_x g(x)^\top)$ where $\gamma > 0$ is called damping parameter and adjusted at each iteration using a

suitable strategy. It is outside the scope of this review to enter into more details; we refer the interested reader to [41, Section 10.3] and [83].

4.2.2 Constrained problems

When dealing with constrained problems, several classes of algorithms are available. Interior-point methods [41, Chapter 19], sequential quadratic programming [41, Chapter 18], penalty and Lagrangian methods [41, Chapter 17], [40, Chapter 5], conditional and projected gradient [84], to mention a few.

In this thesis, we mostly consider Lagrangian methods. Such methods aim at solving the Lagrangian system (4.18).

$$\begin{cases} \nabla_x \mathscr{L}(x,\lambda) = 0 \\ h(x) = 0. \end{cases} \tag{4.18}$$

The simplest method in this family is the first-order Lagrangian method that iterates the following update rules

$$\begin{aligned} x &\leftarrow x - \alpha \nabla_x \mathscr{L}(x,\lambda) \\ \lambda &\leftarrow \lambda + \eta h(x), \end{aligned} \tag{4.19}$$

where $\alpha > 0$ is a small scalar and $\eta > 0$.

The rationale of this method is that the Lagrangian should be minimized along the x direction and maximized along the multiplier's direction λ. This method can be proven to converge to a minimum-Lagrange multiplier pair (x^*, λ^*) under the assumptions that α is small enough, x^* is a regular point and $\nabla^2_{xx} \mathscr{L}(x^*, \lambda^*) \succeq 0$. Other possibilities include Newton-like methods to find the solutions of (4.18) and its variations. The interested reader is referred to [40, Section 4.4] for a detailed presentation.

Related to Lagrangian methods, an alternative is to consider the augmented Lagrangian method (ALM). This method is based on the definition of augmented Lagrangian:

$$\mathscr{L}_A(x,\lambda) = \mathscr{L}(x,\lambda) + \frac{\eta}{2} \|h(x)\|_2^2, \qquad \eta > 0. \tag{4.20}$$

At each iteration, we update the primal variable x according to the unconstrained minimization

$$x \leftarrow \arg\min_{x \in \mathbb{R}^n} \mathscr{L}_A(x, \lambda) \tag{4.21}$$

and the Lagrange multipliers according to

$$\lambda \leftarrow \lambda + \eta h(x). \tag{4.22}$$

It is possible to show that the ALM converges to an optimal pair (x^*, λ^*) under the assumption that x^* is a regular point and that η is sufficiently large. The method can be extended to the presence of inequality constraints by adding positive slack variables $s_j \geq 0$ to transform inequalities $g_j(x) \leq 0$ to equalities $g_j(x) + s_j = 0$ and then applying a projected gradient method to handle bounds on the variables in the optimization (4.21); see, e.g., [41, Chapter 17].

A variant of the ALM is the alternating direction method of multipliers (ADMM). ADMM solves optimization problems of the form

$$\begin{aligned} \min_{x,z \in \mathbb{R}^n} \quad & F(x) + G(z) \\ \text{s.t.} \quad & x = z \end{aligned} \tag{4.23}$$

We consider the augmented Lagrangian

$$L_\rho(x, z, u) = F(x) + G(z) + \rho u^\top (x - z) + \frac{\rho}{2} \|x - z\|_2^2 \tag{4.24}$$

where $\rho > 0$ is a scalar penalty parameter and $u \in \mathbb{R}^n$ is the scaled dual variable; we refer the reader to [85, Section 3.1] for details on the scaled ADMM. In the following, we denote by x_k, z_k, and u_k the estimates of the variables x, z, and u, respectively, at a generic k-th iteration of the algorithm. ADMM consists of a loop iterating over three steps, called x-update, z-update, and u-update:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} L_\rho(x, z_k, u_k) = \arg\min_{x \in \mathbb{R}^n} F(x) + \frac{\rho}{2} \|x - z_k + u_k\|_2^2. \tag{4.25}$$

$$z_{k+1} = \arg\min_{z \in \mathbb{R}^n} L_\rho(x_k, z, u_k) = \arg\min_{z \in \mathbb{R}^n} G(z) + \frac{\rho}{2} \|x_{k+1} - z + u_k\|_2^2. \tag{4.26}$$

$$u_{k+1} = u_k + x_{k+1} - z_{k+1}. \tag{4.27}$$

Under the assumptions that (i) the functions F and G are closed, proper, and convex, and (ii) the non-augmented Lagrangian $\mathscr{L}(x,z,u)$ of the optimization problem has a saddle point, then ADMM is proved to be convergent to the global optima [85]. This algorithm has also been successfully applied to non-convex problems, e.g., in [86–88], but establishing convergence for the general case is still an open problem.

4.3 Optimization algorithms as dynamical systems

A possible approach to analyzing optimization algorithms is related to continuous-time (CT) dynamical systems. The general paradigm is as follows: first, we take the CT limit of the iterations defined by the considered algorithm, and then we investigate the convergence of the obtained dynamical system. This approach is motivated by the consideration that when the selected discretization step size is small enough, the original discrete-time algorithm will also enjoy the found properties.

This section reviews existing literature studying dynamical systems whose trajectories converge toward the solution to a given optimization problem. Specifically, we review gradient flows for unconstrained minimization, primal-dual gradient flows, and null-space gradient flows for constrained problems.

Beyond these, we also mention the possibility of constructing a dynamical system that converges towards the solution to a set of equations. We can use these methods to solve optimization problems if these equations represent optimality conditions. Among these methods, we mention [89], which defines a control-theoretic approach to the problem, and [90], which proposes defining an autonomous system using a functional of the equations, called pseudo-transient continuation. The latter has been recently applied to define an efficient algorithm for convex quadratic programming in [91].

4.3.1 Gradient flows

We begin by considering unconstrained and differentiable optimization problems

$$x^* = \arg\min_{x\in\mathbb{R}^n} f(x). \tag{4.28}$$

As discussed in Section 4.2, the mainstream algorithm to solve these problems is gradient descent. The CT counterpart of the gradient descent algorithm is obtained by letting $\alpha \to 0$ and considering updates of x at infinitesimal time increment, i.e., the forward time shift operator is replaced by the time derivative. This leads to:

$$\dot{x}(t) = -\nabla_x f(x), \qquad x(0) = x_0, \tag{4.29}$$

which is referred to as *(negative) gradient dynamics*. The trajectories of this dynamics, i.e., solutions of the initial value problem (4.29), are called *(negative) gradient flows*. Their existence is guaranteed under the condition that $\nabla_x f(x)$ is at least locally Lipschitz.

Early works on the gradient dynamics established convergence of the flows to a stationary point of the optimization problem. In [92], the author proves that if $f(x)$ is analytic, then every solution starting in a compact sublevel set of $f(x)$ has a finite length (as a curve in $\mathbb{R}^n$) and converges to an equilibrium point $\bar{x}$.

By definition of an equilibrium point, $\nabla_x f(\bar{x}) = 0$ and therefore $\bar{x}$ is a stationary point of (4.28). Moreover, a straightforward application of the Lyapunov linearization method (Theorem 3.1.1, Section 3.1) leads to the conclusion that $\bar{x}$ is locally exponentially stable if and only if $\nabla_x^2 f(\bar{x})$ is positive definite, and $\bar{x}$ is unstable if at least one eigenvalue of $\nabla_x^2 f(\bar{x})$ is strictly negative. In other words, local minima satisfying second-order conditions are always stable. For further references, we refer the reader to [93, Appendix A.6], [94], and [95].

A more general approach allows for considering any possible descent directions rather than only the steepest decent one. This approach consists of the analysis of the dynamics

$$\dot{x} = -Q(x)\nabla_x f(x), \qquad Q(x) \succ 0. \tag{4.30}$$

See, e.g., [96] for a discussion on the properties of the equilibria of this dynamics and [97] for a convergence result.

More recently, motivated by the growing popularity of accelerated gradient methods in machine learning applications, many efforts have been devoted to analyzing *accelerated gradient flows*. An instance of such a method is Nesterov's accelerated gradient flow, the CT version of Nesterov's method, i.e.,

$$\ddot{x} + \frac{3}{t}\dot{x} + \nabla_x f(x) = 0. \tag{4.31}$$

See, e.g., [98] and [99] for a detailed discussion.

In [100, 101], the authors propose a unifying framework to study optimization algorithms as dynamical systems whose flow converges to a minimum of an unconstrained optimization problem. Specifically, they propose defining a feedback interconnection between a linear system and a nonlinear component defined by $\nabla_x f(x)$. Different optimization algorithms arise depending on the choice of the linear subsystem, including the gradient descent (4.29) and the Nesterov's (4.31) dynamics. The stability and convergence of the algorithm are studied by treating the nonlinear component as an uncertain block satisfying integral quadratic constraints.

We also mention [102], where the authors consider dynamical systems related to ADMM and accelerated ADMM. Interestingly, despite the conceptual gap between the corresponding optimization methods, the ADMM dynamics is strictly related to the negative gradient dynamics.

4.3.2 PDGD

Primal-dual gradient dynamics (PDGD), or saddle-point dynamics, is the CT limit of the iterations defined by the Lagrangian method; see Section 4.2.

When dealing with equality-constrained optimization problems in the form

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\quad \text{subject to:} \\ &\quad h(x) = 0, \end{aligned} \tag{4.32}$$

we recall the definition of Lagrangian

$$\mathscr{L}(x,\lambda) = f(x) + \lambda^\top h(x). \tag{4.33}$$

The PDGD is defined as the dynamical system

$$\dot{x} = -\nabla_x \mathscr{L}(x,\lambda) \tag{4.34a}$$
$$\dot{\lambda} = \eta \nabla_\lambda \mathscr{L}(x,\lambda) = \eta h(x) \tag{4.34b}$$

where $\eta \in \mathbb{R}$ is a positive constant. In [103], the authors establish the global exponential stability of PDGD for strongly convex optimization problems. We provide the formal statement in Theorem 4.3.1.

Theorem 4.3.1. (Global exponential stability of PDGD, [103, Section II-A])

Assume f is twice differentiable and there exists $\mu, \ell > 0$ such that, for all $x, y \in \mathbb{R}^n$, $\mu \|x-y\|^2 \leq \langle \nabla_x f(x) - \nabla_x f(y), x-y \rangle \leq \ell \|x-y\|^2$. Moreover, assume that $h(x) = Cx + d$ (the optimization problem is convex), with C full row rank and such that there exist constants κ_1, κ_2 such that $\kappa_1 I \preceq CC^\top \preceq \kappa_2 I$. Define $\tau_{eq} = \min\{\frac{\eta\kappa_1}{4\ell}, \frac{\kappa_1\mu}{4\kappa_2}\}$. Then there exist positive constants $c_1, c_2 \in R$ such that

$$\|x(t) - x^*\| \leq c_1 e^{-\frac{1}{2}\tau_{eq} t}, \qquad \|\lambda(t) - \lambda^*\| \leq c_2 e^{-\frac{1}{2}\tau_{eq} t}. \tag{4.35}$$

The proof of Theorem 4.3.1 relies on constructing a quadratic Lyapunov function.

In [103], the authors also consider inequality-constrained problems of the kind

$$\begin{aligned} x^* &= \arg\min_{x \in \mathbb{R}^n} f(x) \\ &\text{s.t. } g(x) = Cx - d \leq 0 \end{aligned} \tag{4.36}$$

where $C \in \mathbb{R}^{m,n}, d \in \mathbb{R}^m$. When dealing with this case, they propose defining the augmented Lagrangian

$$\mathscr{L}_a(x,\lambda) = f(x) + \sum_{j=1}^{m} H_\rho(g_i(x), \lambda_i) \tag{4.37}$$

where

$$H_\rho(g_i(x), \lambda_i) = \begin{cases} g_i(x)\lambda_i + \frac{\rho}{2} g_i^2(x) & \text{if } \rho g_i(x) + \lambda \geq 0 \\ -\frac{1}{2\rho}\lambda_i^2 & \text{otherwise} \end{cases} \tag{4.38}$$

They define the primal-dual gradient dynamics for the augmented Lagrangian (Aug-PDGD) as

$$\dot{x} = -\nabla_x \mathscr{L}_a(x,\lambda) \tag{4.39a}$$
$$\dot{\lambda} = \eta \nabla_\lambda \mathscr{L}_a(x,\lambda) \tag{4.39b}$$

and establish the following theorem:

Theorem 4.3.2. (Global exponential stability of Aug-PDGD [103, Section II-B])

Assume f is twice differentiable and there exists $\mu, \ell > 0$ such that, for all $x, y \in \mathbb{R}^n$, $\mu \|x-y\|^2 \leq \langle \nabla_x f(x) - \nabla_x f(y), x-y \rangle \leq \ell \|x-y\|^2$. Moreover, assume that C full row rank and such that there exist constants κ_1, κ_2 such that $\kappa_1 I \preceq CC^\top \preceq \kappa_2 I$. Define $\tau_{ineq} = \frac{\eta \kappa_1^2}{40\ell\kappa_2 \max\{\frac{\rho\kappa_2}{\mu}, \frac{\ell}{\mu}\}^2 \max\{\frac{\eta}{\ell\rho}, \frac{\ell}{\mu}\}^2}$. Then there exist positive constants $c_3, c_4 \in \mathbb{R}$ such that

$$\|x(t) - x^*\| \leq c_3 e^{-\frac{1}{2}\tau_{ineq} t}, \qquad \|\lambda(t) - \lambda^*\| \leq c_4 e^{-\frac{1}{2}\tau_{ineq} t}. \tag{4.40}$$

More recently, [104] considered smooth convex optimization problems with generic convex inequality constraints. The analysis of this more general problem is more challenging than the linear case counterpart, and, in general, the corresponding Augmented PDGD does not enjoy global exponential stability. This statement is motivated by a counterexample given in [104]. Instead, in this case, PDGD achieves semiglobal exponential convergence, i.e., the exponential rate of convergence must be allowed to depend on initial conditions.

4.3.3 Null-space and projected gradient flows

When dealing with non-convex equality-constrained optimization problems, PDGD may not converge or be inefficient.

An alternative approach that relies on dynamical systems to achieve convergence to a minimum of the optimization problem

$$\begin{aligned} x^* = \arg\min_{x \in \mathbb{R}^n} & f(x) \\ \text{s.t.} \quad & h(x) = 0_m \end{aligned} \tag{4.41}$$

is based on the idea of updating $x(t)$ along a combination of the null space of the feasible set and its orthogonal complement. For this reason, we refer to such methods as *null space gradient methods*.

The earliest work in this direction is [105], where the author considers the dynamics defined by

$$\dot{x} = -(I - J_h^\top (J_h J_h^\top)^{-1} J_h) \nabla_x f(x). \tag{4.42}$$

(4.42) is the generalization CT limit of Rosen's gradient projection method [106]. This method enjoys forward invariance of the feasible set Ω (i.e., trajectories starting in Ω remain in Ω), but trajectories starting outside do not converge to it.

To address this issue, in [107], the author propose the dynamics

$$\dot{x} = -(I - J_h^\top (J_h J_h^\top)^{-1} J_h) \nabla_x f(x) - J_h^\top (J_h J_h^\top)^{-1} h(x). \tag{4.43}$$

The introduction of the so-called calibration term $-J_h^\top (J_h J_h^\top)^{-1} h(x)$ renders the feasible set Ω globally attractive, i.e., any trajectory will converge onto it independently on initial conditions.

Variants of this dynamical system have also been studied in [108], [109] and [110]. In particular, these works propose introducing a constant $K > 0$ multiplying the second term of (4.43) and adopting different integration schemes for obtaining the flow numerically. (4.42) can be thought as choosing $K = 0$ in [109].

In [111], the author analyze the more general

$$\dot{x} = -(I - J_h^\top (J_h J_h^\top)^{-1} J_h) \nabla_x f(x) - J_h^\top A(x) h(x) \tag{4.44}$$

where $A(x)$ solves the Lyapunov equation $J_h J_h^\top A + A J_h J_h^\top = W$ for some $W \succ 0$. For the choice $W = 2I$, (4.44) is equivalent to (4.43).

In the case of linear constraints, we may interpret the null-space gradient dynamics as an instance of the globally projected dynamics as defined in, e.g., [112], i.e.,

$$\dot{x} = \lambda (\mathbb{P}_\Omega(x - \alpha \nabla f(x)) - x) \tag{4.45}$$

where $\lambda, \alpha \in \mathbb{R}_+$ and $\mathbb{P}_\Omega(z)$ is the operator projecting z into the feasible set Ω. Indeed, if constraints are linear equalities, i.e., $h(x) = Cx + d = 0$ for some $C \in \mathbb{R}^{m,n}, d \in \mathbb{R}^m$,

the projection operator allows for an explicit formula as

$$\mathbb{P}_{\Omega}(z) = z - C^{\top}(CC^{\top})^{-1}(Cz - d). \tag{4.46}$$

Replacing (4.46) into (4.45), we get

$$\dot{x} = \lambda(x - \alpha\nabla f(x) - C^{\top}(CC^{\top})^{-1}(C(x - \alpha\nabla f(x)) - d) - x) \tag{4.47}$$

$$= \lambda(-\alpha\nabla f(x) + C^{\top}(CC^{\top})^{-1}(\alpha C\nabla f(x) - h(x))). \tag{4.48}$$

Collecing $\nabla f(x)$ and selecting $\lambda = K, \alpha = K^{-1}$, we get the dynamics studied in [109] when specialized for linear constraints. For nonlinear equality constraints, the feasible set Ω is non-convex, and the projection operator is not well-defined due to the potential non-uniqueness of the optima; therefore, the globally projected dynamic is not well-defined in this case.

Part II

Constrained optimization through control

Chapter 5

Feedback control of Lagrange multipliers

This chapter introduces the main results of Part II, which will play a central role in developing the algorithms we will introduce in the following chapters.

In this chapter, we develop a new continuous-time (CT) framework for constrained optimization. The proposed framework leverages a feedback control perspective. We start from the first-order necessary conditions for minima to build a CT dynamic system whose control input is the vector of the Lagrange multipliers. The output represents the constraints, which we regulate accordingly. Several control laws are suitable for the Lagrange multipliers to achieve the desired regulation; this gives rise to a family of control-based first-order methods.

The feedback control interpretation has also been proposed in the recent and independent work [113], where the authors leverage control barrier functions to develop dynamics that solve constrained optimization problems. Instead, in this work, we are not limited to one specific control method, and we provide Lemmas whose validity is independent of the controller choice.

In Chapters 6, 7, and 8, we propose considering proportional-integral and feedback linearization controller design to derive novel optimization algorithms for both equality- and inequality-constrained problems.

5.1 Equality constrained problems

In this section, we illustrate the proposed framework to find a stationary point of the problem

$$\begin{aligned} x^* = \arg\min_{x\in\mathbb{R}^n} f(x) \\ \text{s.t.} \quad \mathrm{h(x)} = 0_\mathrm{m} \end{aligned} \tag{5.1}$$

by building a suitable CT dynamic system with controlled Lagrange multipliers.

Let us define the ficticius plant $\mathscr{P}$ as a dynamical system system with state $x(t) \in \mathbb{R}^n$, input $\lambda(t) \in \mathbb{R}^m$ and output $y(t) \in \mathbb{R}^m$, described by the equations

$$\mathscr{P} : \begin{cases} \dot{x}(t) = -\nabla f(x(t)) - J_h(x(t))^\top \lambda(t) \\ y(t) = h(x(t)). \end{cases} \tag{5.2}$$

The following result holds.

Lemma 5.1.1. *An equilibrium point $(x^\star, \lambda^\star)$ of $\mathscr{P}$ is a stationary point of problem (5.1) if and only if $h(x^\star) = 0$.*

Proof. By definition, an equilibrium point $(x^\star, \lambda^\star)$ of $\mathscr{P}$ satisfies first-order optimality conditions (4.9) in Theorem 4.1.2. If $h(x^\star) = 0$, then also (4.10) holds and $(x^\star, \lambda^\star)$ is a stationary point to problem (5.1). Conversely, any stationary point satisfies (4.9), then it corresponds to an equilibrium point $(x^\star, \lambda^\star)$ of $\mathscr{P}$ with $h(x^\star) = 0$. □

Lemma 5.1.1 suggests we can compute a stationary point of problem (5.1) by designing a suitable input $\lambda(t)$ that drives $\mathscr{P}$ to converge to an equilibrium point and regulates the output to zero. A standard way to approach this regulation problem is to design a suitable feedback controller $\mathscr{K}$. In Figure 5.1, we depict a general scheme; $y(t)$ is the feedback signal to the input of $\mathscr{K}$, possibly together with the state of $\mathscr{P}$.

Remark. We underline that $\mathscr{P}$ is a representation of an optimization algorithm; therefore, the state is known as well as the output, which is different from physical systems where the observation of the state may be critical.

The aim of the development presented in the following chapters is to address the following problem.

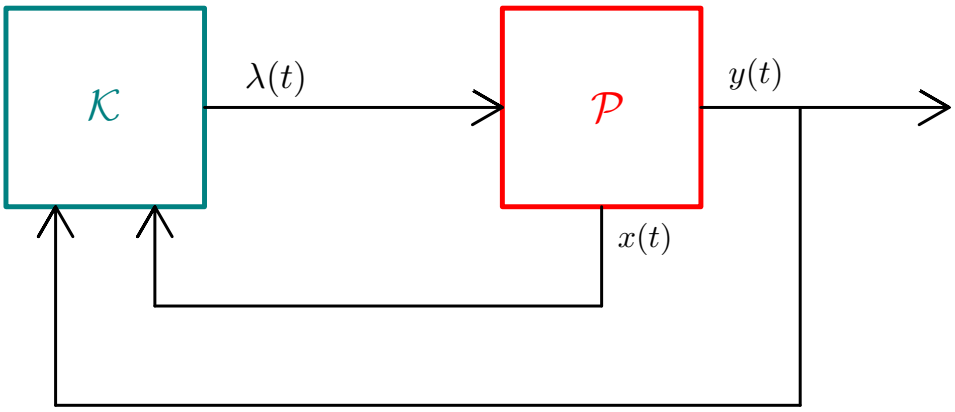

Fig. 5.1 Structure of the proposed controlled multipliers optimization approach. The state and the output of $\mathscr{P}$ defined in (5.2) are fed back to the controller $\mathscr{K}$, whose output is the vector of the Lagrange multipliers.

Problem 5.1.1. *Design a feedback controller $\mathscr{K}$ for $\mathscr{P}$ such that*

$$\begin{aligned} \lim_{t\to\infty} x(t) &= x^\star, \\ \lim_{t\to\infty} y(t) &= 0 \end{aligned} \tag{5.3}$$

for some $x^\star \in \mathbb{R}^n$.

The feedback controller $\mathscr{K}$ should provide convergence of the $\mathscr{P}$ state variables towards the a stationary point. From a controller design perspective, we ensure that the feedback system enjoys such convergence behavior by stabilizing the system. As a consequence, in the following development, we are mainly interested in ensuring the stability of the optimization problem minima.

5.2 Inequality constrained problems

Let $f, h_1, \ldots, h_m : \mathbb{R}^n \to \mathbb{R}$ be smooth functions. We consider the constrained optimization problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \\ &h(x) \leq 0_m, \end{aligned} \tag{5.4}$$

where "$\leq$" denotes the componentwise inequality.

We consider the following augmented Lagrangian, used, e.g., in [114, 103], to deal with the inequality constraints:

$$\mathscr{L}_a(x,\lambda) = f(x) + g(x,\lambda), \tag{5.5}$$

where we define $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ as

$$\begin{aligned} g(x,\lambda) &= \sum_{j=1}^{m} g_j(x,\lambda_j) \quad \text{with} \\ g_j(x,\lambda_j) &= \begin{cases} \lambda_j h_j(x) + \dfrac{\rho}{2} h_j^2(x) & \text{if } h_j(x) \geq -\dfrac{\lambda_j}{\rho}; \\ -\dfrac{1}{2\rho}\lambda_j^2 & \text{otherwise} \end{cases} \end{aligned} \tag{5.6}$$

and $\rho > 0$ is a scalar design hyperparameter. We notice that, with this definition, g is continuously differentiable.

The function $g_j(x,\lambda_j) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ penalizes the constraint violation. We notice that it is continuous and has a continuous gradient for each $(x,\lambda_j) \in \mathbb{R}^{n+1}$.

We remark that (5.5) is not the standard Lagrangian for inequality-constrained problems defined in [114, Chapter 3], i.e.,

$$\mathscr{L}_{\text{ineq}}(x,\lambda) = f(x) + \sum_{j=1}^{m} \lambda_j g_j(x) \tag{5.7}$$

used, e.g., in [115, 116]. As noticed in [103], the saddle point of (5.5)-(5.6) corresponds to the saddle point of the standard Lagrangian.

Similarly to the equality-constrained case, we define a dynamical system $\mathscr{P}$ having state $x(t) \in \mathbb{R}^n$, input $\lambda(t) \in \mathbb{R}^m$ and output $y(t) \in \mathbb{R}^m$. In this case, we define $\mathscr{P}$ by the equations

$$\mathscr{P} : \begin{cases} \dot{x}(t) = -\nabla_x \mathscr{L}_a(x(t),\lambda(t)) = -\nabla_x f(x) - \displaystyle\sum_{j=1}^{m} \nabla_x g_j(x,\lambda_j) \\ y(t) = \nabla_\lambda \mathscr{L}_a(x(t),\lambda(t)) = \nabla_\lambda g(x,\lambda). \end{cases} \tag{5.8}$$

The following result holds

Lemma 5.2.1. *An equilibrium point $(x^\star, \lambda^\star)$ of $\mathscr{P}$ satisfies the KKT conditions (4.13) for the optimization problem (5.4) if and only if $\nabla_\lambda \mathscr{L}_a(x^\star, \lambda^\star) = 0$.*

Proof. $\nabla_\lambda g_j(x^\star, \lambda^\star) = 0$ is equivalent to

$$\begin{cases} h_j(x^\star) = 0 & \text{if } h_j(x^\star) \geq -\dfrac{\lambda_j^\star}{\rho} \\ -\dfrac{\lambda_j^\star}{\rho} = 0 & \text{if } h_j(x^\star) \leq -\dfrac{\lambda_j^\star}{\rho} \end{cases} \tag{5.9}$$

In turn, this is equivalent to saying that for each constraint i, either $\lambda_i^\star = 0$ and $h_i(x^\star) \leq 0$ or $\lambda_i^\star \geq 0$ and $h_i(x^\star) = 0$. These corresponds to the KKT conditions (4.13b)-(4.13d), i.e., complementary slackness $\lambda_i^\star h_i(x^\star) = 0$, primal feasibility $h_i(x^\star) \leq 0$, and dual feasibility $\lambda_i^\star \geq 0$.

We notice that

$$\nabla_x g(x^\star, \lambda^\star) = \begin{cases} (\lambda_j^\star + \rho h_j(x^\star))\nabla_x h_j(x^\star) & \text{if } h_j(x^\star) \geq -\dfrac{\lambda_j^\star}{\rho} \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

From (5.9), it follows that $h_j(x^\star) = 0$ when $h_j(x^\star) \geq -\frac{\lambda_j^\star}{\rho}$. Moreover, $\lambda_j^\star = 0$ when $h_j(x^\star) < -\frac{\lambda_j^\star}{\rho}$. Then, we conclude that

$$\nabla_x g_j(x^\star, \lambda^\star) = \lambda_j^\star \nabla_x h_j(x^\star). \tag{5.11}$$

Using (5.11), we notice the definition of equilibrium point for $\mathscr{P}$

$$-\nabla_x f(x^\star) - \sum_{j=1}^{m} \nabla_x g_j(x^\star, \lambda_j^\star) = 0 \tag{5.12}$$

is equivalent to KKT condition (4.13a), i.e.,

$$\nabla_x f(x^\star) + \sum_{j=1}^{m} \lambda_j^\star \nabla_x h_j(x^\star) = 0. \tag{5.13}$$

Since (4.13a)-(4.13d) are fulfilled, then any equilibrium point of (5.4) that satisfies $\nabla_\lambda g(x^\star, \lambda^\star) = 0$ is a point that satisfies the KKT conditions of (5.4). Conversely,

any stationary point that satisfies the KKT conditions of (5.4) and $\nabla_\lambda g(x^\star, \lambda^\star) = 0$, also satisfies (5.12) and therefore it is an equilibrium point of $\mathscr{P}$. □

Similarly to the equality-constrained case, we resort to Lemma 5.2.1 to recast the problem of finding a point that satisfies KKT conditions for the optimization problem (5.4) into a control problem.

Problem 5.2.1. *Design a feedback controller $\mathscr{K}$ for $\mathscr{P}$ such that*

$$\lim_{t\to\infty} x(t) = x^\star, \tag{5.14a}$$

$$\lim_{t\to\infty} y(t) = 0. \tag{5.14b}$$

for some $x^\star \in \mathbb{R}^n$.

Notice that, formally, the problem is equivalent to the equality-constrained counterpart, i.e., Problem 5.1.1. The main non-trivial difference between Problem 5.1.1 and Problem 5.2.1 is that in 5.2.1 the definition of $y(t)$ is not just a function of the state variables, but $y(t) = \nabla_\lambda g(x, \lambda)$ also explicitly (i.e., statically) depends on the inputs $\lambda(t)$. This feature renders applying certain specific controller design methods, such as feedback linearization, infeasible.

Similarly to the equality-constrained case, from a controller design perspective, we are interested in guaranteeing the stability of the problem minima.

Chapter 6

The PI-CMO algorithm

This chapter investigates the first possible strategy to design $\mathscr{K}$ for Problem 5.1.1. To achieve this goal, we propose to apply a proportional-integral (PI) control action on $y(t) = h(x(t))$. The PI control is widespread in industrial and engineering applications, thanks to its effectiveness in regulating many processes by tuning two parameters only.

In our framework, the PI control law is as follows:

$$\lambda(t) = K_p y(t) + K_i \int_0^t y(\tau)\,\mathrm{d}\tau, \tag{6.1}$$

where $K_p \in \mathbb{R}$ and $K_i \in \mathbb{R}$ are the coefficients of the proportional and integral terms, respectively. We depict the corresponding feedback scheme in Figure 6.1.

As a consequence, $\mathscr{K}$ is a dynamic system described by the differential equation

$$\begin{aligned} \dot{\lambda}(t) &= K_p \frac{\mathrm{d}}{\mathrm{dt}} y(t) + K_i y(t) \\ &= K_p J_h(x(t))\dot{x}(t) + K_i y(t). \end{aligned} \tag{6.2}$$

In the following, we drop the variable t in long formulas, namely $x = x(t)$, $\dot{x} = \dot{x}(t)$, $\lambda = \lambda(t)$ and $\dot{\lambda} = \dot{\lambda}(t)$. Given the definition of $\dot{x}$ in (5.2), the closed-loop dynamics is

$$\begin{aligned} \dot{x} &= -\nabla f(x) - J_h(x)^\top \lambda \\ \dot{\lambda} &= -K_p J_h(x) \left[\nabla f(x) + J_h(x)^\top \lambda \right] + K_i h(x). \end{aligned} \tag{6.3}$$

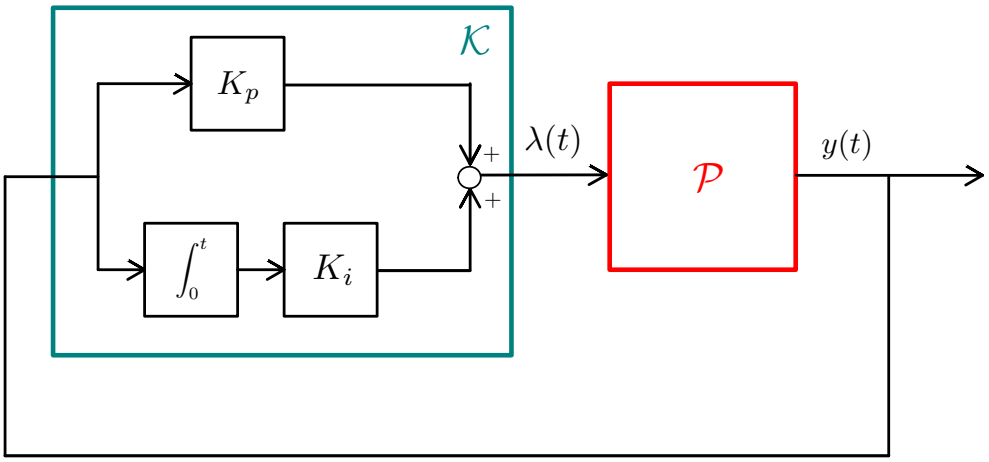

Fig. 6.1 Structure of the PI control for CMO. We feedback the output $y(t)$ to the controller $\mathscr{K}$, which applies proportional and integral actions.

or, equivalently,

$$\begin{aligned} \dot{x} &= -\nabla_x \mathscr{L}(x,\lambda) \\ \dot{\lambda} &= -K_p J_h(x) \nabla_x \mathscr{L}(x,\lambda) + K_i \nabla_\lambda \mathscr{L}(x,\lambda). \end{aligned} \tag{6.4}$$

In the remainder of this thesis, we shall refer to the closed-loop system (6.3) as PI-CMO dynamics or simply PI-CMO.

For any equilibrium point $(x^\star, \lambda^\star)$, from (6.3) we get the condition $h(x^\star) = 0$. Therefore, according to Lemma 5.1.1, each equilibrium point corresponds to a stationary point in the case of PI control.

In equation (6.1), we consider scalar K_p and K_i to reduce the number of design parameters and to simplify the convergence analysis. The extension to the most general case where K_p and K_i are matrix gains in $\mathbb{R}^{m,m}$ is straightforward.

6.1 Convergence result

This section proves that the PI-CMO dynamics is globally exponentially convergent in the convex setting with affine constraints. The proposed proof relies on Lyapunov stability theory; see Chapter 3 for an introduction to the main theorems. Moreover, we analyze its convergence rate. We consider the same assumptions as [103], allowing us to compare the results thoroughly to PDGD.

Assumption 1. *f is strongly convex and twice differentiable on $\mathbb{R}^n$.*

Assumption 2. *h is affine, i.e., $h(x) = Cx + d$, $C \in \mathbb{R}^{m,n}$, $d \in \mathbb{R}^m$. Moreover, C is full rank and there exist $0 < \alpha_1 \leq \alpha_2$ such that*

$$\alpha_1 I_m \preceq CC^\top \preceq \alpha_2 I_m. \tag{6.5}$$

The assumption that C is full rank guarantees that the feasible set $\{x \in \mathbb{R}^n : Cx + d = 0\}$ is non-empty and that there are no linearly dependent constraints. Moreover, it ensures that each point $x \in \mathbb{R}^n$ is regular, therefore ensuring that $x^\star$ is regular: a condition required in Theorem 4.1.2.

We define

$$z(t) := (x(t)^\top, \lambda(t)^\top)^\top \tag{6.6}$$

and

$$z^\star := (x^{\star\top}, \lambda^{\star\top})^\top \tag{6.7}$$

is the equilibrium point of (6.3), which corresponds to a saddle point of $\mathscr{L}(x, \lambda)$. Assumptions 1 and 2 guarantee the existence and the uniqueness of $z^\star$.

Before establishing the main result, we review the following lemma.

Lemma 6.1.1. (Affine form of $\nabla f(x) - \nabla f(x^\star)$. [103, Lemma 1]). *Let $f(x)$ be a strongly convex function and $x^\star \in \mathbb{R}^n$. There exists a symmetric $B(x) \in \mathbb{R}^{n,n}$ satisfying $\beta_1 I_n \preceq B(x) \preceq \beta_2 I_n$ for some $0 < \beta_1 < \beta_2$ such that*

$$\nabla f(x) - \nabla f(x^\star) = B(x)(x - x^\star). \tag{6.8}$$

Moreover, according to [103], the constants β_1, β_2 satisfy

$$\beta_1 I \leq \nabla^2 f(x^\star + t(x - x^\star)) \leq \beta_2 I, \quad \forall t \in [0, 1]. \tag{6.9}$$

where $\nabla^2 f(x)$ is the Hessian matrix of $f(x)$. Equation (6.9) relates the constants β_1 and β_2 with the second-order information of $f(x)$. If $f(x)$ is quadratic, i.e., $f(x) = \frac{1}{2} x^\top W x$ with $W \in \mathbb{R}^{n,n}, W \succ 0$, then β_1 and β_2 are the minimum and maximum eigenvalues of W, respectively. If $f(x)$ is not quadratic, we can evaluate β_1 and β_2 as outlined in [117], which presents a technique for estimating tight bounds of the eigenvalues of $\nabla^2 f(x)$. The estimates of β_1, β_2 obtained in this way are useful to determine the convergence rate of PI-CMO, according to the following theorem.

In the following, we write $B = B(x)$ for brevity.

Theorem 6.1.1 (Global exponential convergence of PI-CMO)**.** *Let assumptions 1 and 2 hold. Given $K_i > 0$ and $K_p > 0$, if*

$$K_p < \frac{2K_i}{\beta_2}, \tag{6.10}$$

then there exist real positive constants c_1 and c_2 such that

$$\|x(t) - x^\star\|_2 \leq c_1 e^{-\frac{1}{2}\mu t}, \quad \|\lambda(t) - \lambda^\star\|_2 \leq c_2 e^{-\frac{1}{2}\mu t} \tag{6.11}$$

where

$$\mu = \min\left\{K_p\alpha_1, 2\beta_1 - \frac{K_p}{K_i}\beta_1\beta_2\right\} > 0. \tag{6.12}$$

Proof. First of all, we define the candidate Lyapunov function

$$V\big(z(t)\big) = \big(z(t) - z^\star\big)^\top P\big(z(t) - z^\star\big) \tag{6.13}$$

where

$$P := \begin{pmatrix} K_i I_n & 0 \\ 0 & I_m \end{pmatrix} \in \mathbb{R}^{m+n,m+n}. \tag{6.14}$$

If we prove that

$$\dot{V}\big(z(t)\big) \leq -\mu V\big(z(t)\big) \tag{6.15}$$

for some $\mu > 0$, then the theorem statement holds. Therefore, in the following, we focus on conditions that guarantee (6.15).

We start with some preliminary computations. Since $\nabla_x \mathscr{L}(x^\star, \lambda^\star) = 0$, $\nabla_\lambda \mathscr{L}(x^\star, \lambda^\star) = 0$ and $J_h(x) = C$, we have

$$\begin{aligned} \nabla_x \mathscr{L}(x,\lambda) &= \nabla_x \mathscr{L}(x,\lambda) - \nabla_x \mathscr{L}(x^\star,\lambda^\star) \\ &= \nabla f(x) - \nabla f(x^\star) + J_h(x)^\top \lambda - J_h(x^\star)^\top \lambda^\star \\ &= \nabla f(x) - \nabla f(x^\star) + C^\top(\lambda - \lambda^\star) \end{aligned} \tag{6.16}$$

and

$$\begin{aligned} \nabla_\lambda \mathscr{L}(x,\lambda) &= \nabla_\lambda \mathscr{L}(x,\lambda) - \nabla_\lambda \mathscr{L}(x^\star,\lambda^\star) \\ &= Cx + d - (Cx^\star + d) = C(x - x^\star). \end{aligned} \tag{6.17}$$

By using (6.4), (6.8), (6.16) and (6.17), we obtain

$$\begin{aligned}\dot{z}(t) &= \begin{pmatrix} \dot{x}(t) \\ \dot{\lambda}(t) \end{pmatrix} \\ &= \begin{pmatrix} -B(x-x^\star) - C^\top(\lambda-\lambda^\star) \\ K_iC(x-x^\star) - K_pC[B(x-x^\star) + C^\top(\lambda-\lambda^\star)] \end{pmatrix} \\ &= \begin{pmatrix} -B & -C^\top \\ K_iC - K_pCB & -K_pCC^\top \end{pmatrix} \big(z(t)-z^\star\big).\end{aligned} \tag{6.18}$$

Let us define

$$G := \begin{pmatrix} -B & -C^\top \\ K_iC - K_pCB & -K_pCC^\top \end{pmatrix} \tag{6.19}$$

so that

$$\dot{z}(t) = G\big(z(t)-z^\star\big).$$

Then,

$$\begin{aligned}\dot{V}\big(z(t)\big) &= \dot{z}(t)^\top P\big(z(t)-z^\star\big) + \big(z(t)-z^\star\big)^\top P\dot{z}(t) \\ &= \big(z(t)-z^\star\big)^\top \big(G^\top P + PG\big)\big(z(t)-z^\star\big).\end{aligned} \tag{6.20}$$

Therefore, a sufficient condition for $\dot{V}\big(z(t)\big) \le -\mu V\big(z(t)\big)$, see (6.15), is

$$-G^\top P - PG - \mu P \succeq 0. \tag{6.21}$$

Consequently, our next goal is to provide sufficient conditions for (6.21). We compute

$$PG = \begin{pmatrix} -K_iB & -K_iC^\top \\ K_iC - K_pCB & -K_pCC^\top \end{pmatrix} \tag{6.22}$$

while $G^\top P = (PG)^\top$. Hence,

$$\begin{aligned}&-G^\top P - PG - \mu P \\ &= \begin{pmatrix} 2K_iB - K_i\mu I & K_pBC^\top \\ K_pCB & 2K_pCC^\top - \mu I \end{pmatrix} \\ &\succeq \begin{pmatrix} 2K_iB - K_i\mu I & K_pBC^\top \\ K_pCB & K_pCC^\top \end{pmatrix}\end{aligned} \tag{6.23}$$

where the last step derives from $2K_pCC^\top - \mu I \succeq CC^\top$ which holds for $\mu \leq K_p\alpha_1$. Since $CC^\top \succ 0$ is invertible from (6.5) in Assumption 2, we can apply the Schur complement argument: the matrix

$$\begin{pmatrix} 2K_iB - K_i\mu I & K_pBC^\top \\ K_pCB & K_pCC^\top \end{pmatrix}$$

is positive semidefinite if and only if

$$2K_iB - K_i\mu I - K_pBC^\top \frac{1}{K_p}(CC^\top)^{-1}K_pCB \succeq 0. \tag{6.24}$$

Moreover, since $CC^\top$ is invertible, then $C^\top(CC^\top)^{-1}C \preceq I$. Therefore, a sufficient condition for (6.24) is

$$(2K_iI - K_pB)B \succeq K_i\mu I. \tag{6.25}$$

Under assumption (6.10), (6.25) holds if

$$(2K_i - K_p\beta_2)\beta_1 \geq K_i\mu \tag{6.26}$$

which is equivalent to

$$\mu \leq 2\beta_1 - \frac{K_p}{K_i}\beta_1\beta_2. \tag{6.27}$$

This completes the proof. □

According to (6.13)-(6.14), the considered Lyapunov function is

$$V(x,\lambda) = K_i\|x - x^\star\|_2^2 + \|\lambda - \lambda^\star\|_2^2.$$

We highlight the role of the parameter K_i, which tunes the weight assigned to the terms in x compared to the terms in λ. According to Theorem 6.1.1, a larger K_i increases the bound on the convergence rate μ, i.e., the weight given to the terms in x must be sufficiently large compared to the terms in λ.

6.2 Relation with the literature

This section compares PI-CMO (6.3) with PDGD (4.34).

It is worth noticing that by considering a purely integral control, i.e., $K_p = 0$, we obtain the dynamics

$$\begin{aligned} \dot{x} &= -\nabla_x \mathscr{L}(x,\lambda) \\ \dot{\lambda} &= K_i \nabla_\lambda \mathscr{L}(x,\lambda). \end{aligned} \tag{6.28}$$

which corresponds to PDGD defined in Chapter 3, Equation (4.34) (see also, e.g., [103, Equations (2a)-(2b)]), where the symbol η is used instead of K_i. In other words, we can interpret PDGD as applying an integral control to the Lagrange multipliers, which recast PDGD into the proposed control-theoretic framework. At the same time, the proposed PI-CMO extends PDGD to a novel family of continuous-time optimization algorithms.

According to [103], PDGD is globally exponentially convergent with rate $\frac{1}{2}\mu_{PDGD}$

$$\mu_{PDGD} = \min\left\{\frac{\eta\alpha_1}{4\beta_2}, \frac{\alpha_1\beta_1}{4\alpha_2}\right\} \tag{6.29}$$

Since η is a design parameter, we can set $\eta \geq \frac{\beta_1\beta_2}{\alpha_2}$ it so that μ_{PDGD} saturates to $\frac{\alpha_1\beta_1}{4\alpha_2}$.

The following result holds.

Corollary 6.2.1. *For $\varepsilon \in \left(0, 1 - \frac{\alpha_1}{8\alpha_2}\right)$, let*

$$K_p = \varepsilon\frac{2K_i}{\beta_2} \quad \textit{and} \quad K_i > \frac{1}{\varepsilon}\frac{\beta_1\beta_2}{8\alpha_2}. \tag{6.30}$$

Then

$$\mu > \mu_{PDGD} \tag{6.31}$$

i.e., PI-CMO enjoys a faster convergence rate compared to PDGD [103].

Proof. By replacing $K_p = \varepsilon\frac{2K_i}{\beta_2}$ in (6.12), we obtain

$$\mu = \min\left\{2\varepsilon K_i\frac{\alpha_1}{\beta_2}, 2\beta_1(1-\varepsilon)\right\}.$$

Now, $\varepsilon \in \left(0, 1 - \frac{\alpha_1}{8\alpha_2}\right)$ implies $2\beta_1(1-\varepsilon) > \frac{\alpha_1\beta_1}{4\alpha_2}$, while $K_i > \frac{1}{\varepsilon}\frac{\beta_1\beta_2}{8\alpha_2}$ implies $2\varepsilon K_i\frac{\alpha_1}{\beta_2} > \frac{\alpha_1\beta_1}{4\alpha_2}$. This proves the statement because $\mu_{PDGD} \leq \frac{\alpha_1\beta_1}{4\alpha_2}$. □

Finally, we notice that in the convergence proof of PDGD in [103], the Lyapunov function is defined on a non-diagonal matrix P, making interpreting the parameters more difficult when compared with the given interpretation of (6.13)-(6.14).

6.2.1 Illustrative example

To complete the analysis, we present an example that illustrates how the tuning of K_i and K_p may affect the convergence rate. We compare the results to the case $K_p = 0$, namely PDGD. Let us consider the simple, univariate optimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}} \ & \frac{1}{2} w x^2 \\ \text{s.t.} \ & x = 0 \end{aligned} \tag{6.32}$$

where $w > 0$. By applying the PI control, the corresponding closed-loop dynamics is

$$\begin{aligned} \dot{x} &= -wx - \lambda \\ \dot{\lambda} &= (K_i - K_p w)x - K_p \lambda . \end{aligned} \tag{6.33}$$

Equation (6.33) is a second-order CT linear time-invariant system

$$\begin{pmatrix} \dot{x} \\ \dot{\lambda} \end{pmatrix} = \mathrm{A} \begin{pmatrix} x \\ \lambda \end{pmatrix} \tag{6.34}$$

with

$$\mathrm{A} := \begin{pmatrix} -w & -1 \\ K_i - K_p w & -K_p \end{pmatrix}. \tag{6.35}$$

The eigenvalues of A are

$$\frac{-(K_p + w) \pm \sqrt{(K_p + w)^2 - 4K_i}}{2}. \tag{6.36}$$

For any $K_i > 0$, the eigenvalues are either real and negative or complex with negative real parts. In particular, if $K_i \geq (K_p + w)^2/4$, K_i contributes only to the imaginary part; therefore, it does not impact the convergence rate.

According to the authors of [103], although $\eta > 0$ can be arbitrarily large for PDGD, increasing η beyond a certain threshold does not lead to a faster decaying

rate. In our example, if we choose $K_p = 0$, we obtain PDGD with $\eta = K_i$ and, from (6.36), we see that if $K_i \geq w^2/4$, then K_i has no impact on the real parts of the eigenvalues. This fact explains the observation in [103].

On the other hand, in the PI control method, we can tune $K_p > 0$ to enhance the convergence rate, provided that the conditions of Theorem 6.1.1 are satisfied, which is a benefit compared to PDGD.

6.2.2 PI control in non-convex quadratic optimization with linear constraints

To conclude this section, we analyze some properties of the PI control method for quadratic optimization with linear constraints. In particular, we prove that optimization problems exist with non-convex cost functions in which the PI control method converges to a stationary point while PDGD is divergent.

We consider

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \frac{1}{2} x^\top W x \\ \text{s.t.} \quad Cx + d = 0 \end{aligned} \tag{6.37}$$

where $W \in \mathbb{R}^{n,n}$ is symmetric and $C \in \mathbb{R}^{m,n}$ and $d \in \mathbb{R}^m$, with invertible $CC^\top$.

Since the cost function is quadratic and the constraints are linear, the closed-loop dynamics with the PI control corresponds to the linear time-invariant system

$$\begin{pmatrix} \dot{x} \\ \dot{\lambda} \end{pmatrix} = \mathrm{A} \begin{pmatrix} x \\ \lambda \end{pmatrix} + \begin{pmatrix} 0 \\ K_i d \end{pmatrix} \tag{6.38}$$

with

$$\mathrm{A} := \begin{pmatrix} -W & -C^\top \\ K_i C - K_p C W & -K_p C C^\top \end{pmatrix} \tag{6.39}$$

Therefore, if A is Hurwitz, then the system is asymptotically stable, and by construction, the output $y(t) = Cx(t) + d$ is regulated to zero. Thus, for the PI control method, W does not need to be positive definite, i.e., the system does not need to be strongly convex.

As an example, let us consider $W = \text{diag}(1,-1)$ and $C = (0,2)$. Since W is indefinite, the quadratic cost function is not convex. Notice, however, that by replacing the equality constraint into the objective, we obtain the equivalent problem $\min_x \frac{1}{2}(x_1^2 - \frac{d^2}{4})$, which is strongly convex and therefore, we expect a unique solution. The eigenvalues of the dynamic matrix of PI-CMO are -1 and $\frac{1-4K_p \pm \sqrt{(1-4K_p)^2-16K_i}}{2}$. If $K_p = 0$, all the eigenvalues have positive real parts for all K_i. In other terms, PDGD is always unstable. Instead, for $K_p > \frac{1}{4}$, all the eigenvalues have negative real part for all $K_i > 0$.

In conclusion, problems exist with non-convex functions where the PI control method converges to the unique minimum point as long as we provide a suitable tuning of K_p. In contrast, PDGD is divergent for any hyperparameter choice. This observation encourages future study of the PI control method in non-convex optimization.

6.3 Numerical examples

This section presents two numerical examples to demonstrate the effectiveness of the proposed PI-CMO.

6.3.1 PI-CMO vs PDGD in convex optimization

In the first example, we resort to the quadratic optimization problem with linear constraints proposed in [103, Section IV.A]. Specifically, we consider

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \frac{1}{2} x^\top W x \\ &\text{s.t.} \quad Cx + d = 0 \end{aligned} \tag{6.40}$$

where $W = 10I + W_0 W_0^\top \in \mathbb{R}^{n,n}$ is positive definite; $W_0 \in \mathbb{R}^{n,n}$, $C \in \mathbb{R}^{m,n}$ and $d \in \mathbb{R}^m$ have independent and normally distributed components with zero mean and unitary variance.

To solve this problem, we implement PI-CMO and compare it to PDGD [103]. Since the cost function is quadratic and the constraints are linear, the PI-CMO dynamics is linear time-invariant, as illustrated in Section 6.2.2.

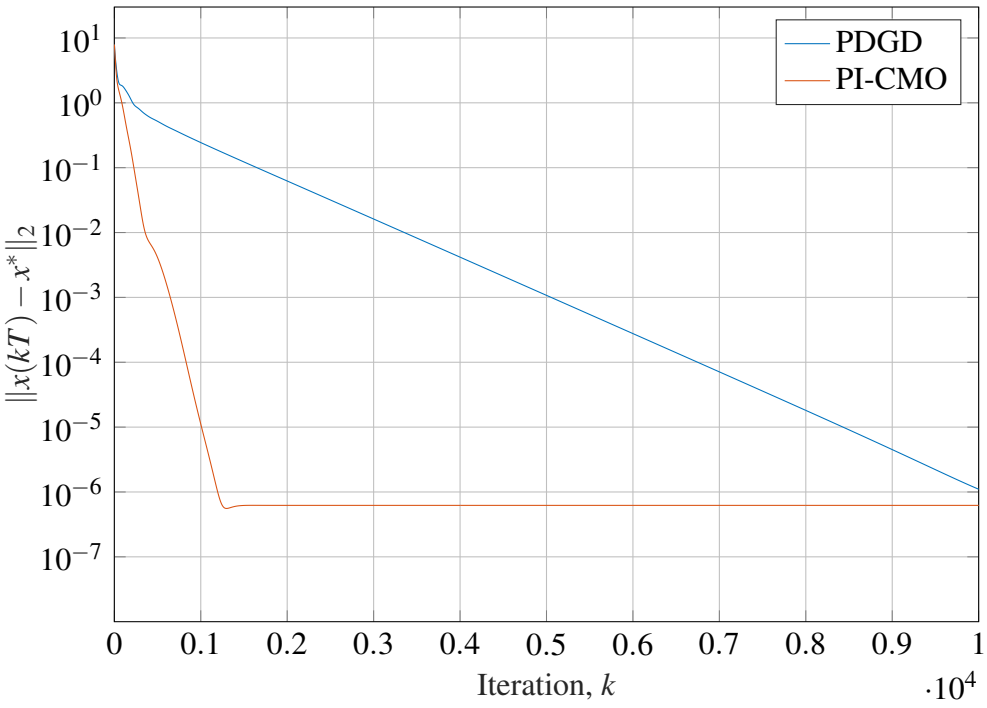

Fig. 6.2 Comparison of PDGD and PI-CMO, with $m = 26$ constraints.

Corollary 6.2.1 shows that the convergence rate bound of PI-CMO is favorable to PDGD for strongly convex problems with linear constraints. In this example, we support these theoretical results with a numerical analysis of the convergence speed.

We consider $n = 50$ variables and $m \in [2, 26]$ constraints. For PDGD, we set $\eta = \frac{\beta_1 \beta_2}{\alpha_2}$ to obtain the best convergence rate bound; see (6.29). For PI-CMO, we set K_i and K_p to meet the conditions of Corollary 6.2.1, specifically

$$K_i = \frac{50}{\varepsilon} \frac{\beta_1 \beta_2}{8\alpha_2}, \quad K_p = \varepsilon \frac{2K_i}{\beta_2} \tag{6.41}$$

where $\varepsilon = \frac{4}{5}\left(1 - \frac{\alpha_1}{8\alpha_2}\right)$.

For all the algorithms, we simulate the dynamics using forward Euler discretization with step size $T = 10^{-3}$ s, which guarantees stability, see [103, Section III.C] for details. In Figure 6.3, we show the distance from the optimum $\|x(kT) - x^\star\|_2$ as a function of the iteration k for PDGD and PI-CMO in the case $m = 26$ constraints, for one random realization of the data. As expected from the theoretical results, the distance from the optimum decreases faster in the case of PI-CMO.

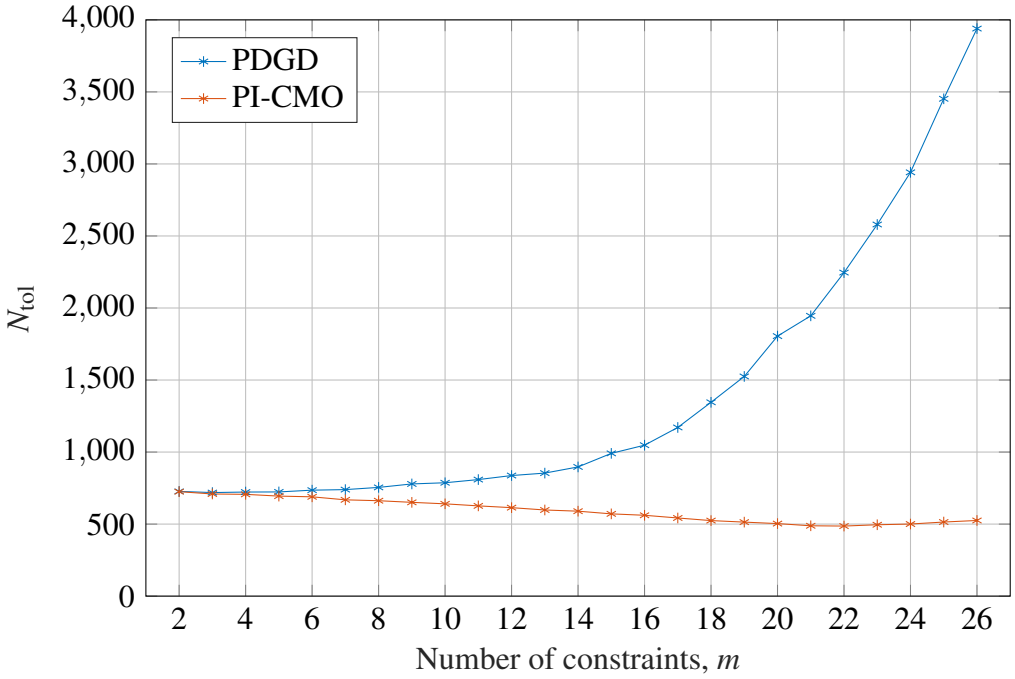

Fig. 6.3 Number of iterations to converge for PI-CMO and PDGD in a convex quadratic example, with m constraints. The results are averaged over 200 random runs.

Next, we perform 200 random runs and evaluate the average number of iterations N_ε required to converge to the desired minimum (with tolerance $\|x(kT) - x^\star\|_2 \leq \varepsilon = 10^{-3}$) as a function of m. We show the average N_ε against m in Figure 6.3. As expected from the theoretical results, PI-CMO requires fewer iterations on average than PDGD for all the considered values of m. Moreover, we notice that while N_ε increases with m for PDGD, it is almost constant with respect to m in the case of PI-CMO.

6.3.2 Shidoku puzzle

Shidoku is a 4x4 version of the popular 9x9 Sudoku puzzle. Given an initial scheme, as reported in Figure 6.4, the aim is to fill the empty cells with integers $x_{i,j} \in \{1,2,3,4\}$ such that each row, each column, and each 2x2 corner block contain the integers $1,2,3,4$. We can formulate the game as a set of polynomial equations on the values $x_{i,j}$ of each cell (i,j), $i,j = 1,\dots,4$. First of all, $x_{i,j} \in \{1,2,3,4\}$ is guaranteed by $\prod_{h=1}^{4}(x_{i,j} - h) = 0$. Then, we obtain no repetition in groups of 4 cells by imposing the product equal to 24 and the sum equal to 10. In summary, given the

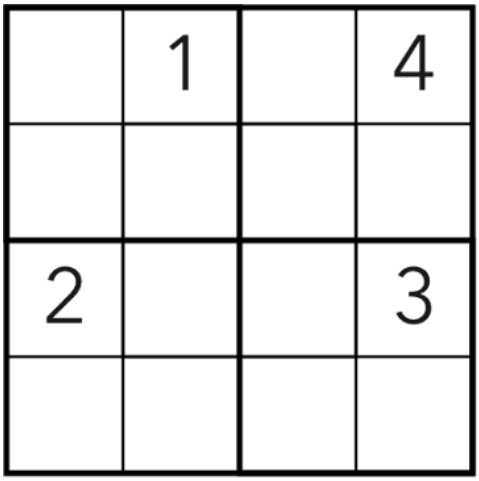

Fig. 6.4 Shidoku puzzle to be solved by PI-CMO algorithm.

corner blocks

$$B_1 = \{(1,1),(1,2),(2,1),(2,2)\}$$
$$B_2 = \{(1,3),(1,4),(2,3),(2,4)\}$$
$$B_3 = \{(3,1),(3,2),(4,1),(4,2)\}$$
$$B_4 = \{(3,3),(3,4),(4,3),(4,4)\}.$$

we have

Columns: for $j = 1,\ldots,4$,

$$\sum_{i=0}^{4} x_{ij} = 10, \quad \prod_{i=0}^{4} x_{ij} = 24. \tag{6.42a}$$

Rows: for $i = 1,\ldots,4$,

$$\sum_{j=0}^{4} x_{ij} = 10, \quad \prod_{j=0}^{4} x_{ij} = 24. \tag{6.42b}$$

Blocks: for $k = 1,\ldots,4$,

$$\sum_{(i,j)\in B_k} x_{ij} = 10, \quad \prod_{(i,j)\in B_k} x_{ij} = 24. \tag{6.42c}$$

Restriction to integers: for $i, j = 1,\ldots,4$,

$$\prod_{h=1}^{4} (x_{ij} - h) = 0. \tag{6.42d}$$

and

$$\text{Initial conditions as in Figure 6.4:}$$
$$x_{1,2} = 1 \quad x_{1,4} = 4 \quad x_{3,1} = 2 \quad x_{3,4} = 3. \tag{6.42e}$$

Equations (6.42) represent non-convex polynomial constraints. We can solve the corresponding optimization problem through the proposed feedback control framework if we associate them with any constant cost function. In particular, we use the PI-CMO dynamics to test its convergence and effectiveness in non-convex problems.

As to PI-CMO, we set $K_i = 1, K_p = 0.1$ and we generate random initial conditions according to $x_{i,j}(0) = |\xi_{i,j}|, \xi_{i,j} \sim \mathcal{N}(0,1)$, for each $i, j = 1, \ldots, 4$ and $\lambda_k(0) \sim \mathcal{N}(0,1), k = 1, \ldots, m$, where $\mathcal{N}(0,\sigma)$ denotes the normal distribution with zero mean and variance σ^2. We integrate the ordinary differential equations that describe the closed-loop dynamics thanks to the MATLAB ode45 solver in the time interval $[0, 100]$ seconds, corresponding to approximately 151700 iterations. We perform 20 runs with different random initial conditions. The PI control method is always convergent in the given time interval. We show an instance in Figure 6.5, where we depict the evolution of the optimization variables for six equispaced sampling instants between the random initialization and the convergence to the correct solution at iteration 91027, shown in Figure 6.6.

In conclusion, this test shows that PI-CMO is convergent even in a non-convex problem that does not satisfy Assumptions 1 and 2. For further investigation, we compare PI-CMO against two state-of-the-art approaches for non-convex constrained optimization, interior-point method (IPM) and sequential quadratic programming, through the fmincon function in the MATLAB optimization toolbox. We perform 20 runs with random initial conditions for the two of them.
We observe that IPM fails in all the runs due to numerical issues. More precisely, the linear system of KKT conditions to solve at each iteration is poorly conditioned, which affects the solution; see, e.g., [41, Chapter 19] for details. On the other hand, sequential quadratic programming converges to an infeasible point in all the runs.

0.8	1	0.0	4
0.6	0.3	0.3	1.7
2	1.3	0.6	3
0.2	0.6	0.6	0.8

1.5	1	3.1	4
3.0	4.1	1.3	1.5
2	1.0	4.0	3
3.1	4.3	1.4	1.3

1.7	1	3.1	4
4.1	3.3	1.0	1.7
2	1.8	3.1	3
2.4	3.9	2.5	1.1

2.0	1	2.9	4
4.8	2.5	1.7	1.2
2	3.9	0.9	3
1.0	2.4	5.3	1.9

3.1	1	2.0	4
3.6	2.1	3.2	1.0
2	3.4	1.0	3
1.0	3.3	3.7	2.1

2.9	1	2.0	4
4.1	1.9	2.9	1.0
2	4.1	1.0	3
1.0	2.9	4.1	2.0

Fig. 6.5 PI-CMO method solves the Shidoku puzzle. Evolution of the solution to the optimization variables at six equispaced sampling instants between initialization and convergence step 91027, from top left to bottom right.

3	1	2	4
4	2	3	1
2	4	1	3
1	3	4	2

Fig. 6.6 PI-CMO method solves Shidoku puzzle. Final correct solution.

Chapter 7

Modified PI-CMO for inequality-constrained problems

In this chapter, we take a proportional-integral (PI) control approach to design continuous-time (CT) optimization algorithms to solve strongly convex optimization problems with linear inequality constraints.

To approach the problem, we resort to the controlled multipliers optimization (CMO) framework introduced in Chapter 5, and we solve Problem 5.2.1, which, in turn, provides a solution to the optimization problem

$$\begin{aligned} &\arg\min_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t.} \quad \mathrm{h(x)} \leq 0_{\mathrm{m}}. \end{aligned} \tag{7.1}$$

The key idea is to provide an extension of the control scheme defining the PI-CMO dynamics (6.3) obtained in the equality-constrained case. In the case of inequality constraints, the most natural solution to achieve this extension would be the application of PI control action on the output $y(t) = \nabla_\lambda \mathscr{L}_a(x(t), \lambda(t))$ of the plant (5.8) defined by the augmented Lagrangian (5.5)-(5.6), i.e., to define the controller

$$\lambda(t) = K_p \nabla_\lambda \mathscr{L}_a(x(t), \lambda(t)) + K_i \int_0^t \nabla_\lambda \mathscr{L}_a(x(\tau), \lambda(\tau)) \mathrm{d}\tau. \tag{7.2}$$

In the following, we drop the variable t in long formulas, namely $x = x(t)$, $\dot{x} = \dot{x}(t)$, $\lambda = \lambda(t)$ and $\dot{\lambda} = \dot{\lambda}(t)$. Following the proposed approach, we obtain the following set of differential equations describing the resulting feedback control system:

$$\begin{aligned} \dot{x} &= -\nabla_x \mathscr{L}_a(x,\lambda) = -\nabla f(x) - \nabla_x g(x,\lambda) \\ \dot{\lambda} &= K_i \nabla_\lambda g(x,\lambda) + K_p \frac{d}{dt} \nabla_\lambda g(x,\lambda). \end{aligned} \tag{7.3}$$

where

$$\nabla_x g_j(x,\lambda_j) = \max\{\rho h_j(x) + \lambda_j, 0\} \nabla_x h_j(x). \tag{7.4}$$

We can interpret the dynamic system (7.3) as a PI control on $h_i(x)$ when a constraint is not satisfied, which pushes $h_j(x)$ towards zero. In contrast, when a constraint is satisfied, i.e., $h_j(x) \leq 0$, then $\nabla_\lambda g_j(x,\lambda_j) = \lambda_j$, thus λ_j converges to zero and it does not control x.

Even if (7.3) is a natural extension of (6.3), proving its exponential convergence is challenging. In particular, we notice that

$$\frac{d}{dt} \nabla_\lambda g_j(x,\lambda_j) = \mathbb{1}_j (J_h(x))_j^\top \dot{x} + \left(\mathbb{1}_j - 1\right) \dot{\lambda}, \tag{7.5}$$

where $\mathbb{1}_j \doteq \mathbb{1}(h_j(x) \geq -\frac{\lambda_j}{\rho})$ is the indicator function of the set

$$\left\{ x \in \mathbb{R}^n : h_j(x) \geq -\frac{\lambda_j}{\rho} \right\} \tag{7.6}$$

A non-trivial indicator function on the right-hand side of (7.5) makes the dynamics discontinuous, and, consequently, the solution flows may be non-unique. See, e.g., [118] for a detailed treatment of differential equations with a discontinuous right-hand side.

For this motivation, we modify (7.3) as follows:

$$\dot{x} = -\nabla_x \mathscr{L}(x,\lambda) = -\nabla f(x) - \nabla_x g(x,\lambda) \tag{7.7a}$$

$$\dot{\lambda} = K_i \nabla_\lambda g(x,\lambda) + K_p J_h(x) \dot{x}. \tag{7.7b}$$

The rationale behind this definition is that the desired properties from the dynamics will hopefully be preserved even if we approximate the indicator function in (7.5) with the constant identity function. Basically, we replace $\mathbb{1}_j(\cdot)$ by 1, thus, in turn, $\frac{d}{dt}\nabla_\lambda g(x,\lambda)$ by $\nabla_x h(x)\dot{x}$. In (7.3), $\dot{\lambda}$ does not depend on $\dot{x}$ when constraints are satisfied. In contrast, in (7.7), the dependence on $\dot{x}$ is always present. In the following, we refer to (7.7) as the modified PI-CMO dynamics, or M-PI-CMO for short.

Despite introducing an approximation to make the dynamics continuous, we can still interpret the dynamic system (7.7) as a feedback control system, with state $x(t)$ and control input represented by the Lagrange multipliers $\lambda(t)$. The controller is fully described by Equation (7.7b). We notice that to regulate the output of the system $\nabla_\lambda \mathscr{L}_a(x,\lambda)$ to zero is equivalent, for each $j = 1,\dots,m$, to regulate either $h_j(x)$ to zero or λ_j to zero; the latter corresponds to switching off the control input when it is not necessary to constrain the solution. Based on this observation, we might set $\lambda_j = 0$ when the control is unnecessary instead of considering convergent dynamics to zero. However, a smooth behavior for $\lambda(t)$ facilitates the convergence analysis of the overall system. In other words, by considering the term $K_p J_h(x)\dot{x}$, we control λ via state feedback even when the constraints are satisfied; this enhances the convergence, simplifies the numerical computation and simplifies the convergence analysis, as shown in the following sections.

We remark that (7.7) is not the direct extension of PI-CMO for $h(x) = 0$ reported in (6.3). Although feasible, the use of (6.3) on a fictitious output defined by

$$y_i(t) = \max\{h_i(x(t)), 0\} \tag{7.8}$$

produces switched dynamics, creating both theoretical and practical issues. On the one hand, the proof of convergence is greatly complicated due to the need to show the existence and uniqueness of flows besides their properties; on the other hand, simulation of such dynamics requires non-trivial techniques (e.g., adaptive integration step size) to handle the chattering behavior that originates when the states reach the border of the feasible set.

7.1 Convergence result

In this section, we analyze the convergence of the M-PI-CMO dynamics (7.7). We define

$$z(t) := (x(t)^\top, \lambda(t)^\top)^\top \tag{7.9}$$

and

$$z^\star := (x^{\star\top}, \lambda^{\star\top})^\top \tag{7.10}$$

is the equilibrium point of (7.7), which corresponds to a saddle point of $\mathscr{L}_a(x,\lambda)$.

The following result holds.

Lemma 7.1.1. *The equilibrium point of* (7.7) *satisfies the KKT conditions* (4.9) *for problem* (5.1).

Proof. The result directly follows from the definition of equilibrium point and the application of Lemma 5.2.1. □

As done in Chapter 6 when analyzing PI-CMO, we consider the same assumptions as [103], allowing us to compare the results thoroughly to Aug-PDGD (see Chapter 4 and [103]).

Assumption 3. *f is strongly convex and twice differentiable on $\mathbb{R}^n$.*

Assumption 4. *h is affine, i.e., $h(x) = Cx + d$, $C \in \mathbb{R}^{m,n}$, $d \in \mathbb{R}^m$. Moreover, C is full rank and there exist $0 < \alpha_1 \leq \alpha_2$ such that*

$$\alpha_1 I_m \preceq CC^\top \preceq \alpha_2 I_m. \tag{7.11}$$

The assumption that C is full rank guarantees that the feasible set is non-empty even when all constraints are active and that there are no linearly dependent constraints. Specifically, it implies that the linear independence constraint qualification condition holds, i.e., at the optimal solution $x^\star$, the constraint's Jacobian is full row rank. Moreover, it guarantees that all points $x \in \mathbb{R}^n$ are regular.

For the chapter's self-consistency, we recall [103, Lemma 1], as already stated in Chapter 6. We refer the reader to [103] and Chapter 6 for more details.

Lemma 7.1.2. (Affine form of $\nabla f(x) - \nabla f(x^\star)$. [103, Lemma 1]). *Let $f(x)$ be a strongly convex function and $x^\star \in \mathbb{R}^n$. There exists a symmetric $B(x) \in \mathbb{R}^{n,n}$ satisfying $\beta_1 I_n \preceq B(x) \preceq \beta_2 I_n$ for some $0 < \beta_1 < \beta_2$ such that*

$$\nabla f(x) - \nabla f(x^\star) = B(x)(x - x^\star). \tag{7.12}$$

Next, we present this chapter's main result.

Theorem 7.1.1. (Global exponential convergence of M-PI-CMO)

Let assumptions 3 and 4 hold. Let $\rho < \bar{c}^{-1}$. Then, there exist real positive constants α_1 and α_2 such that

$$\|x(t) - x^\star\|_2 \leq \alpha_1 e^{-\frac{1}{2}\mu t}, \quad \|\lambda(t) - \lambda^\star\|_2 \leq \alpha_2 e^{-\frac{1}{2}\mu t} \tag{7.13}$$

where

$$\mu \leq \min\left\{ \frac{1}{2} K_p \underline{c}, \frac{2K_i \underline{g} - K_p \bar{g}^2}{K_i} \right\} \tag{7.14}$$

where $0 < \underline{g} \leq \bar{g}$ are assessed in the proof.

Proof. We define the candidate Lyapunov function

$$V\big(z(t)\big) = \big(z(t) - z^\star\big)^\top P \big(z(t) - z^\star\big) \tag{7.15}$$

where

$$P := \begin{pmatrix} \sigma I_n & 0 \\ 0 & I_m \end{pmatrix} \in \mathbb{R}^{m+n,m+n} \tag{7.16}$$

for some $\sigma > 0$. If

$$\dot{V}\big(z(t)\big) \leq -\mu V\big(z(t)\big) \tag{7.17}$$

then the theorem statement holds. Therefore, let us study the conditions that guarantee (7.17).

Let us consider the diagonal matrix $\Gamma = \Gamma(z) \in [0,1]^{m,m}$ as defined in [103, Lemma 3]. Since $\nabla_x \mathscr{L}(x^\star, \lambda^\star) = \nabla_\lambda \mathscr{L}(x^\star, \lambda^\star) = 0$, $J_h(x) = C$ and by using (7.12),

$$\begin{aligned} \dot{x} &= -\nabla_x \mathscr{L}(x,\lambda) = -\nabla_x \mathscr{L}(x,\lambda) + \nabla_x \mathscr{L}(x^\star,\lambda^\star) \\ &= -B(x - x^\star) - \rho C^\top \Gamma C(x - x^\star) - C^\top \Gamma(\lambda - \lambda^\star) \end{aligned} \tag{7.18}$$

as obtained for Aug-PDGD, see [103, Section III-B] for details. Furthermore,

$$\begin{aligned}\dot{\lambda} &= \nabla_\lambda \mathscr{L}(x,\lambda) = \nabla_\lambda \mathscr{L}(x,\lambda) - \nabla_\lambda \mathscr{L}(x^\star,\lambda^\star) \\ &= K_i \Gamma C(x-x^\star) + \frac{K_i}{\rho}(\Gamma - I)(\lambda - \lambda^\star) + K_p C\dot{x}.\end{aligned} \tag{7.19}$$

Equations (7.18)-(7.19) represent (7.7) in a "linear" form.

Let $G_1 := B + \rho C^\top \Gamma C$ and $G_2 := C^\top \Gamma$. Since B is positive definite, G_1 is positive definite; let $\underline{g} I \preceq G_1^\top \preceq \overline{g} I$. Then, we can rewrite (7.18)-(7.19) in a matrix form

$$\dot{z}(t) = G\big(z(t) - z^\star\big) \tag{7.20}$$

where

$$G := \begin{pmatrix} -G_1 & -G_2 \\ K_i G_2^\top - K_p C G_1 & \frac{K_i}{\rho}(\Gamma - I) - K_p C G_2 \end{pmatrix} \tag{7.21}$$

Since

$$\begin{aligned}\dot{V}\big(z(t)\big) &= \dot{z}(t)^\top P\big(z(t) - z^\star\big) + \big(z(t) - z^\star\big)^\top P\dot{z}(t) \\ &= \big(z(t) - z^\star\big)^\top \big(G^\top P + PG\big)\big(z(t) - z^\star\big)\end{aligned} \tag{7.22}$$

a sufficient condition for $\dot{V}\big(z(t)\big) \leq -\mu V\big(z(t)\big)$, see (7.17), is

$$-G^\top P - PG - \mu P \succeq 0. \tag{7.23}$$

Consequently, our next goal is to provide sufficient conditions for (7.23). We have

$$-G^\top P - PG - \mu P = \begin{pmatrix} Q_1 & Q_2 \\ Q_2^\top & Q_3 \end{pmatrix}. \tag{7.24}$$

where

$$Q_1 = 2\sigma G_1 - \sigma\mu I_n \tag{7.25a}$$

$$Q_2 = (\sigma - K_i) G_2 + K_p G_1^\top C^\top \tag{7.25b}$$

$$Q_3 = K_p C G_2 + K_p G_2^\top C^\top + 2\frac{K_i}{\rho}(I - \Gamma) - \mu I_m \tag{7.25c}$$

If $K_i \geq K_p$, by applying [103, Lemma 6] for $\frac{1}{\rho} > \overline{c}$,

$$K_p C G_2 + K_p G_2^\top C^\top + 2\frac{K_i}{\rho}(I - \Gamma) \succeq \frac{3}{2} K_p C C^\top. \tag{7.26}$$

Thus,

$$Q_3 \succeq \frac{3}{2} K_p C C^\top - \mu I_m \succeq K_p C C^\top \tag{7.27}$$

where the last step is a consequence of the assumption $\mu \leq \frac{1}{2} K_p \underline{c}$.

To simplify the computations, we set $K_i = \sigma$. Then,

$$Q_2 = K_p G_1^\top C^\top. \tag{7.28}$$

In conclusion,

$$-G^\top P - PG - \mu P \succeq \begin{pmatrix} 2\sigma G_1 - \sigma\mu I_n & K_p G_1^\top C^\top \\ \left[K_p G_1^\top C^\top\right]^\top & K_p C C^\top \end{pmatrix}. \tag{7.29}$$

Since $CC^\top \succ 0$ is invertible from (7.11), we can apply the Schur complement argument: the matrix in (7.29) is positive semidefinite if and only if

$$2\sigma G_1 - \sigma\mu I_n - K_p G_1^\top C^\top \left(K_p C C^\top\right)^{-1} K_p C G_1 \succeq 0. \tag{7.30}$$

Since $CC^\top$ is invertible, then $C^\top (CC^\top)^{-1} C \preceq I$. Therefore, a sufficient condition for (7.30) is

$$2\sigma G_1 - \sigma\mu I_n - K_p G_1^\top G_1 \succeq 0. \tag{7.31}$$

Finally, (7.31) holds if

$$2\sigma \underline{g} - \sigma\mu I_n - K_p \overline{g}^2 \geq 0 \tag{7.32}$$

which is equivalent to

$$\mu \leq \frac{2K_i \underline{g} - K_p \overline{g}^2}{K_i}. \tag{7.33}$$

This completes the proof. □

Remark. In the proof, we set $K_i = \sigma$ to simplify the computations. Other choices may enhance the convergence rate.

Remark. A theoretical comparison of the convergence rates μ and τ_{ineq} in [103] is challenging due to the fact τ_{ineq} depends on many constants that are not easy to assess. Conversely, we can estimate μ straightforwardly, given some knowledge of the given optimization problem.

Remark. The proof of Theorem 7.1.1 is more straightforward than the proof of Theorem 2 in [103] because the diagonal form of the Lyapunov function (7.17) reduces the computations when compared to the Lyapunov function with cross terms in [103].

Theorem 7.1.1 also suggests some insights on the selection of K_i, K_p. We notice that K_p must be kept small to avoid reducing the convergence rate, while, as we shall see in the next section, K_i plays the same role as η in (6.28).

7.2 Relation with the literature

The difference between the proposed M-PI-CMO approach (7.7) and Aug-PDGD (4.39) is in the presence of the additional term $K_p J_h(x)\dot{x}$ in the dynamics of λ. Indeed, if $K_p = 0$, M-PI-CMO reduces to Aug-PDGD.

To understand the rationale of this term, we go through a feedback control interpretation according to the CMO framework proposed in Chapter 5. According to this framework, we can interpret Aug-PDGD as an algorithm with integral control on λ_j representing a non-satisfied constraint. On the other hand, in the presence of a satisfied constraint, we do not control the system through λ.

In (7.7), we modify the dynamics of λ by adding $K_p J_h(x)\dot{x}$. In the following, we show the benefits of this adjustment in terms of convergence rate.

7.2.1 Convergence rate: illustrative example to compare M-PI-CMO and Aug-PDGD

We propose a simple illustrative example to compare the convergence rates of Aug-PDGD in (4.39) and M-PI-CMO in (7.7).

We consider the scalar quadratic optimization problem

$$\begin{aligned} &\min_{x\in\mathbb{R}} \frac{1}{2} w x^2 \\ &\text{s.t.} \quad x \leq 0 \end{aligned} \tag{7.34}$$

where $w > 0$.

Since the cost function is quadratic and the constraints are linear, the closed-loop dynamics with the PI control corresponds to a switched linear time-invariant system with the following two modes:

$$\begin{pmatrix} \dot{x} \\ \dot{\lambda} \end{pmatrix} = \mathrm{A}_1 \begin{pmatrix} x \\ \lambda \end{pmatrix}, \quad \begin{pmatrix} \dot{x} \\ \dot{\lambda} \end{pmatrix} = \mathrm{A}_2 \begin{pmatrix} x \\ \lambda \end{pmatrix} \tag{7.35}$$

where

$$\begin{aligned} \mathrm{A}_1 &= \begin{pmatrix} -w-\rho & -1 \\ K_i - K_p(w+\rho) & -K_p \end{pmatrix} \\ \mathrm{A}_2 &= \begin{pmatrix} -w & 0 \\ -wK_p & -\frac{K_i}{\rho} \end{pmatrix}. \end{aligned} \tag{7.36}$$

In the first mode, the control by λ is active; in the second mode, $x(t)$ satisfies the constraints and evolves according to the derivative of the cost function.

Similarly, for Aug-PDGD, the dynamics correspond to a switched linear time-invariant system of kind (7.35) with two modes described by

$$\tilde{\mathrm{A}}_1 = \begin{pmatrix} -w-\rho & -1 \\ \eta & 0 \end{pmatrix}, \quad \tilde{\mathrm{A}}_2 = \begin{pmatrix} -w & 0 \\ 0 & -\frac{\eta}{\rho} \end{pmatrix} \tag{7.37}$$

which basically correspond to A_1 and A_2 with $K_p = 0$ and $K_i = \eta$. Now, we notice that A_2 and $\tilde{\mathrm{A}}_2$ have the same eigenvalues for $K_i = \eta$, i.e., $-w$ and $-\frac{K_i}{\rho}$. Therefore, M-PI-CMO and Aug-PDGD enjoy the same convergence rate in the second mode. Concerning the first mode, the eigenvalues of A_1 are

$$\frac{-K_p - w - \rho \pm \sqrt{(K_p + w + \rho)^2 - 4K_i}}{2}. \tag{7.38}$$

Therefore, if K_i is sufficiently large, the eigenvalues are complex conjugate, and the convergence rate is $K_p + w + \rho$.

On the other hand, the eigenvalues of $\tilde{A}_1$ are

$$\frac{-w-\rho \pm \sqrt{(w+\rho)^2 - 4\eta}}{2} \tag{7.39}$$

with the best convergence rate equal to $w+\rho$. In conclusion, in this example, M-PI-CMO has a better convergence rate than Aug-PDGD, provided that a suitable K_p is selected. In particular, for Aug-PDGD, increasing the convergence rate beyond $w+\rho$ is impossible.

We remark that although it is always possible to increase ρ both in (7.7) and (4.39), this may cause numerical issues during the integration of the differential equations.

7.3 Numerical results

In this section, we illustrate two numerical simulations to support the effectiveness of the proposed algorithm (7.7). In particular, we compare it to Aug-PDGD.

7.3.1 Quadratic programming

In this simulation, we consider a strongly convex quadratic programming (QP) in the following form:

$$\begin{aligned} x^\star = \arg\min_{x\in\mathbb{R}^n} & \frac{1}{2} x^\top \left(I + W^\top W\right) x + b^\top x \\ \text{s.t.} & \\ Cx - d \leq 0 & \end{aligned} \tag{7.40}$$

where $W \in \mathbb{R}^{n,n}, b \in \mathbb{R}^n, C \in \mathbb{R}^{m,n}, d \in \mathbb{R}^m$ are randomly generated vectors or matrices, with independent, normally distributed components. We set $n = 50$ and $m = 45$.

We solve the optimization problem using M-PI-CMO (7.7) and Aug-PDGD (4.39). We integrate the differential equations (4.39) and (7.7) in the time interval

$[0,30]$ s through the *ode45* MATLAB command to select the discretization step size optimally. We set $K_i = \eta = 1$ and $K_p = -0.7$.

Figure 7.1 shows the time evolution of $\|\max(Cx(k)-d,0)\|_2$, where k is the current iteration. This metric represents the violation of the constraints, and it is equal to zero when the state x satisfies the constraints.

Figure 7.2 shows the ℓ_2 distance from the global optimum $x^\star$, computed through the MATLAB package CVX [119].

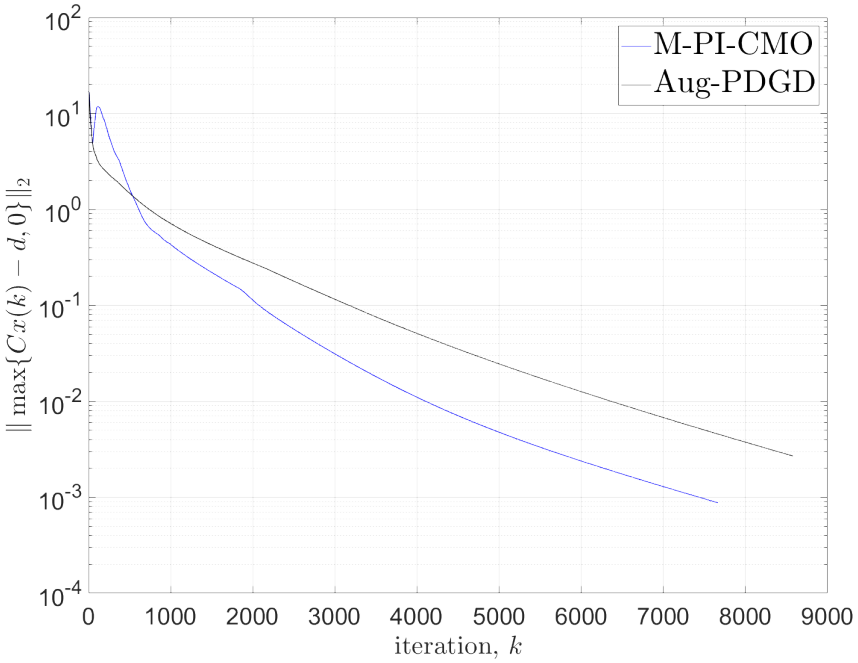

Fig. 7.1 Comparison of the constraints violation decay rate for M-PI-CMO and Aug-PDGD.

We perform 100 random runs with different realizations of W,b,C,d. M-PI-CMO requires fewer iterations than Aug-PDGD in all the runs and, consequently, less computational time. In Table 7.1, we report some statistics that show the enhanced convergence speed of M-PI-CMO compared to Aug-PDGD.

Figure 7.1 and Figure 7.2, obtained from one randomly selected run, show us that the proposed approach converges more quickly than Aug-PDGD, both concerning the constraints fulfillment and the achievement of the minimum.

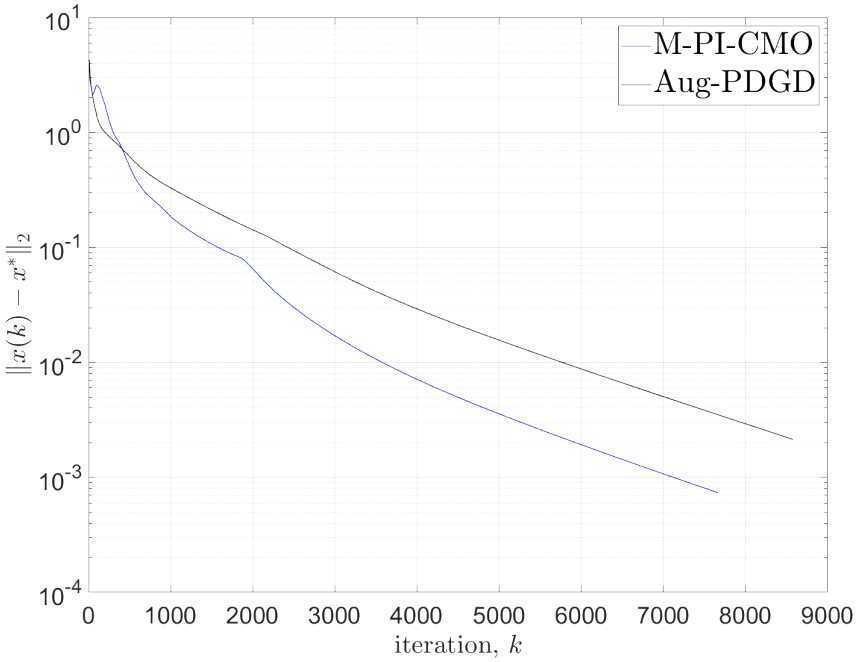

Fig. 7.2 Comparison of the distance from the global optimum decay rate for M-PI-CMO and Aug-PDGD.

	mean	standard deviation	worst case
N Aug-PDGD	8128.3	491.9	9361
N M-PI-CMO	6903.2	312.1	7613
T Aug-PDGD	1.23×10^{-1}	8.04×10^{-3}	1.53×10^{-1}
T M-PI-CMO	1.02×10^{-1}	4.74×10^{-3}	1.15×10^{-1}

Table 7.1 Comparison of the M-PI-CMO and Aug-PDGD algorithms. Statistics over 100 random runs. N is the required numbers of iteration; T the computational time (in seconds).

7.3.2 Linear system identification through linear programming

We apply our approach to a problem of system identification. We consider the problem of identifying an unknown stable linear dynamic system $H(z)$ using uncertain input-output measurements $\{u_k, \tilde{y}_k\}$, for $k \in \{1, \ldots, N\}$, where $\tilde{y}_k = y_k + e_k$, y_k is the k-th noise-free output sample and e_k is the k-th sample of the noise sequence.

To perform the identification, we select a linear-in-the-parameters model structure $\tilde{H}(z, \theta)$ of the form

$$\tilde{H}(z) = \sum_{i=1}^{P} \theta_i B_i(z). \tag{7.41}$$

This parametrization is a standard choice in the context of system identification, and commonly considered choices of $B_i(z)$ are Laguerre or Kautz filters, see, e.g., [120],[121]. For the sake of simplicity, in this example, we select $B_i(z)$ to be first-order transfer functions with poles linearly spaced in the interval $[-0.9, 0.9]$. More precisely, we select

$$B_i(z) = \frac{z}{z - p_i}, \quad p_i \in \{-0.9, -0.85, \ldots, 0.9\}. \tag{7.42}$$

To generate time-domain data, we simulate the randomly selected system

$$H(z) = \frac{-0.4z^2 + 0.32z + 0.26}{z^3 - 1.9z^2 + 1.21z - 0.259} \tag{7.43}$$

excited by a random input uniformly distributed in $[0, 1]$, and we corrupt the output data with normally distributed noise with zero mean and variance $\sigma_\eta^2 = 0.1$.

We look for the value of the parameter θ that minimizes the ℓ_∞-norm of the simulation error

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^P} \|y_k(\theta) - \tilde{y}_k\|_\infty . \tag{7.44}$$

By adding a slack variable $\Delta \in \mathbb{R}$, we recast problem (7.44) to the following linear programming problem:

$$\begin{aligned} \theta^*, \Delta^* = \arg & \min_{\theta \in \mathbb{R}^P, \Delta \in \mathbb{R}} \Delta \\ & \text{s.t.} \\ & -\Delta \leq Z_k \theta - \tilde{y}_k \leq \Delta, \quad k \in \{1, \ldots, N\} \end{aligned} \tag{7.45}$$

where $Z_k = [z_1(k), \ldots, z_P(k)] \in \mathbb{R}^P$, $z_i(k) = B_i(q^{-1})u(k)$.

We solve the optimization problem using the proposed M-PI-CMO algorithm in (7.7) and Aug-PDGD in (4.39). We integrate the differential equations (4.39) and (7.7) in the time interval $[0, 1000]$ s through the *ode23* MATLAB command to select the discretization step size optimally. We set $K_i = \eta = 1$ and $K_p = -0.5$.

In Figure 7.3, we compare the outputs of the true model and the ones estimated using the Aug-PDGD and M-PI-CMO algorithms on data not used for identification. We notice that the outputs of the three models are almost exactly overlapped. We also evaluate the validation performances of the two algorithms in terms of the best

fit rate (BFR) index, defined as

$$BFR = 100\left(1 - \sqrt{\frac{\left\|y^{val} - \hat{y}^{val}\right\|_2^2}{\left\|y^{val} - m_y^{val}\right\|_2^2}}\right) \tag{7.46}$$

where y^{val} is the true output, $m_y^{val} = \frac{1}{N}\sum_{k=1}^{N} y_k^{val}$ and $\hat{y}^{val}$ is the response of the identified model. We obtain the same BFR, i.e., BFR $= 98.5\%$, for both the identified models. Figure 7.3 and the computed BFR values show that both algorithms converge to the optimal solution as expected from the theory. We report the comparison of the two algorithms in terms of computational effort in Table 7.2, where we show the required number of iterations and computational time. Such results show that the M-PI-CMO algorithm is about two times faster than Aug-PDGD.

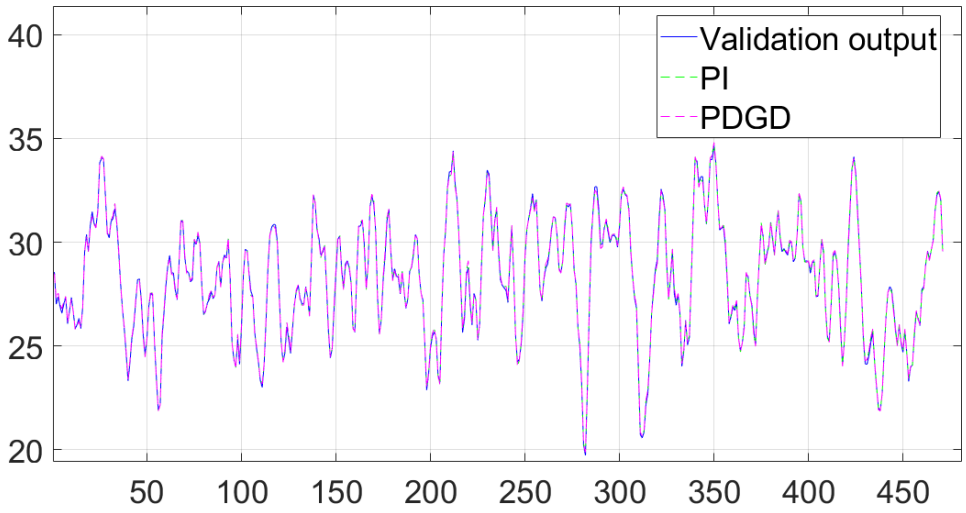

Fig. 7.3 Comparison between Aug-PDGD and M-PI-CMO on a linear system identification problem. Validation of the identified models.

	N	T
Aug-PDGD	10^6	679.46
M-PI-CMO	5×10^5	233.09

Table 7.2 Comparison between Aug-PDGD and M-PI-CMO on a linear system identification problem. N is the required numbers of iteration; T the computational time (in seconds).

Chapter 8

The FL-CMO algorithm

In this chapter, we resort to feedback linearization (FL) to design the controller $\mathscr{K}$, providing a solution to Problem 5.1.1. Moreover, we study the conditions under which the controlled dynamics are stable and the algorithms converge to the desired solution.

The key point is that $\mathscr{P}$ has the input-affine structure of (3.13), with

$$F(x) = -\nabla f(x),\ G(x) = -J_h(x)^\top,\ H(x) = h(x). \tag{8.1}$$

This structure is compatible with FL theory and, thus, renders it possible to apply the theory reported in Section 3.2 to design an FL control law for the plant $\mathscr{P}$ as defined in (5.2).

Specifically, we can apply Theorem 3.2.1 to obtain a decoupled controller if $\mathscr{P}$ has some vector relative degree. Let us analyze this point using the following assumption.

Assumption 5. *We assume that*

$$rank\,(J_h(x)) = m, \qquad \forall x \in \mathbb{R}^n \tag{8.2}$$

i.e., the Jacobian of the constraints is always full-rank.

This assumption is standard in constrained optimization; see, e.g., [41, 122, 113, 107]. In practice, it is met in many applications, including a broad class of system

identification problems, see, e.g., the example in Section 8.4.1; optimal control problems [123] and distributed optimization over networks [124].

In the case of affine constraints $h(x) = Cx + d = 0$, similarly to Assumption 2 in Chapter 6, Assumption 5 guarantees that at least one feasible solution exists and there are no linearly dependent constraints. Conversely, in the case of non-convex constraints, Assumption 5 implies that the feasible set $\Omega = \{x \in \mathbb{R}^n : h(x) = 0\}$ is an $(n-m)$-dimensional smooth manifold, i.e., locally similar to a $\mathbb{R}^{n-m}$ at all points. Consequently, Ω is not empty, and we can define a dynamical system whose trajectories evolve onto it, i.e., the zero dynamics. We finally remark that Assumption 5 implies the regularity condition of Theorem 4.1.2.

Lemma 8.0.1. (Relative degree of (5.2)).

Under assumption 5, the system $\mathscr{P}$ in (5.2) has a vector relative degree $r = (1,1,\ldots,1)^\top \in \mathbb{R}^m$.

Proof. Let us consider (3.13) with (8.1). For $r_i = 1$, it is sufficient to check (3.15) to prove the thesis. From (8.1), $H_i(x) = h_i(x)$ and $G_j(x) = -\nabla h_j(x)$. Let us set $r_i = 1$ in (3.15). Then,

$$L_{G_j} L_F^0 H_i(x) = L_{G_j} H_i(x) = (\nabla h_i(x))^\top \left(-\nabla h_j(x)\right). \tag{8.3}$$

Thus,

$$\begin{aligned} &\operatorname{rank}\left(\left[L_{G_j} L_F^0 H_i(x)\right]_{1\le i,j\le m}\right) = \\ &= \operatorname{rank}\left(\left[(\nabla h_i(x))^\top \left(-\nabla h_j(x)\right)\right]_{ij}\right) \\ &= \operatorname{rank}\left(-J_h(x) J_h(x)^\top\right) = m. \end{aligned} \tag{8.4}$$

Hence, (3.15) is satisfied for any x. □

According to Lemma 8.0.1, Theorem 3.2.1 holds for system (3.20) with (8.1). Therefore, we use the non-interacting control solution in Theorem 3.2.1. Specifically, we define the static feedback control law

$$\lambda(t) = A(x)^{-1}(-b(x) + v(t)) \tag{8.5}$$

where $A(x)$ and $b(x)$ are given by (3.22) with $r_i = 1$, i.e.,

$$\begin{aligned} A(x) &= \left[L_{G_j} H_i(x)\right]_{1\le i,j\le m} = -J_h(x) J_h(x)^\top \in \mathbb{R}^{m,m} \\ b(x) &= \left[L_F H_i(x)\right]_{i=1,\dots,m} = -J_h(x)\nabla f(x) \in \mathbb{R}^m. \end{aligned} \tag{8.6}$$

By applying the control law (8.5)-(8.6), the relationship between the new input $v(t) \in \mathbb{R}^m$ and the output $y(t) = h(x(t)) \in \mathbb{R}^m$ is

$$y_i(t) = \int_0^t v_i(\tau)\,\mathrm{d}\tau, \quad i = 1,\dots,m, \tag{8.7}$$

which is linear and decoupled in each component i.

Next, we have to design $v(t)$ such that we regulate $y(t)$ to zero. A possible solution is to design the controller $\mathscr{G}$ in Figure 8.1

$$v(t) = \mathscr{G}(y(t)) \tag{8.8}$$

such that the closed-loop dynamics is asymptotically stable and $y(t) \to 0$ as $t \to \infty$. In the remainder of this thesis, we refer to the closed-loop dynamics defined by (5.2)-(8.5)-(8.8), i.e.,

$$\dot{x}(t) = -\nabla f(x) + J_h^\top (J_h J_h^T)^{-1} (J_h \nabla f(x) + \mathscr{G}(y)), \tag{8.9}$$

as to FL-CMO dynamics.

Given the single integral structure of (8.7), the simplest way to design $\mathscr{G}$ is to consider m static linear feedback controllers

$$v_i(t) = -K_i y_i(t) \tag{8.10}$$

with $K_i > 0$ for $i = 1,\dots,m$. This leads to

$$y_i(t) = \alpha_i e^{-K_i t}, \quad i = 1,\dots,m \tag{8.11}$$

where α_i is a constant that depends on the initial conditions.

Several more possibilities are available for the $\mathscr{G}$ design, each leading to a different CT optimization algorithm in the FL-CMO family characterized by different properties. Among the infinite possible choices, we mention:

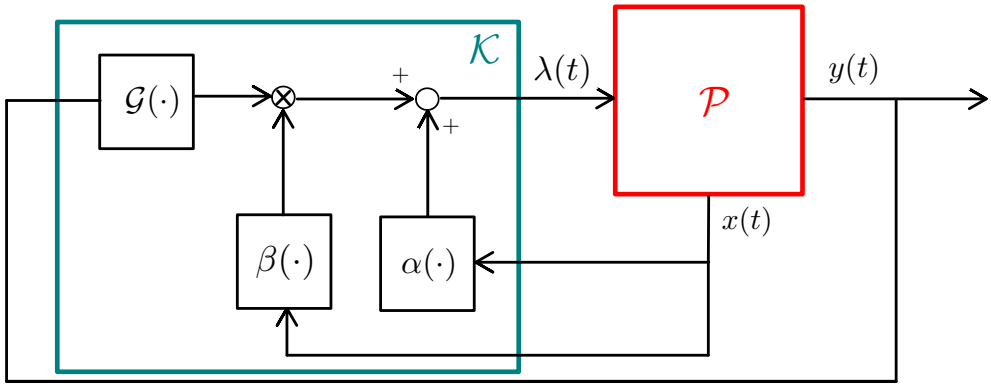

Fig. 8.1 Structure of the FL control for CMO. We feedback the output $y(t)$ to the controller $\mathscr{K}$ to compute $v(t)$ according to (8.8), and the state $x(t)$ to compute $u(t)$ according to (3.19).

- a generic LTI controller of order n_k

$$\begin{aligned} \dot{\zeta}_i(t) &= A_v \zeta(t) + B_v y_i(t) \\ v_i(t) &= C_v \zeta(t) + D_v y_i(t) \end{aligned} \tag{8.12}$$

  In this case, the closed-loop dynamics is

$$\begin{bmatrix} \dot{\zeta} \\ \dot{y}_i \end{bmatrix} = \underbrace{\begin{bmatrix} A_v & B_v \\ C_v & D_v \end{bmatrix}}_{K_v} \begin{bmatrix} \zeta \\ y_i \end{bmatrix} \tag{8.13}$$

  and stable if the matrix $K_v \in \mathbb{R}^{n_k+1, n_k+1}$ is Hurwitz, for all $i = 1, \ldots, m$.

- nonlinear controllers. A possible example in this class is $v_i(t) = K\text{sign}(y_i(t))$, with $K > 0$: in this case the overall control scheme corresponds to sliding mode control; see, e.g., [125].

Remark. FL-CMO requires the inversion of $A(x)$, which is a computational bottleneck for large m. To address this point, we notice that $-A(x) = J_h(x)J_h(x)^\top$ is positive definite by construction. Consequently, we can efficiently address the inversion through Cholesky factorization. Alternatively, we can compute the QR factorization $J_h(x)^\top = QR$. This yields the Cholesky factorization of $-A$ while avoiding the computation of the product $J_h(x)J_h(x)^\top$; in fact, $-A = R^\top Q^\top QR = R^\top R$. The $J_h(x)$ sparsity level determines which of the two methods is more effective.

From the development of FL-CMO, we can observe that its performance depends on two factors. On the one hand, the convergence rate to the feasible set Ω is entirely determined by the user when designing the external controller $\mathscr{G}$. In other terms, the convergence to Ω is arbitrarily fast. On the other hand, the zero dynamics determines the speed of convergence to the optimal solution. Since the convergence to Ω is arbitrarily fast, the overall time required for convergence is de facto determined by the uncontrollable zero dynamics. Despite this, numerical experiments demonstrate that if the convergence rate to Ω is too fast, the differential equations defining the closed-loop system become stiff, rendering the numerical integration more challenging.

To conclude, in Figure 8.1, we summarize the feedback control scheme defining FL-CMO. In the following sections, we analyze the stability of the FL-CMO dynamics, compare it with similar literature, and provide numerical examples.

8.1 Local convergence for non-convex problems

In this section, we prove that the local minima of the optimization problem 5.1 are locally asymptotically stable equilibria of the closed-loop dynamics.

First, we notice that, by construction, the feasible set Ω is globally attractive for the FL-CMO dynamics, i.e., for any initial condition, trajectories converge onto it. In the following, we prove the stability of isolated local minima for general non-convex problems.
According to Theorem 3.2.2, it is sufficient that the zero dynamics is locally asymptotically stable in the neighborhood of any local minimum of (5.1). Therefore, we focus on zero dynamics.

Let us introduce the second-order sufficient conditions; see also Chapter 4.

Definition 8.1.1. (Second-order sufficient conditions).

For problem (5.1), let $H_{xx}\mathscr{L}(x,\lambda)$ be the Hessian matrix of $\mathscr{L}(x,\lambda)$ with respect to x. We say that the second-order sufficient conditions hold at $(x^\star,\lambda^\star)$ if

$$v^\top H_{xx}\mathscr{L}(x^\star,\lambda^\star)v > 0 \tag{8.14}$$

for any $v \in \mathbb{R}^n$ such that $J_h(x^\star)v = 0$.

If the second-order sufficient conditions hold, $x^\star$ is a strict local minimum of (5.1). We prove the following result of local convergence.

Theorem 8.1.1. (Local convergence of FL-CMO dynamics).

Let the second-order sufficient conditions hold at $(x^\star, \lambda^\star)$. Then, $x^\star$ is a locally asymptotically stable equilibrium point of the FL-CMO dynamics.

Proof. Consider Theorem 3.2.2. First of all, we notice that $v(t)$ stabilizes (3.23) by construction and regulates the output to zero, see (8.8).

Therefore, it is sufficient to prove that the zero dynamics of (3.13) is asymptotically stable to obtain the asymptotic stability of the closed-loop dynamics (3.20)-(8.1)-(8.8). We prove this fact in a neighborhood of an equilibrium point $(x^\star, \lambda^\star)$ of $\mathscr{P}$ defined in (5.2).

To analyze the zero dynamics, first of all, we define the mapping $\Phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ as

$$\Phi(x) = \begin{pmatrix} h(x) \\ J_h^\perp(x^\star)(x - x^\star) \end{pmatrix}, \tag{8.15}$$

where we define $J_h^\perp(x) \in \mathbb{R}^{n-m,n}$ as follows: its rows are an orthonormal basis for the null space of the rows of J_h. As a consequence, $J_h^\perp(x) J_h^\top(x) = 0$ for all $x \in \mathbb{R}^n$. The Jacobian matrix of $\Phi(x)$ is

$$J_\Phi(x) = \begin{pmatrix} J_h(x) \\ J_h^\perp(x^\star) \end{pmatrix}. \tag{8.16}$$

Since $J_h(x)$ has rank m by Assumption 5 and $J_h^\perp(x)$ has orthogonal rows by definition, then $\operatorname{rank}(J_\Phi(x)) = n$. Therefore, Φ is invertible and provides a suitable change of coordinates in the state space of $\mathscr{P}$. In particular, we can express the transformed state $z = \Phi(x)$ as

$$z = \begin{pmatrix} \xi \\ \eta \end{pmatrix} \tag{8.17}$$

where $\xi = y \in \mathbb{R}^m$ corresponds to the output and $\eta \in \mathbb{R}^{n-m}$ represents the state of the zero dynamics of the system. We refer the reader to [43, Sec 5.1].

Now, by exploiting the change of variables via Φ, we write the normal form of the system and analyze the zero dynamics.

From (5.2), we have

$$\dot{x} = g(x,\lambda) := -\nabla f(x) - J_h(x)^\top \lambda. \tag{8.18}$$

Then, the normal form is

$$\dot{z} = \frac{\mathrm{d}}{\mathrm{d}t}\Phi(x) = J_\Phi(x) g(x,\lambda) = J_\Phi(\Phi^{-1}(z)) g(\Phi^{-1}(z),\lambda). \tag{8.19}$$

We notice that $\Phi(x^\star) = 0$. Then, let us represent (8.19) through its Taylor expansion around $(\Phi(x^\star),\lambda^\star) = (0,\lambda^\star)$, i.e.,

$$\dot{z} = \mathscr{Q} z + \mathscr{R}(\lambda - \lambda^\star) + o(z) + o(\lambda - \lambda^\star). \tag{8.20}$$

Specifically,

$$\begin{aligned} \mathscr{Q} &= \frac{\partial}{\partial z}\left[J_\Phi(\Phi^{-1}(z)) g(\Phi^{-1}(z),\lambda^\star)\right]_{|z=0} \\ &= \frac{\partial}{\partial x}\left[J_\Phi(x) g(x,\lambda^\star)\right]_{|x=x^\star} J_{\Phi^{-1}}(0) \end{aligned} \tag{8.21}$$

Since $g(x^\star,\lambda^\star) = 0$,

$$\begin{aligned} &\frac{\partial}{\partial x}\left[J_\Phi(x) g(x,\lambda^\star)\right]_{|x=x^\star} = \\ &= \frac{\partial}{\partial x}\left[J_\Phi(x)\right]_{|x=x^\star} g(x^\star,\lambda^\star) + J_\Phi(x^\star)\frac{\partial}{\partial x}\left[g(x,\lambda^\star)\right]_{|x=x^\star} \\ &= J_\Phi(x^\star)\frac{\partial}{\partial x}\left[g(x,\lambda^\star)\right]_{|x=x^\star}. \end{aligned} \tag{8.22}$$

Thus,

$$\mathscr{Q} = -J_\Phi(x^\star) H_{xx}\mathscr{L}(x^\star,\lambda^\star) J_{\Phi^{-1}}(0), \tag{8.23}$$

where

$$H_{xx}\mathscr{L}(x^\star,\lambda^\star) = -\frac{\partial}{\partial x}\left[g(x,\lambda^\star)\right]_{|x=x^\star}. \tag{8.24}$$

Moreover, by the inverse function theorem

$$J_{\Phi^{-1}}(0) = J_{\Phi}^{-1}(x^\star) = \left(J_h^\dagger(x^\star), \quad J_h^{\perp\top}(x^\star)\right) \tag{8.25}$$

where $J_h^\dagger(x^\star) = J_h^\top(x^\star)[J_h(x^\star)J_h^\top(x^\star)]^{-1}$. In conclusion,

$$\mathscr{Q} = -J_\Phi(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)\left(J_h^\dagger(x^\star), \quad J_h^{\perp\top}(x^\star)\right). \tag{8.26}$$

On the other hand, given $\frac{\partial}{\partial\lambda}g(x^\star,\lambda)_{|\lambda=\lambda^\star} = -J_h^\top(x^\star)$, we have

$$\begin{aligned}\mathscr{R} &= \frac{\partial}{\partial\lambda}\left(J_\Phi(x^\star)g(x^\star,\lambda)\right)|_{\lambda=\lambda^\star} \\ &= -J_\Phi(x^\star)J_h^\top(x^\star) = -\begin{pmatrix} J_h(x^\star)J_h^\top(x^\star) \\ J_h^\perp(x^\star)J_h^\top(x^\star)\end{pmatrix} = \\ &= -\begin{pmatrix} J_h(x^\star)J_h^\top(x^\star) \\ 0\end{pmatrix}.\end{aligned} \tag{8.27}$$

Next, we obtain the zero dynamics by setting $\xi = 0$ and considering the last $n-m$ equations of (8.19):

$$\begin{aligned}\dot\eta &= -J_h^\perp(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)\left(J_h^\dagger(x^\star), \quad J_h^{\perp\top}(x^\star)\right)\begin{pmatrix}0\\ \eta\end{pmatrix} + o(z) \\ &= -J_h^\perp(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)J_h^{\perp\top}(x^\star)\eta + o(z).\end{aligned} \tag{8.28}$$

We notice that the zero dynamics does not depend on λ.

Finally, by neglecting the high-order terms $o(z)$, the linearization of the zero dynamics is

$$\dot\eta = -J_h^\perp(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)J_h^{\perp\top}(x^\star)\eta. \tag{8.29}$$

The original nonlinear zero-dynamics is locally asymptotically stable at $\eta = 0$ if (8.29) is asymptotically stable, i.e., if the symmetric matrix

$$J_h^\perp(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)J_h^{\perp\top}(x^\star) \tag{8.30}$$

is positive definite. That holds if the second-order sufficient conditions reported in Definition 8.1.1 are satisfied. In fact, let $v = J_h^{\perp,\top}(x^\star)w$ for any non-null $w \in \mathbb{R}^{n-m}$;

since $J_h(x^\star)J_h^{\perp,\top}(x^\star) = 0$ by definition, $J_h(x^\star)v = 0$. Then,

$$v^\top H_{xx}\mathscr{L}(x^\star,\lambda^\star)v = w^\top J_h^\perp(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)J_h^{\perp,\top}(x^\star)w > 0. \tag{8.31}$$

Finally, the result follows from the solution to the non-interacting control problem (Theorem 3.2.2) and the application of Lyapunov's linearization method (Theorem 3.1.1). □

In our setting, the linearized zero dynamics in (8.29) corresponds to the zero dynamics of the linearization of $\mathscr{P}$, as we prove in the following. We notice that the commutativity of the operations of linear approximation and computation of the zero dynamics always holds for single-input single-output systems; see, e.g., [43, Remark 4.3.2]. However, commutativity is not guaranteed for general multiple-input multiple-output systems. For this reason, it is worth remarking that the class of multiple-input, multiple-output systems considered in this work, characterized by $r = (1,1,\dots,1) \in \mathbb{R}^m$ and $G = -J_H^\top$ according to (8.1), enjoys the commutativity.

By Taylor expansion, the linearization of $\mathscr{P}$ in a neighbourhood of $(x^\star,\lambda^\star)$ is

$$\begin{cases} \dot{x} = -H_{xx}\mathscr{L}(x^\star,\lambda^\star)(x-x^\star) - J_h^\top(x^\star)(\lambda-\lambda^\star) \\ y = J_h(x^\star)(x-x^\star) \end{cases} \tag{8.32}$$

Let us consider the mapping $\Phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ as

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = \Phi(x) = \begin{pmatrix} J_h(x^\star) \\ J_h^\perp(x^\star) \end{pmatrix}(x-x^\star) \tag{8.33}$$

We notice that $\xi = y$. In normal form,

$$\begin{pmatrix} \dot{\xi} \\ \dot{\eta} \end{pmatrix} = \begin{pmatrix} J_h(x^\star) \\ J_h^\perp(x^\star) \end{pmatrix}\dot{x}. \tag{8.34}$$

Let us focus on η, which represents the state variable of the zero dynamics. We have

$$\begin{aligned} \dot{\eta} &= J_h^\perp(x^\star)\dot{x} \\ &= -J_h^\perp(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)(x-x^\star). \end{aligned} \tag{8.35}$$

As expected, $\dot{\eta}$ does not depend on λ, since $J_h^{\perp}(x^\star)J_h(x^\star) = 0$. By inversion,

$$x = \begin{pmatrix} J_h^{\dagger}(x^\star), & J_h^{\perp,\top}(x^\star) \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + x^\star \tag{8.36}$$

To study the zero dynamics, we set $\xi = y = 0$. Thus,

$$x - x^\star = \begin{pmatrix} J_h^{\dagger}(x^\star), & J_h^{\perp,\top}(x^\star) \end{pmatrix} \begin{pmatrix} 0 \\ \eta \end{pmatrix} = J_h^{\perp,\top}(x^\star)\eta \tag{8.37}$$

and

$$\dot{\eta} = -J_h^{\perp}(x^\star)H_{xx}\mathscr{L}(x^\star,\lambda^\star)J_h^{\perp,\top}(x^\star)\eta \tag{8.38}$$

which is equal to (8.29).

8.2 Global exponential convergence for strongly convex problems

We study the convergence of the FL-CMO algorithm under the same assumptions we considered when dealing with PI-CMO in Chapter 6. Specifically, we consider the case when the optimization problem is strongly convex and show that, in such a case, the method enjoys global and exponential stability of the unique global minimum $x^\star$. For the self-consistency of this chapter, we state again the considered assumptions and recall some lemmas that we will use in the main result proof.

Assumption 6. *f is strongly convex and twice differentiable on $\mathbb{R}^n$.*

Assumption 7. *h is affine, i.e., $h(x) = Cx + d$, $C \in \mathbb{R}^{m,n}$, $d \in \mathbb{R}^m$. Moreover, C is full rank and there exist $0 < \alpha_1 \leq \alpha_2$ such that*

$$\alpha_1 I_m \preceq CC^\top \preceq \alpha_2 I_m. \tag{8.39}$$

The assumption that C is full rank guarantees that $Cx + d = 0$ has solutions and that there are no linearly dependent constraints.

Next, we introduce two useful lemmas. The first lemmas state that matrices' eigenvalue bounds are preserved after multiplication by orthonormal matrices.

Lemma 8.2.1. (Eigenvalues bounds preservation. [126, Observation 7.1.8]). *Let $B \in \mathbb{R}^{n\times n}, H \in \mathbb{R}^{m\times n}$ where $m < n$. Assume there exists constants $0 < \beta_1 < \beta_2$ such that*

$$\beta_1 I_n \preceq B \preceq \beta_2 I_n \tag{8.40}$$

and that the rows of H are orthonormal vectors, i.e., $HH^\top = I_m$. Then

$$\beta_1 I_m \preceq HBH^\top \preceq \beta_2 I_m. \tag{8.41}$$

Proof. Let us consider the left-hand-side inequality, i.e., $B - \beta_1 I_n \succeq 0$. Multiplying by H on the left and by $H^\top$ on the right, we get

$$H(B-\beta_1 I_n)H^\top = HBH^\top - \beta_1 HI_nH^\top = HBH^\top - \beta_1 I_m \succeq 0 \tag{8.42}$$

which proves the left-hand-side inequality in (8.41). Similarly, considering the right-hand-side $B - \beta_2 I_n \preceq 0$ we get

$$H(B-\beta_2 I_n)H^\top = HBH^\top - \beta_2 HI_nH^\top = HBH^\top - \beta_2 I_m \preceq 0, \tag{8.43}$$

which completes the proof. □

The second lemma is about the affine form of $\nabla f(x) - \nabla f(x^\star)$, as already introduced in Chapters 6 and 7.

Lemma 8.2.2. (Affine form of $\nabla f(x) - \nabla f(x^\star)$. [103, Lemma 1]). *Let $f(x)$ be a strongly convex function and $x^\star \in \mathbb{R}^n$. There exists a symmetric $B(x) \in \mathbb{R}^{n,n}$ satisfying $\beta_1 I_n \preceq B(x) \preceq \beta_2 I_n$ for some $0 < \beta_1 < \beta_2$ such that*

$$\nabla f(x) - \nabla f(x^\star) = B(x)(x - x^\star). \tag{8.44}$$

Next, we present the main result of this section.

Theorem 8.2.1 (Global exponential convergence of FL-CMO)**.** *Let assumptions 6 and 7 hold. Let us choose $\mathscr{G}$ in (8.8) such that $\bar{y} = 0$ is a globally exponentially stable equilibrium for $\dot{y} = \mathscr{G}(y)$, with convergence rate $\mu_g \geq \beta_1$, i.e., there exists*

a constant $c_g \in \mathbb{R}_+$ such that

$$\|y(t)\|_2 \leq c_g e^{-\mu_g t}. \tag{8.45}$$

Then, the global optimum $x^\star$ of problem (5.1) is a globally exponentially stable equilibrium for FL-CMO, i.e., there exists a constant $c \in \mathbb{R}_+$ such that

$$\|x - x^\star\|_2 \leq c e^{-\beta_1 t}. \tag{8.46}$$

Proof. Under assumptions 6-7, the global change of coordinates defined by (8.15) is

$$z = \Phi(x) = \begin{pmatrix} \zeta \\ \eta \end{pmatrix} = \begin{pmatrix} Cx + d \\ C^\perp (x - x^\star) \end{pmatrix} \tag{8.47}$$

where the rows of $C^\perp \in \mathbb{R}^{m-n,n}$ are an orthonormal basis for the null space of the rows of C.

Since (8.47) is an affine transformation,

$$x - x^\star = \begin{pmatrix} C \\ C^\perp \end{pmatrix}^{-1} (z - z^\star) \tag{8.48}$$

where $z^\star = \Phi(x^\star)$.

Given the spectral norm $\sigma := \left\| \begin{pmatrix} C \\ C^\perp \end{pmatrix}^{-1} \right\|_2$ and by applying the triangle inequality,

$$\|x - x^\star\|_2 \leq \sigma \|z - z^\star\|_2 \leq \sigma \left(\left\| \left(\zeta - \zeta^\star \right) \right\|_2 + \left\| \left(\eta - \eta^\star \right) \right\|_2 \right). \tag{8.49}$$

Then, we study the convergence of $\|x - x^\star\|_2$ based on the convergence of $\left\| \left(\zeta - \zeta^\star \right) \right\|_2$ and $\left\| \left(\eta - \eta^\star \right) \right\|_2$.

Since $\zeta = y$ and $\zeta^\star = 0$, the global exponential convergence of ζ is given by (8.45).

As to η, by using $\nabla_x \mathscr{L}(x^\star, \lambda(x^\star)) = 0$, the zero dynamics is

$$\begin{aligned}\dot{\eta} &= C^\perp \dot{x} = C^\perp \left[\nabla_x \mathscr{L}(x, \lambda(x)) - \nabla_x \mathscr{L}(x^\star, \lambda(x^\star))\right] = \\ &= -C^\perp \left[I + C^\top (CC^\top)^{-1} C\right)\big(\nabla f(x) - \nabla f(x^\star)\big)\Big] \\ &= -C^\perp \big(\nabla f(x) - \nabla f(x^\star)\big)\end{aligned} \tag{8.50}$$

where the last equality follows from $\mathscr{G}(0) = 0$ and $C^\perp C^\top = 0$.

According to Lemma 8.2.2, we can write

$$\dot{\eta} = C^\perp B(x)(x - x^\star) \tag{8.51}$$

In the remainder of this proof, we denote $B = B(x)$. Moreover,

$$x - x^\star = C^\top (CC^\top)^{-1}(\zeta - \zeta^\star) + C^{\perp\top}(\eta - \eta^\star) = C^{\perp\top}\eta \tag{8.52}$$

where we use that $\zeta = \zeta^\star = 0$ by definition of zero dynamics and $\eta^\star = 0$.

In conclusion,

$$\dot{\eta} = -C^\perp B C^{\perp\top}\eta. \tag{8.53}$$

From Lemma 8.2.2 we have $B - \beta_1 I \succeq 0$. Moreover, by applying Lemma 8.2.1, we have

$$C^\perp B C^{\perp\top} \succeq \beta_1 I. \tag{8.54}$$

Thus, the matrix $C^\perp B C^{\perp\top}$ is Hurwitz and there exists $c_\eta \in \mathbb{R}$ such that

$$\|\eta\| \leq c_\eta e^{-\beta_1 t}. \tag{8.55}$$

By merging (8.49), (8.45), and (8.55), we get

$$\|x - x^\star\|_2 \leq \sigma\left(c_g e^{-\mu_g t} + c_\eta e^{-\beta_1 t}\right) \leq \sigma(c_g + c_\eta) e^{-\beta_1 t} \tag{8.56}$$

which proves (8.46) with $c = \sigma(c_g + c_\eta)$. □

Remark. We can set μ_g to any desired, arbitrarily large value by using, e.g., pole placement design according to (8.10). In particular, we can always set $K = \mu_g > \beta_1$.

Remark. Theorem 8.2.1 shows that FL-CMO enjoys a better convergence rate than PI-CMO in strongly convex problems. The rate β_1 of FL-CMO is always larger than $\frac{1}{2}\mu < \beta_1$ evaluated in Theorem 6.1.1 for PI-CMO. Moreover, in the linear constraints case (see Assumption 7), a factorization of $CC^\top$ can be performed before the algorithm's iterations start. Therefore, the computational drawback of the need to solve a linear system of equations is not present in this case.

Remark. The convergence rate of FL-CMO uniquely depends on β_1, i.e., on $f(x)$, while it is independent of the constraints $h(x)$. Conversely, the convergence rate of PI-CMO and PDGD depend on constants α_1 and α_2, which, in turn, are related to the constraints.

8.3 Relation with the literature

In this section, we explore the differences and similarities of the proposed approach when compared with similar methods.

If $\mathscr{G}(\cdot)$ in (8.8) is designed to be a simple pole placement controller, i.e.,

$$\mathscr{G}(y(t)) = -Ky(t), \quad K > 0, \tag{8.57}$$

as defined in defined in (8.10), then the FL-CMO dynamics correspond to the null-space gradient dynamics studied in [108] and [107]. See Section 4.3.3 for details. In turn, this dynamics is equivalent to the safe gradient flow with CBF-based feedback control recently developed in [113] when only equality constraints are present. Interestingly, the algorithm's derivation in [113] also relies on a feedback control interpretation. They propose constructing a feedback control system where the plant is defined starting from first-order conditions, and the Lagrange multipliers are interpreted as the input. However, the algorithm derivation is completely different from that in this work. Vector control barrier functions (VCBF) are used to control the system, and the properties of the dynamics are not studied relying on control-theoretic argument; see [113] for details.

Interestingly, our framework allows several possible generalizations depending on the choice of the controller $\mathscr{G}(\cdot)$. Possible examples are listed in Section 3.2. As a matter of fact, the proposed FL-CMO extends the null-space gradient approach to a family of controllers $\mathscr{G}$ beyond (8.10). Although a comprehensive study of the

properties of different $\mathscr{G}$'s is the subject of future work, in Section 8.4.3, we propose a numerical example that compares static and dynamic $\mathscr{G}$'s in a robust optimization problem, to shed light on the benefits of dynamic controllers and draw additional considerations on this aspect. Investigating the advantages and disadvantages of the proposed extension compared to simple pole placement is a topic of future research.

8.4 Numerical examples

This section proposes three numerical examples to test the effectiveness of FL-CMO.

8.4.1 SI of a gray-box nonlinear model

In this example, we consider the identification of the discrete-time nonlinear system described by the following regressor form

$$y_t = \theta_1 e^{-(y_{t-1})^2} + \theta_2 (u_{t-1})^2 + \theta_3 u_{t-2} y_{t-1} + \theta_4 (u_{t-2})^{\theta_5} \tag{8.58}$$

using $N = 400$ measurements of the input sequence u_t and noisy measurements of the output sequence $\tilde{y}_t = y_t + \zeta_t$, where ζ_t represents the noise. Since the system is not affine in the parameters, we cannot employ standard solutions based on least-squares regression. We look for the parameter vector $\theta^\star \in \mathbb{R}^5$ that minimizes the ℓ_2 norm, i.e., the noise sequence ζ_t energy. This approach leads to the formulation of the following non-convex optimization problem:

$$\begin{aligned} [\theta^\star, y^\star] = & \underset{\theta \in \mathbb{R}^5, y \in \mathbb{R}^N}{\arg\min} \sum_{t=1}^{N} (y_t - \tilde{y}_t)^2 \\ & \text{subject to:} \\ & -y_t + \theta_1 e^{-y_{t-1}^2} + \theta_2 (u_{t-1})^2 + \theta_3 u_{t-2} y_{t-1} + \\ & + \theta_4 (u_{t-2})^{\theta_5} = 0 \qquad t = 3, \ldots, N. \end{aligned} \tag{8.59}$$

Due to the presence of non-affine equality constraints, the problem is non-convex. We solve problem (8.59) using the FL-CMO method developed in this chapter. We define the controller $\mathscr{G}$ in (8.8) according to (8.10) , with $K_i = 1$ for each $i = 1, \ldots, m$.

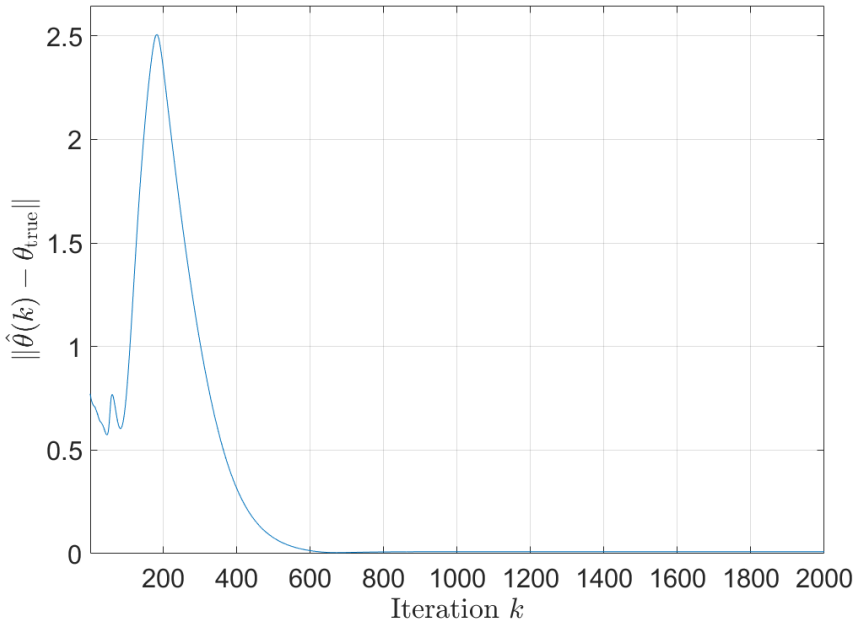

Fig. 8.2 FL-CMO in gray-box nonlinear system identification: evolution of the estimation error. We denote by $\hat{\theta}(k)$ the estimates obtained with the FL-based algorithm at iteration k, and by θ_{true} the true parameter vector.

We integrate the ordinary differential equations that describe the closed-loop dynamics in the time interval $t = [0, 20]$ seconds by using Euler discretization with step size 10^{-2} seconds. We set normally distributed initial conditions.

Figure 8.2 shows the ℓ_2-norm of the estimation error as a function of the algorithm iteration. Figure 8.3 shows the evolution of the ℓ_∞-norm of the constraints: as expected, it converges to zero.

For comparison, we solve problem (8.59) through IPM implemented with the fmincon MATLAB function and initialized with the same initial conditions used for the FL method. We use MATLAB R2021b on a processor i7-10700, 2.90GHz with 32 GB of DDR4 RAM. Table 8.1 compares the estimated parameters. We observe that both FL-CMO and IPM provide accurate estimates, the respective errors being $\|\theta_{\text{true}} - \hat{\theta}\|_2 = 0.0163$ and $\|\theta_{\text{true}} - \theta_{\text{IPM}}\|_2 = 0.0175$.

Let us analyze the computational complexity. The time required by the proposed FL approach is 9.8 seconds, while IPM requires 64.4 seconds. Moreover, the proposed FL method is less memory expensive than IPM, as expected when we compare first-order and second-order methods because it requires storing only the Jacobian of the constraints and not the Hessian matrix. More precisely, given $m \leq n$,

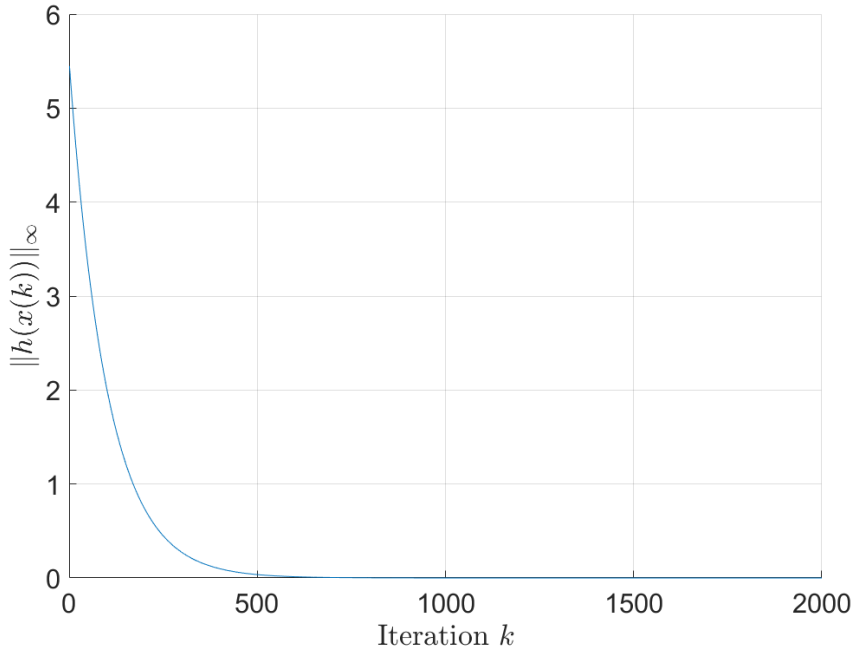

Fig. 8.3 FL-CMO in gray-box non-linear system identification: evolution of the ℓ_∞-norm of the constraints function h. In the figure, $x(k)$ refers to the estimate, at iteration k, of y and θ in (8.59).

θ_{true}	$\hat{\theta}$	θ_{IPM}
0.5	0.496	0.489
-0.3	-0.300	-0.300
-0.7	-0.699	-0.695
-0.35	-0.345	-0.338
0.8	0.815	0.804

Table 8.1 Comparison of parameter estimates for FL-CMO and IPM algorithms in gray-box nonlinear system identification, denoted by $\hat{\theta}$ and θ_{IPM}, respectively.

the FL approach requires $n + nm + \frac{1}{2}m^2 + m$ floating point numbers, while IPM requires $n + n^2 + nm + 3m$ floating point numbers. In both cases, we assess this number by assuming to store only the triangular part of the symmetric matrices. Concerning IPM, the leading term n^2 is due to the sum of $m+1$ Hessian matrices, one for $f(x)$ and m for $h_i(x)$, of dimension $n \times n$. Thus, we need $\frac{1}{2}n^2$ variables to compute the current Hessian, and a further $\frac{1}{2}n^2$ ones are used to store the partial sum.

In conclusion, there is a gain of $2m + n^2 - \frac{1}{2}m^2$ floating-point numbers by using the proposed FL method instead of IPM.

Moreover, each iteration in FL requires $O(m^3)$ floating-point operations (FLOPs) to invert the matrix $J_h J_h^\top \in \mathbb{R}^{m \times m}$, while each iteration of IPM requires $O((n+m)^3)$ FLOPs to solve a linear system of dimension $n+m$.

Similar considerations apply to sequential quadratic programming, which needs approximately the same memory and computations as IPM; see, e.g., [41, Chapter 18].

8.4.2 Industrial chemical process problem

We consider a problem that arises in the context of industrial chemical processes: the propane, isobutane, and n-butane nonsharp separation presented in [127]. It is about a three-component feed mixture required to separate products into two three-component products. This problem is included in the benchmark suite of real-world, non-convex problems proposed in [128] to test optimization algorithms. The mathematical formulation of the problem, reported in [128, Section 2.1.5], is as follows. Given $x = (x_1, \ldots, x_{48})$, minimize $f(x)$ defined as

$$\begin{aligned} f(x) = c_{11} + (c_{21} + c_{31}x_{24} + c_{41}x_{28} + c_{51}x_{33} + c_{61}x_{34})x_5 \\ + c_{12} + (c_{22} + c_{32}x_{26} + c_{42}x_{31} + c_{52}x_{38} + c_{62}x_{39})x_{13}, \end{aligned}$$

where $c_{11} = 0.23947$, $c_{12} = 0.75835$, $c_{21} = -0.0139904$, $c_{22} = -0.0661588$, $c_{31} = 0.0093514$, $c_{32} = 0.0338147$, $c_{41} = 0.0077308$, $c_{42} = 0.0373349$, $c_{51} = -5.719 \times$

10^{-4}, $c_{52} = 0.0016371$, $c_{61} = 0.0042656$, $c_{62} = 0.0288996$, subject to

$$
\begin{aligned}
h_1(x) &= x_4 + x_3 + x_2 + x_1 = 300,\\
h_2(x) &= x_6 - x_8 - x_7 = 0,\\
h_3(x) &= x_9 - x_{12} - x_{10} - x_{11} = 0,\\
h_4(x) &= x_{14} - x_{17} - x_{15} - x_{16} = 0,\\
h_5(x) &= x_{18} - x_{20} - x_{19} = 0,\\
h_6(x) &= x_6 x_{21} - x_{24} x_{25} = 0,\\
h_7(x) &= x_{14} x_{22} - x_{26} x_{27} = 0,\\
h_8(x) &= x_9 x_{23} - x_{28} x_{29} = 0\\
h_9(x) &= x_{18} x_{30} - x_{31} x_{32} = 0,\\
h_{10}(x) &= x_{25} - x_5 x_{33} = 0,\\
h_{11}(x) &= x_{29} - x_5 x_{34} = 0,\\
h_{12}(x) &= x_{35} - x_5 x_{36} = 0,\\
h_{13}(x) &= x_{37} - x_{13} x_{38} = 0,\\
h_{14}(x) &= x_{27} - x_{13} x_{39} = 0,\\
h_{15}(x) &= x_{32} - x_{13} x_{40} = 0,\\
h_{16}(x) &= x_{25} - x_6 x_{21} - x_9 x_{41} = 0,\\
h_{17}(x) &= x_{29} - x_6 x_{42} - x_9 x_{23} = 0,\\
h_{18}(x) &= x_{35} - x_6 x_{43} - x_9 x_{44} = 0,\\
h_{19}(x) &= x_{37} - x_{14} x_{45} - x_{18} x_{46} = 0,\\
h_{20}(x) &= x_{27} - x_{14} x_{22} - x_{18} x_{47} = 0,\\
h_{21}(x) &= x_{32} - x_{14} x_{48} - x_{18} x_{30} = 0,\\
h_{22}(x) &= 0.33 x_1 + x_{15} x_{45} - x_{25} = 0,\\
h_{23}(x) &= 0.33 x_1 + x_{15} x_{22} - x_{29} = 0,\\
h_{24}(x) &= 0.33 x_1 + x_{15} x_{48} - x_{35} = 0,\\
h_{25}(x) &= 0.33 x_2 + x_{10} x_{41} - x_{37} = 0,\\
h_{26}(x) &= 0.33 x_2 + x_{10} x_{23} - x_{27} = 0,\\
h_{27}(x) &= 0.33 x_2 + x_{10} x_{44} - x_{32} = 0,
\end{aligned}
$$

$$
\begin{aligned}
h_{28}(x) &= 0.33x_3 + x_7x_{21} + x_{11}x_{41} + x_{16}x_{45} + x_{19}x_{46} = 30,\\
h_{29}(x) &= 0.33x_3 + x_7x_{42} + x_{11}x_{23} + x_{16}x_{22} + x_{19}x_{47} = 50,\\
h_{30}(x) &= 0.33x_3 + x_7x_{43} + x_{11}x_{44} + x_{16}x_{48} + x_{19}x_{30} = 30,\\
h_{31}(x) &= x_{33} + x_{34} + x_{36} = 1,\\
h_{32}(x) &= x_{21} + x_{42} + x_{43} = 1,\\
h_{33}(x) &= x_{41} + x_{23} + x_{44} = 1,\\
h_{34}(x) &= x_{38} + x_{39} + x_{40} = 1,\\
h_{35}(x) &= x_{45} + x_{22} + x_{48} = 1,\\
h_{36}(x) &= x_{46} + x_{47} + x_{30} = 1,\\
h_{37}(x) &= x_{43} = 0, \qquad h_{38}(x) = x_{46} = 0.
\end{aligned}
$$

with bounds

$$
\begin{aligned}
&0 \leq x_1, \ldots, x_{20} \leq 150,\\
&0 \leq x_{25}, x_{27}, x_{32}, x_{35}, x_{37}, x_{29} \leq 30,\\
&0 \leq x_{21}, x_{22}, x_{23}, x_{30}, x_{33}, x_{34}, x_{36}, x_{38}, x_{39} \leq 1,\\
&0 \leq x_{40}, x_{42}, x_{43}, x_{44}, x_{45}, x_{46}, x_{47}, x_{48} \leq 1,\\
&0.85 \leq x_{24}, x_{26}, x_{28}, x_{31} \leq 1.
\end{aligned}
$$

The problem comprises $n = 48$ optimization variables, a bilinear cost function, and $m = 38$ linear and bilinear equality constraints. The overall problem is non-convex. Moreover, there are 47 constrained variables in bounded intervals, which give rise to 94 inequality constraints. To deal with the inequality constraints, we reformulate them by using squared-slack variables, as discussed, e.g., in [129, Section 3.3.2]. The basic idea is that we can rewrite any inequality $g(x) \leq 0$ as $g(x) + z^2 = 0$, where $z \in \mathbb{R}$ is a slack variable.

For this problem, the benchmark objective value is $f(x) = 2.1158$, as reported in [128, Table 3].

We perform the optimization through FL-CMO, implemented in MATLAB R2021b, on a processor i7-10700, 2.90GHz with 32 GB of DDR4 RAM. As in the previous example, we design the $\mathscr{G}$ controller by pole placement; see (8.8). We place the closed-loop pole at -10. We integrate the closed-loop differential equations in the time interval $[0,3]$ seconds using Euler discretization with step size $5 \cdot 10^{-5}$ seconds.

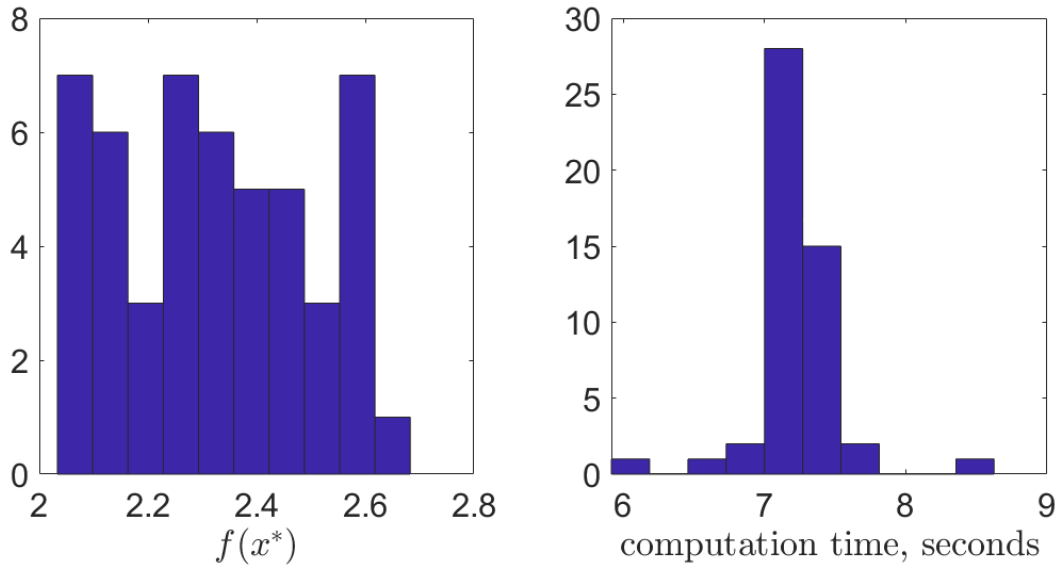

Fig. 8.4 FL-CMO results on the industrial chemical process problem: distribution of the achieved objective values (left) and distribution of the required time to converge (right)

We randomly set the initial conditions with uniform distribution in $[0,50]$. We run the algorithm 50 times with different realizations of the initial conditions. In all the runs, the algorithm converges to feasible solutions within a 10^{-7} tolerance for each constraint. The achieved objective values are distributed according to the histogram in Figure 8.4; the mean value is 2.332, and the standard deviation is 0.1852. Moreover, the best achieved objective value is 2.0333, which improves the benchmark 2.1158. In Figure 8.4, we also report the distribution of the time required to converge; the mean value is 7.22 seconds, and the standard deviation is 0.33 seconds.

We finally remark that IPM is an alternative solver for this problem, but it suffers the increase of required memory to store the Hessian matrix; see the computational complexity analysis in Section 8.4.1.

In conclusion, this experiment proves that FL-CMO is valuable in large-scale, non-convex optimization problems arising from real-world applications. Moreover, using squared-slack variables is a feasible approach to deal with inequality constraints in this example, even if it increases the number of optimization variables and introduces additional non-convex constraints.

On the other hand, when considering the FL-CMO algorithm, some numerical issues related to the condition number of the matrix $A(x)$ may arise. This condition is

due to the transformation used to transform inequality constraints to equalities, which is well-known to cause numerical problems in most optimization algorithms for equality-constrained optimization. Despite this, we remark that FL-CMO provides good solutions even when inequality constraints are present if this numerical issue does not arise, and this depends on the selected initial conditions.

8.4.3 Robustness of FL-CMO against inexact computations

As noticed in Sec. 3.2, FL-CMO requires the inversion of $A(x) \in \mathbb{R}^{m,m}$. This is a delicate point from different perspectives. On the one hand, the inverse of a poorly condition matrix is sensitive to small perturbations in the data. On the other hand, inversion is computationally intense and approximate solutions can be used to reduce the burden. In both cases, we deal with an inexact matrix $A(x)^{-1}$.
In the following example, we demonstrate how the flexibility in the design of the controller $\mathscr{G}$ for FL-CMO leads to algorithms characterized by different robustness properties in the presence of such inaccurate data.
Moreover, we provide a comparison between PI-CMO and FL-CMO equipped with a PI control law for $v(t)$.

We consider the quadratic problem

$$\min_{x\in\mathbb{R}^3} \frac{1}{2}x^\top \begin{pmatrix} 1 & 0 & 1 \\ 0 & 4 & -2 \\ 1 & -2 & 8 \end{pmatrix} x + \begin{pmatrix} -1 \\ 2 \\ -1 \end{pmatrix}^\top x \tag{8.60a}$$

$$\text{s.t.} \quad x_1 + 1 = 0, \tag{8.60b}$$

$$3x_1 + 2x_2 - 4x_3 = 0. \tag{8.60c}$$

Then $A(x)^{-1} = -\left(CC^\top\right)^{-1} = \begin{pmatrix} -1.45 & 0.15 \\ 0.15 & -0.05 \end{pmatrix}$. Based on previous considerations, we assume to know only a perturbed version $A(x)^{-1} \odot (\mathbf{1}_2 + P)$ where $\mathbf{1}_2$ is the 2×2 matrix of all ones, $\odot$ is the Hadamard product, and $P \in \mathbb{R}^{2,2}$ is such that each of its entries P_{ij} is bounded by $|P_{ij}| \leq 0.25$, for $1 \leq i, j \leq 2$.

For FL-CMO, we consider the static controller $v(t) = \mathscr{G}(y(t)) = Ky(t)$ with $K = -4$, which retraces the algorithm proposed in [107] and [113], and the dynamic

PI controller

$$v(t) = \mathscr{G}(y(t)) = k_p y(t) + k_i \int_0^t y(\tau) \mathrm{d}\tau. \tag{8.61}$$

where $k_i = -1$ and $k_p = -4$. Moreover, we implement FL-CMO with exact $A(x)^{-1}$ as a benchmark. Finally, we also consider PI-CMO with the same coefficients k_i and k_p for further comparison.

We perform 40 simulations of the four dynamics, where for each run, we start from different initial conditions and sample different P_{ij} from the uniform distribution in $[-0.25, 0.25]$. We show the simulation results in terms of the average distance from the optimum $\|x(t) - x^\star\|_2$ and the maximum constraint violation $\|h(x(t))\|_\infty$ in Figure 8.5 and Figure 8.6, respectively. As expected, the exact FL-CMO and PI-CMO converge to the correct solution. Interestingly, we can see that the pole placement controller makes FL-CMO converge quickly to a wrong solution under the effect of perturbation. On the other hand, by introducing the integral term in $\mathscr{G}$, the inexact FL-CMO converges to the correct solution, more quickly than PI-CMO.

In conclusion, this example suggests that FL-CMO with the dynamic controller is more robust to data perturbation and inaccuracies. Future work will analyze the robustness properties of FL-CMO in more general settings.

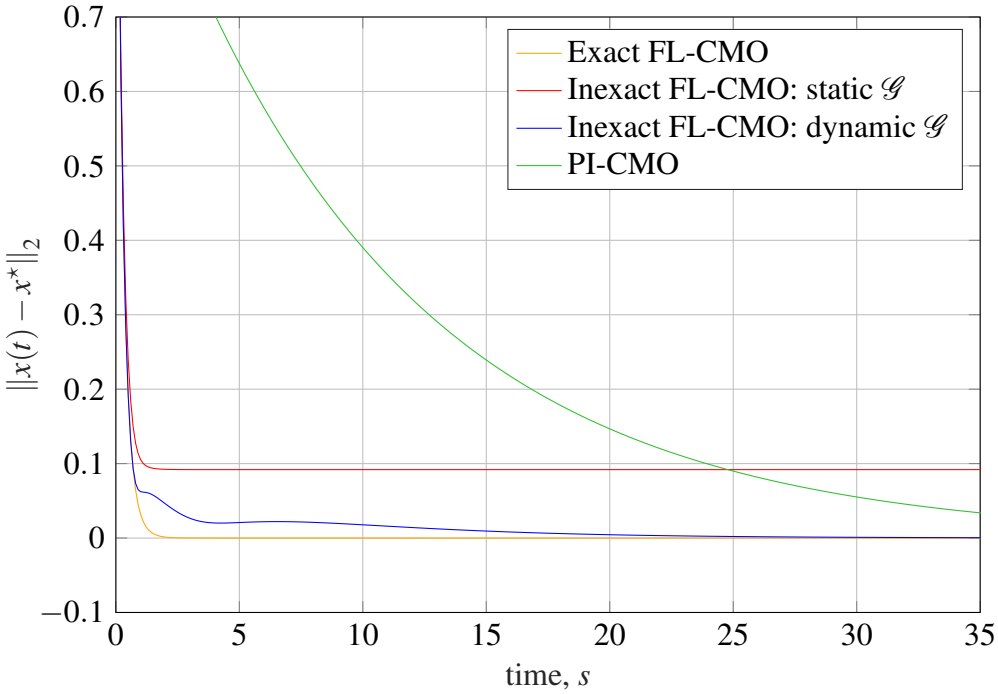

Fig. 8.5 Perturbed quadratic problem: optimization variables' trajectories $x(t)$ for the different algorithms. Exact FL-CMO refers to the unperturbed case.

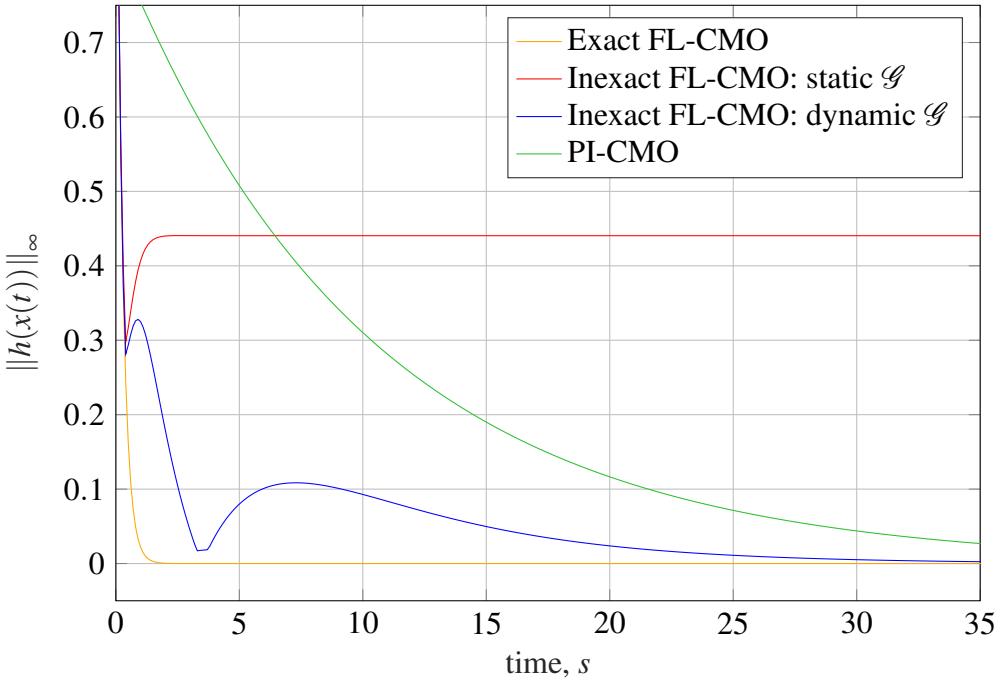

Fig. 8.6 Perturbed quadratic problem: optimization variables' constraints value $h(x(t))$ for the different algorithms. Exact FL-CMO refers to the unperturbed case.

Part III

Constrained optimization for system identification

Chapter 9

Identification of nonlinear input-output models through FL-CMO

This chapter deals with nonlinear input-output (NIO) models of the form

$$y_t = \mathscr{M}(y_{t-1}, \ldots, y_{t-n}, u_t, \ldots, u_{t-n}|\theta) \tag{9.1}$$

where $\mathscr{M} : \underbrace{\mathbb{R}^p \times \cdots \times \mathbb{R}^p}_{n\ times} \times \underbrace{\mathbb{R}^q \times \cdots \times \mathbb{R}^q}_{n+1\ times} \to \mathbb{R}^p$, $u_t \in \mathbb{R}^q$ is the input of the system, $y_t \in \mathbb{R}^p$ is the output of the system and $\theta \in \mathbb{R}^{n_\theta}$ is a vector of parameters of the model.

As discussed in Chapter 2, the considered NIO model class is rather general as, under a weak observability assumption, any nonlinear state-space model is equivalently rewritten in this form. See Section 2.1.1 for a detailed discussion. Consequently, the NIO model can account for a great variety of commonly considered models, including, for example,

- **linear time-invariant** (LTI) models

$$y_t = \sum_{j=1}^{n} \theta_j y_{t-j} + \sum_{j=0}^{n} \theta_{n+1+j} u_{t-j} \tag{9.2}$$

- **gray-box** models. The structure depends on the physics of the system.

- **nonlinear neural output error** (NNOE) neural networks. In this case, $\mathscr{M}$ is a multi-layer perceptron. In the case of a single-layer network, the NNOE network reads

$$y_t = W_2 \tanh(W_1[y_{t-1},\dots,y_{t-n},u_t,\dots,u_{t-n}]^\top + b_1) + b_2 \tag{9.3}$$

where $\theta = [vec(W_1)^\top, vec(W_2)^\top, b_1^\top, b_2^\top]^\top$ and W_1, W_2, b_1, b_2 are matrices or vectors of appropriate dimension.

9.1 Problem formulation

We consider the problem of identifying a NIO model given noisy experimental data $\{u_t, \tilde{y}_t\}$ for $t = 1,\dots,N$. Specifically, we consider the simulation error minimization problem (SEM). We formally define the SEM problem consistently with previous works on SEM, see, e.g., [15, 130]. Specifically, we consider the problem:

$$\underset{\theta\in\mathbb{R}^{n_\theta}, y_1\in\mathbb{R}^p,\dots,y_n\in\mathbb{R}^p}{\arg\min} \sum_{t=1}^{N} (\tilde{y}_t - y_t(\theta, y_1,\dots,y_n))^2 + \rho(\theta), \tag{9.4}$$

where

$$y_t(\theta, y_1,\dots,y_n) = \mathscr{M}(y_{t-1}(\theta, y_1,\dots,y_n),\dots,y_{t-n}(\theta, y_1,\dots,y_n), u_k,\dots,u_{k-n}, \theta), \tag{9.5}$$

for $t = n+1,\dots,N$, and $\rho(\theta)$ is a differentiable regularization term.

Remark: an alternative formulation of the SEM identification problem, which is frequently encountered in the machine learning literature (see, e.g., [19]), is

$$\underset{\theta\in\mathbb{R}^{n_\theta}}{\arg\min} \sum_{t=1}^{N} (\tilde{y}_t - y_t(\theta))^2 + \rho(\theta) \tag{9.6}$$

where

$$y_t(\theta) = y_t^0 \in \mathbb{R}^p, \quad \text{for } t = 1,\dots,n \tag{9.7a}$$

$$y_t(\theta) = \mathscr{M}(y_{t-1}(\theta),\dots,y_{t-n}(\theta), u_k,\dots,u_{k-n}, \theta), \quad \text{for } t = n+1,\dots,N. \tag{9.7b}$$

Common choices for selecting y_t^0 are assuming zero initial conditions, i.e., $y_t^0 = 0$, or assuming that no noise corrupts the initial samples, i.e., $y_t^0 = \tilde{y}_t$.
Problems (9.4) and (9.6) are not equivalent. The formulation (9.4) generalizes (9.6) to the case where we estimate initial conditions together with the network's parameters.

9.2 State-of-the-art

The most popular approach to solving Problems (9.4) and (9.6) is the application of gradient-based optimization methods. Among these, we mention gradient descent (GD) [131], Levenberg-Marquardt (LM) [83], and Broyden-Fletcher-Goldfarb-Shanno (BFGS) [21] algorithms. When we use a gradient-based algorithm to learn a dynamical system, the resulting algorithm is called *backpropagation through time* (BPTT), and it is often implemented by resorting to automatic differentiation tools; see, e.g., [27].

Despite the simplicity of the approach and the large availability of software tools, it is well-known that methods based on the gradient computation suffer from issues known as vanishing and exploding gradient; see, e.g., [28] and [29] for more details. In the context of machine learning, when dealing with recurrent neural networks, vanishing gradient is tackled by defining state-space neural network models known as long-short-term-memory (LSTM) and gated recurrent units (GRU), for which vanishing gradient is attenuated; see, e.g., [19]. LSTM and GRU, however, cannot be used to handle gray-box SI and are prone to overfitting in the black-box setting. Alternative approaches to tackle vanishing gradient issues include defining optimization algorithms that are not based on the computation of the loss function gradient. Examples of works along this direction include [38], where the authors propose a solution based on approximate sequential quadratic programming, and [36, 37], which apply metaheuristic optimization algorithms but are mostly avoided due to the high computational cost and the absence of theoretical guarantees.

9.3 Proposed FL-CMO-based identification algorithm

9.3.1 Constrained optimization formulation

We propose formulating the NIO identification problem (9.4) as a constrained optimization, i.e., we propose solving the problem

$$\begin{aligned} &\underset{\theta\in\mathbb{R}^{n_\theta},y_1\in\mathbb{R}^p,\ldots,y_N\in\mathbb{R}^p}{\arg\min} \sum_{t=1}^{N} (\tilde{y}_t - y_t)^2 + \rho(\theta) \\ &\text{s.t.} \\ &y_t = \mathscr{M}(y_{t-1},\ldots,y_{t-n},\tilde{u}_t,\ldots,\tilde{u}_{t-n},\theta), \\ &\quad t = n+1,\ldots,N. \end{aligned} \tag{9.8}$$

The constrained optimization formulation has previously been considered in the literature. Specifically, we mention the approach based on inexact SQP for training neural state-space models in [38] and works in the set-membership identification literature, e.g., [54] and [39].

As also noted in [38], the constrained formulation allows deriving new training algorithms and establishing convergence results. Similarly, in the following section, we will define a training algorithm based on the FL-CMO algorithm presented in Chapter 8. Moreover, we characterize the relation between the local solutions of the original unconstrained formulation (9.4) and the solutions of (9.8) through the following result.

Theorem 9.3.1. (Relation between optima of Problem (9.8) and (9.4)).

Any solution to the first-order optimality conditions of Problem (9.8) is a stationary point of Problem (9.4).

Proof. Denote $z = [\theta^\top, y_0^\top]^\top \in \mathbb{R}^{n_\theta+n}$ and $w = [y_{n+1},\ldots,y_N]^\top \in \mathbb{R}^m$. Let $h(z,w)$ denote the constraints of (9.8) and $J_h(z,w) = [J_{h,z}, J_{h,w}]$ its Jacobian. $J_{h,w} \in \mathbb{R}^{m\times m}$ is upper-triangular with all elements on the principal diagonal being non-zero; therefore, $J_{h,w}$ is invertible. Denote $G(w) = \sum_{t=n+1}^{N}(y_t - \tilde{y}_t)$ and $R(z) = \sum_{t=1}^{n}(y_t - \tilde{y}_t) + \rho(\theta)$.

The first-order optimality conditions are

$$\begin{aligned} \nabla_z R + J_{h,z}^\top \lambda = 0 \\ \nabla_w G + J_{h,w}^\top \lambda = 0 \end{aligned} \tag{9.9}$$

Then

$$\lambda = -[J_{h,w}^\top]^{-1} \nabla_w G \tag{9.10}$$

and replacing λ into the first block

$$\nabla_z R - J_{h,z}^\top J_{h,w}^{\top,-1} \nabla_w G = 0 \tag{9.11}$$

As a consequence of the Implicit Function Theorem, since $h(z,w) = 0$ and $J_{h,w}$ is invertible, there exists a function $\gamma : \mathbb{R}^{n_\theta + n} \to \mathbb{R}^m$ such that $w = \gamma(z)$ and, moreover,

$$J_{\gamma,z} = -J_{h,w}^{-1} J_{h,z} \tag{9.12}$$

Consider now (9.4): its stationary points satisfy

$$\nabla_z G(\gamma(z)) + \nabla_z R(z) = 0. \tag{9.13}$$

Using chain rule and (9.12)

$$\begin{aligned} &\nabla_z R(z) + \nabla_z G(\gamma(z)) = \\ &= \nabla_z R + \left[\nabla_{\gamma(z)} G^\top J_{\gamma,z} \right]^\top = \\ &= \nabla_z R - \left[\nabla_w G^\top J_{h,w}^{-1} J_{h,z} \right]^\top = \\ &= \nabla_z R - J_{h,z}^\top J_{h,w}^{\top,-1} \nabla_w G. \end{aligned} \tag{9.14}$$

which is zero because of Equation (9.11). □

We explain the significance of the result as follows. The constrained and unconstrained formulations have the same global optimal solutions by construction. However, local solutions are not necessarily the same. The provided theorem guarantees that we do not introduce any additional local (thus, potentially suboptimal) solution by solving for the constrained formulation in place of the unconstrained one.

9.3.2 Proposed FL-CMO algorithm

We propose addressing the constrained optimization problem (9.8) through the FL-CMO optimization algorithm for constrained optimization, which we introduced in Chapter 8.

Computational considerations motivate selecting the FL-CMO algorithm among many alternatives for constrained optimization, including sequential quadratic programming and interior-point methods (see, e.g., [41]). In fact, FL-CMO is a first-order optimization algorithm that does not require computing and storing Hessian matrices.

Consider Problem (9.8). Let us denote $x = [\theta^\top, y_1^\top, \dots, y_N^\top]^\top \in \mathbb{R}^{n_\theta + pN}$ its optimization variables, $h_t(x) = y_{t+n} - \mathscr{M}(y_{t+n-1}, \dots, y_t, u_{t+n}, \dots, u_t, \theta)$ for $t = 1, \dots, N-n$ its constraints, and $f(x) = \sum_{t=1}^{N} \|\tilde{y}_t - y_t\|_W^2 + \rho(\theta)$ its cost function.

According to the FL-CMO approach, we define the dynamical system

$$\dot{x} = -\nabla f - J_h \lambda \tag{9.15a}$$

$$\lambda = (J_h J_h^\top)^{-1}(J_h \nabla f - \mathscr{G}(h)) \tag{9.15b}$$

where $\dot{y} = \mathscr{G}(y)$ is globally asymptotically stable. In this chapter, we focus on the simplest possibility to select $\mathscr{G}$, i.e.,

$$\mathscr{G}(y) = -Ky, \quad K > 0. \tag{9.16}$$

The closed-loop system defined by (9.15)-(9.16) is then expected to converge towards a locally optimal solution to the optimization problem (9.8).

Replacing (9.16) and (9.15b) into (9.15a), we obtain a differential equation describing the closed-loop system, i.e.,

$$\dot{x} = F(x) \doteq -\nabla f - J_h (J_h J_h^\top)^{-1}(J_h \nabla f + Kh). \tag{9.17}$$

We integrate (9.17) through the adaptive Euler-Heun (AEH) method (a.k.a. Runge-Kutta 1-2) [132, 133]. Specifically, given state x_k at the generic time k, the AEH method evaluates

$$s_0 = F(x_k), \quad s_1 = F(x_k + \tau s_0) \tag{9.18}$$

and use them to compare the Euler step z_0 and the Heun step z_1 defined as

$$z_0 = x_k + \tau s_0, \quad z_1 = x_k + \tau \frac{s_0 + s_1}{2} \tag{9.19}$$

The rationale of the AEH method is that if z_0 and z_1 are similar, then the step size τ is fine enough and can be increased for the next iteration. Otherwise, it should be decreased. Consequently, we select a constant $R > 0$ acting as tolerance on the allowed relative error $\|z_0 - z_1\| / \max\{\|z_0\|, \|z_1\|\}$ and $L > 1$ used as the rate of increment/decrease of τ.

This method requires computing the right-hand-side of the differential equation (9.17) two times per iteration. This choice is the best trade-off between the requirement of keeping each iteration as simple as possible and the requirement of evaluating some measurement of the error needed to adapt the step size. Algorithm 1 depicts the resulting identification algorithm.

Numerical experiments show that this strategy considerably reduces the overall training time compared with the standard Euler method with optimally tuned step size (about $4x$ speed-up on average).

Algorithm 1 Training algorithm based on FL-CMO and Euler-Heun integration

Require: $x_0, \varepsilon_f, \varepsilon_h, K, \tau_0, R, L$
  $k \leftarrow 0$
  while $\max\{\|z_0\|, \|z_1\|\} \geq \varepsilon_f$ or $\|h(x_k)\| \geq \varepsilon_h$ **do**
    $J_0 = J_h(x_k)$
    $\tilde{f}_0 = \nabla f(x_k), h_0 = h(x_k)$
    $s_0 = -\tilde{f}_0 - J_0^\top (J_0 J_0^\top)^{-1}(J_0 \tilde{f}_0 + K h_0)$
    $J_1 = J_h(x_k + \tau_k s_0)$
    $\tilde{f}_1 = \nabla f(x_k + \tau_k s_0), h_1 = h(x_k + \tau_k s_0)$
    $s_1 = -\tilde{f}_1 - J_1^\top (J_1 J_1^\top)^{-1}(J_1 \tilde{f}_1 + K h_1)$
    Evaluate z_0, z_1 according to Equation (9.19)
    if $\|z_0 - z_1\| \leq R$ **then**
      $\tau_{k+1} = L\tau_k$
    else
      $\tau_{k+1} = \frac{1}{L}\tau_k$
    end if
    $x_{k+1} = z_1$
    $k \leftarrow k + 1$
  end while

We remark that the computation of the constraints Jacobian can be performed thanks to available automatic differentiation tools (see, e.g., [27] for a recent survey) or using tensor calculus rules (see, e.g., [134]). In both cases, we can perform this computation efficiently by exploiting parallel computation.

9.3.3 Convergence guarantee

According to [44], see also Chapter 8, the main assumption required to apply the FL-CMO method is that rank $(J_h(x)) = m$ for all x. In Problem (9.8), J satisfies

$$\frac{\partial h_i}{\partial y_k} = \begin{cases} 1 & \text{for } i = k-n \\ 0 & \text{for } i > k-n. \end{cases} \tag{9.20}$$

Therefore, it is always full-rank.

Finally, as a consequence of Theorem 8.1.1, any point (x^*, λ^*) that satisfies the second-order sufficient conditions is a locally asymptotically stable equilibrium point of the closed-loop system dynamics defined as

$$\dot{x} = -\nabla_x f - J_h^\top (J_h J_h^\top)^{-1} (J_h \nabla_x f + Kh(x)). \tag{9.21}$$

9.3.4 Exploiting the Jacobian sparsity

An analysis of the computational complexity of Algorithm 1 reveals that the most expensive computation is the linear system of equations

$$u_i \doteq (J_i J_i^\top)^{-1} r_i \tag{9.22}$$

where $r_i \doteq J_i \tilde{f}_i + Kh_i$, for $i = 1, 2$. This section discusses exploiting the peculiar sparsity pattern of J_i to effectively reduce this step's computational complexity.

To solve the linear system (9.22), we propose

1. Evaluate the Q-free QR decomposition of $J_i^\top$, obtaining the upper triangular factor $R \in \mathbb{R}^{m \times m}$ such that there exists an hortogonal Q such that $J_i^\top = QR$. Consequently, $J_i J_i^\top = R^\top Q^\top QR = R^\top R$ and $C = R^\top$ is a Cholesky factor of $JJ^\top$.

2. consecutively applying forward and backward substitution to get the linear system's solution.

Let ϕ denote the dynamical order of the NIO model. If we use a dense algorithm, the computational complexity of such a procedure is dominated by the computation of R and is given by $O(m^3)$. This observation holds independently on the selected method, including QR and Cholesky decomposition. See, e.g., [135] for a detailed study of the computational complexity of matrix decomposition algorithms.

Theorem 9.3.2. (Computational complexity of Algorithm 1)

By exploiting the sparsity of J_i, the computational complexity is reduced to $O(m^2)$.

Proof. We count the number of operations the Householder algorithm requires when multiplications by zero elements are avoided using a sparse algorithm. Table 9.1 shows the FLOPs count for each instruction of the Housholder algorithm applied to the matrix $X = J_i^\top$, both in the dense and sparse case.

Instruction	Iterations	FLOPs - dense	FLOPs - sparse
$U = 0_{n\times m}, R = 0_{m\times m}$	1	-	-
for $k = 1,\dots,m$			
$\quad U_{k:n,k}, R_{k,k} = \text{housegen}(X_{k:n,n})$	m	$3(n-k+1)$	$3\chi(k)$
$\quad v = U_{k:n,k}^\top X_{k:n,k+1:m}$	m	$(n-k+1)(m-k)$	$\psi(k)$
$\quad X_{k:n,k+1:m} = X_{k:n,k+1:m} - U_{k:n,k} v$	m	$(n-k+1)(m-k)$	$\zeta(k)$
$\quad R_{k,k+1:m} = X_{k,k+1:m}$	m	-	-
end for			

Table 9.1 FLOPs count for each instruction of Housholder algorithm

In particular, the function *housegen* performs the Householder reflection of its argument, i.e., it transforms a given vector x_k into a multiple of a unit vector while preserving length. To achieve this, first, the norm of the vector is computed, and then each element of the vector is multiplied by a scalar twice. Since x_k is a vector formed by taking from the k-th to the n-th element of $J^\top$, this requires $3(n-k+1)$ FLOPs in the dense case. When exploiting sparsity, the same operations are performed only on non-zero elements, thus leading to a complexity of $3\chi(k)$ FLOPs, where $\chi(k)$ is the cardinality of x_k. Looking at the peculiar sparsity patten of X, we compute $\chi(k)$

as

$$\chi(k) = \begin{cases} n_\theta - k + 1 + p(1+\phi) & \text{if } k \leq n_\theta \\ p(1+\phi) & \text{if } k > n_\theta \end{cases} \tag{9.23}$$

Next, to form the product $U_{k:n,k}^\top X_{k:n,k+1:m}$, $(n-k+1)(m-k)$ products are required in the dense case. When exploiting sparsity, we notice that thanks to the fact that $U_{k:n,k}$ has the same sparsity pattern of x_k except eventually for its first element, which is is set to some non-zero quantity by *housegen*. However, due to the structure of X, this element is always multiplied by zeros when forming the considered product. Then, the number of required products drops to

$$\psi(k) = \begin{cases} \psi_1(k) & \text{if } k \leq n_\theta \\ \psi_2(k) & \text{if } k > n_\theta \end{cases} \tag{9.24}$$

where

$$\begin{aligned} \psi_2(k) &= (p-i)p(\phi+1) + \phi p^2 + \cdots + 2p^2 + p^2 \\ &= (p-i)p(\phi+1) + \sum_{z=1}^{\phi} zp^2 \end{aligned} \tag{9.25}$$

$$\psi_1(k) = (n_\theta - k + 1)(m-k) + \psi_2(k) \tag{9.26}$$

where $i = \text{mod}(p,k) + 1$, consequently $k = i + p(j-1)$. Equation (9.25) follows from the fact that the number of coinciding non-zero indices decreases by p every p columns. Similarly, Equation (9.26) is obtained with the same procedure, also considering products given by the first $(n_\theta - k + 1)$ elements of $U_{k:n,k}^\top$.

We now determine $\zeta(k)$. $\zeta(k)$ is the number of FLOPs required to form the product $U_{k:n,k}v$, where $U_{k:n,k} \in \mathbb{R}^{n-k+1,1}$ is a sparse vector with cardinality

$$\chi_1(k) = \begin{cases} \chi(k) & \text{if } k \leq n_\theta \\ \chi(k) + 1 & \text{if } k > n_\theta \end{cases} \tag{9.27}$$

and $v \in \mathbb{R}^{1,m-k}$ is possibly dense. Accordingly, the result is a $n-k+1 \times m-k$ matrix where all rows corresponding to a zero element of $U_{k:n,k}$ are zero. Therefore, $(n-k+1)(m-k)$ FLOPs are required for the dense case, and the sparse case requires

$$\zeta(k) = (m-k)\chi_1(k) \text{ FLOPs.} \tag{9.28}$$

To conclude our proof, we need to show that there exist constants $M > 0$ and $m_0 > 0$ such that for all $m > m_0$:

$$|\sum_{k=1}^{m} 3\chi(k) + \psi(k) + \zeta(k)| \leq Mm^2. \tag{9.29}$$

Let us evaluate each term of the sum:

$$\begin{aligned} &\sum_{k=1}^{m} 3\chi(k) = 3\sum_{k=1}^{n_\theta} (n_\theta - k + 1 + p(1+\phi)) + \\ &+3\sum_{k=n_\theta+1}^{m} p(1+\phi) = 3m(1+\phi)p + \frac{3}{2}(n_\theta^2 + n_\theta) \end{aligned} \tag{9.30}$$

$$\begin{aligned} \sum_{k=1}^{m} \psi(k) &= \sum_{k=1}^{n_\theta} \psi_1(k) + \sum_{k=n_\theta+1}^{m} \psi_2(k) \\ &= \sum_{k=1}^{n_\theta} (n_\theta - k + 1)(m-k) + \sum_{k=1}^{m} \psi_2(k) \end{aligned} \tag{9.31}$$

Let us expand individually the two terms in (9.31).

$$\begin{aligned} &\sum_{k=1}^{n_\theta} (n_\theta - k + 1)(m-k) = \\ &= \frac{m}{2}(n_\theta^2 - n_\theta) - \frac{n_\theta}{6}(n_\theta^2 + 3n_\theta + 2) \end{aligned} \tag{9.32}$$

$$\begin{aligned} \sum_{k=1}^{m} \psi_2(k) &= \sum_{k=1}^{m} \left((p-i)(\phi+1)p + \sum_{z=1}^{\phi} p^2 z \right) = \\ &= \sum_{j=1}^{m/p} \sum_{i=1}^{p} \left((p-i)(\phi+1)p + \sum_{z=1}^{\phi} p^2 z \right) = \\ &= \frac{m}{p} \left(p^3(\phi+1) - \frac{1}{2}(\phi+1)p^2(p+1) \right) + \\ &+ \frac{1}{2} mp^2 \phi(\phi+1) = \\ &= \frac{1}{2} mp(\phi+1)(p+p\phi-1) \end{aligned} \tag{9.33}$$

$$\begin{aligned}
&\sum_{k=1}^{m}\zeta(k)=\sum_{k=1}^{n_\theta}(n_\theta-k+1+p(\phi+1))(m-k)+\\
&\quad+\sum_{k=n_\theta+1}^{m}(p(\phi+1)+1)(m-k)=\\
&=\sum_{k=1}^{n_\theta}(n_\theta-k)(m-k)+\\
&\quad+\sum_{k=1}^{m}(p(\phi+1)+1)(m-k)=\\
&=\frac{m}{2}(n_\theta^2-n_\theta)-\frac{n_\theta}{6}(n_\theta^2-1)+\frac{m^2-m}{2}(p(\phi+1)+1)\\
&=\frac{m^2}{2}(p(\phi+1)+1)+\\
&\quad+\frac{m}{2}(n_\theta^2-n_\theta-p(\phi+1)-1)-\frac{n_\theta}{6}(n_\theta^2-1).
\end{aligned}\tag{9.34}$$

From above equations, we notice that (9.34) dominates (9.30) and (9.31). Letting $M=\frac{1}{2}+\frac{1}{2}p(\phi+1)$ the result follows. □

We highlight that sparsity of J_i can also be exploited when computing the matrix-vector product $J_i\tilde{f}_i$ and $J_i^\top u_i$, requiring $m(n_\theta+n)$ FLOPs instead of $m(n_\theta+N)$ FLOPs and $n_\theta m+nm$ FLOPs instead of m^2 FLOPs, respectively.

When compared with the inexact SQP in [38], proposed for the identification of neural state-space models, but applicable similarly to any state-space model, requires computing the QR factorization of a matrix of dimension $(n+m)\times(n+m)$, where m is the number of constraints. Instead, in our approach, we need to compute the QR factorization of a matrix with dimension $m\times(n+m)$. Moreover, the iterations are exact in our approach, thus simplifying the convergence analysis.

9.4 Method's variants

9.4.1 Handling noise on the input

The considered approach based on constrained optimization allows for an immediate generalization to the case when both input and output available data are noisy, i.e., $\tilde{u}_t=u_t+\varepsilon_t$ and $\tilde{y}_t=y_t+\eta_t$ are given.

We formalize this problem as the solution to the following optimization problem

$$\begin{aligned} &\underset{\theta\in\mathbb{R}^{n_\theta},U\in R^{qN},Y\in R^{pN}}{\arg\min} \sum_{t=1}^{N}(y_t-\tilde{y}_t)^2+\gamma_u\sum_{t=1}^{N}(u_t-\tilde{u}_t)^2+\rho(\theta) \\ &\text{s.t.} \\ &y_t=\mathscr{M}(y_{t-1},\dots,y_{t-n},u_t,\dots,u_{t-n},\theta) \end{aligned} \tag{9.35}$$

where $U=[u_1^\top,\dots,u_N^\top]^\top\in\mathbb{R}^{qN}$, $Y=[y_1^\top,\dots,y_N^\top]^\top\in\mathbb{R}^{pN}$, and γ_u represents the relative amount of input noise compared to the output noise. For $\gamma_u\to\infty$, $\tilde{u}_t=u_t$ for all t, and Problem (9.8) is recovered.

In this case, the analysis of computational complexity is slightly different. Nonetheless, we notice that the number of optimization variables increases while the number of constraints remains unchanged compared to the standard case. Consequently, the dimension of the linear system $[J_iJ_i^\top]^{-1}r_i$ is the same. From numerical experiments, we notice that sparse matrix product and Cholesky factorization perform better than sparse Q-less QR factorization, at least for medium-sized datasets. A theoretical investigation of this phenomenon is the subject of future work.

9.4.2 FL-CMO for the identification of state-space models

This section discusses adapting the proposed approach to identifying nonlinear state-space models. In that case, we recast the identification problem as the equality optimization problem:

$$\begin{aligned} &\arg\underset{\theta\in\mathbb{R}^{n_\theta},X\in\mathbb{R}^{nN}}{\min} \sum_{t=1}^{N}(\mathscr{M}_2(x_t,u_t,\theta)-\tilde{y}_t)_2^2+\rho(\theta) \\ &\text{s.t.} \\ &x_{t+1}=\mathscr{M}_1(x_t,u_t,\theta),\quad t=1,\dots,N-1. \end{aligned} \tag{9.36}$$

where $X=[x_1^\top,\dots,x_N^\top]^\top\in\mathbb{R}^{nN}$ and $\mathscr{M}_1,\mathscr{M}_2$ are the model's output and state equation, respectively.

Problem (9.36) is rather general and can account for a large class of state-space identification problems, including linear, block-oriented, neural state-space, and RNNs.

We notice that in (9.36), the number of optimization variables and constraints is nN, which is higher than the NIO counterpart pN because, generally, $p < n$. Consequently, we expect the state-space identification procedure to be computationally less efficient. This consideration provides an additional motivation for selecting NIO models rather than state-space ones if the choice is free.

9.5 Numerical examples

In this section, we present several numerical examples to demonstrate the practical effectiveness of the FL-CMO-based identification algorithm presented in this chapter.

9.5.1 Fluid Damper benchmark

We consider the magneto-rheological fluid damper problem [136] used in MATLAB's System Identification (SYS-ID) Toolbox, consisting of 2000 training data and 1499 test data.

This example demonstrates the advantages of using the proposed FL-CMO training algorithm compared with the main alternatives for neural input-output modeling in system identification, i.e., backpropagation for NNARX identification and the Levenberg-Marquardt (LM) algorithm for training NNOE.

We select a single-layer network having dynamical order $n_x = 4$ and 6 neurons. We perform the training by

a) NNARX model using Adam stochastic gradient descent with batch size 32. We implement the training of this network using Python's Tensorflow Keras package [137], and we train the network for 2000 epochs.

b) NNOE model trained using *nnoe* function of NNSYD MATLAB toolbox [138] that implements the Levenberg–Marquardt algorithm until both gradient and criterion improvement falls below a tolerance set to 10^{-12}. The maximum number of iterations is 10^5.

c) The proposed CMO algorithm. We select $R = 10^{-4}, L = 1.1, K = 10$, and a maximum number of iterations equal to 300.

	training time	**BFR**
NNOE (FL-CMO)	167.34(5.109)	91.05(1.55)
NNOE (LM)	0.32(0.097)	66.66(27.14)
NNARX (Adam)	176.22(32.272)	68.14(23.32)

Table 9.2 Fluid damper benchmark: Comparison of training times and BFR for NNOE trained using FL-CMO, NNOE trained using LM, and NNARX trained using Adam.

Table 9.2 compares the training time (in seconds) and the validation best-fit rate BFR $= 100(1 - \frac{\|y-\hat{y}\|}{\|y-m_y\|})$, where y is the measured validation output, $\hat{y}$ is the simulated one, and m_y is the sample mean of y.

In this benchmark, the method requires approximately the same computational effort as NNARX (Adam) but performs better. On the other hand, using the LM algorithm to train the same NNOE network requires much less computational effort, but the algorithm is more likely to trap in bad local minima.

The proposed FL-CMO approach achieves state-of-the-art performances. For convenience, we report the main results in Table I in [139]: standard BPTT training (Adam) using RNN and LSTM provide on this dataset validation BFR of approximately 85.5% (for both models). At the same time, the EKF strategy in [139] yields a BFR of 89.78% and 88.97% for RNN and LSTM, respectively.

9.5.2 Bouc-Wen benchmark

The Bouc-Wen model has been intensively exploited to represent hysteretic effects. This problem is part of the nonlinear system identification benchmarks proposed in the yearly Nonlinear System Identification Benchmarks workshop. See [140] for a detailed description of the problem.

For the identification, we generate 2000 in data for training and 5000 for validation using a multi-sine input signal using the provided MATLAB function and default values. We corrupt the output training data by an additive white Gaussian noise with standard deviation 8×10^{-3} mm. For the test, we use the data provided by the files "uval_multisine.mat" and "yval_multisine.mat" containing 8192 data pairs.

We select as model structure an NNOE network with dynamical order $n = 3$ and two layers composed of 8 neurons each. Overall, the network is described by

145 parameters. We perform the training using Algorithm 1 setting $K = 10$ and a maximum number of iterations to 3500. We use as regularization $\rho(\theta) = 10^{-3} \|\theta\|_2^2$.

The identified model achieves a test RMSE $= 3.912 \times 10^{-5}$ mm, corresponding to BFR $= 94.13\%$. Achieved performances are comparable with the state-of-the-art. In particular, the polynomial method in [141], which is tailored for systems with hysteresis, achieves test RMSE $= 1.87 \times 10^{-5}$ mm.

This model achieves superior performance compared to available results using other general-purpose methods. In particular, we mention the DynoNet model [142] that achieves a test RMSE $= 4.52 \times 10^{-5}$ mm and the nonlinear finite impulse response model [143] that achieves test RMSE $= 16.3 \times 10^{-5}$ mm.

We achieve these performances using as few as $N = 2000$ training data. This demonstrates the NNOE model's data efficiency compared with other neural models.

9.5.3 MIMO polynomial Wiener-Hammerstein system

We consider a Wiener-Hammerstein system with two inputs and two outputs described by Equation (9.37).

$$y_1(t) = 7.5 z_4(t) z_1(t) - 51 z_2(t) + \\ + 1.02 z_4^2(t) z_2(t) - 9 z_1^3(t) \tag{9.37a}$$

$$y_2(t) = 2.7 z_2(t) z_3(t) - 0.729 z_4(t) z_2^2(t) \tag{9.37b}$$

$$z_1(s) = \frac{16}{s^2 + 40s + 130} \zeta(s) \tag{9.37c}$$

$$z_2(s) = \frac{80}{s^3 + 31s^2 + 340s + 2730} (u_1(s) + u_2(s)) \tag{9.37d}$$

$$z_3(s) = \frac{30}{s + 10} (2u_2(s) - 1.3\zeta(s)) \tag{9.37e}$$

$$z_4(s) = \frac{-10}{s^2 + 6s + 80} u_1(s) \tag{9.37f}$$

$$\zeta(t) = 0.2 u_1(t) u_2(t) - u_1(t) - u_2(t) + \\ - 0.04 u_2^2(t)(u_2(t) + u_1(t)) \tag{9.37g}$$

We generate a training dataset by simulating system (9.37) for 35 seconds and sampling the obtained signals with period 0.01 seconds, thus obtaining $N = 3500$ training samples. No noise corrupts the generated data. We solve the identification problem by using an NNOE network with dynamical order 3, which is assumed to be the only a-priori information about the system to be identified. Several neural network structures are selected. We select $K = 100$ to perform the training. The algorithm stops when $\|F(\theta, y)\| < 10^{-3}$, or when the number of iterations exceeds 500.

To assess the quality of the obtained model, we evaluate the RMSE simulation error on a test dataset composed of 5000 samples and the BFR as defined in subsection 9.5.1.

We also perform training on LSTM, GRU, and NNARX networks with different structures for comparison. Tables 9.3 and 9.4 show the obtained results. LSTM and GRU take more time and provide less accurate results. As expected, NNARX shows great one-step-ahead prediction performances but very bad simulation performances despite the absence of measurement noise.

Figures 9.1 and 9.2 show the validation performances of the obtained model.

9.5.4 Gray-box identification of magnetic levitation system

We consider the magnetic levitation system described by the CT Equation (9.38).

$$\ddot{z} = g - \frac{k_m i^2 + k_0}{m z^2} \tag{9.38}$$

where z is the distance between the magnet and the ball, m is the mass of the ball, g is gravity acceleration, and k_m and k_0 are constants depending on the magnet characteristics. The objective of the identification procedure is to estimate k_m and k_0, while $m = 24.197\,\mathrm{g}$ and $g = 9.81\,\mathrm{m/s^2}$ are assumed to be exactly known.

We perform a simulation of the differential equation (9.38) and collect $N = 200$ experimental input-output data using a sampling rate $T_s = 1 \times 10^{-3}\,\mathrm{s}$. We then corrupt the output data by an additive measurement noise uniformly distributed in the interval $[-5\,\mathrm{mm}, 5\,\mathrm{mm}]$.

Table 9.3 MIMO polynomial Wiener-Hammerstein system example: Comparison of several networks.

NNOE network				NNOE Training: CMO algorithm (CPU)				NNOE Validation		
order	layers	neuron/layer	# parameters	training RMSE	training RMSE	iterations	time	time/iter	validation RMSE	validation RMSE	validation FIT	validation FIT
3	1	5	87	0.4667	0.3141	320	514.19	1.607	0.2816	0.2670	86.03%	83.00%
3	1	8	138	0.4411	0.2979	285	633.59	2.223	0.4557	0.3528	77.39%	77.54%
3	1	10	172	0.3535	0.0295	500	817.81	1.636	0.1133	0.0864	94.38%	94.50%
3	1	15	257	0.3532	0.0252	420	703.28	1.674	0.1145	0.0925	94.32%	94.11%
3	2	3	65	0.3880	0.1733	300	474.93	1.583	0.1622	0.2078	91.95%	86.77%
3	2	5	117	0.3550	0.0728	360	585.42	1.626	0.0994	0.1071	95.07%	93.18%
3	2	7	177	0.4979	0.2764	480	777.75	1.620	0.2942	0.1718	85.41%	89.06%
3	2	10	282	0.3541	0.0255	500	826.19	1.652	0.1188	0.1097	94.11%	93.02%
3	3	3	77	0.4049	0.1898	260	453.32	1.744	0.1494	0.1958	92.59%	87.54%
3	3	5	147	0.4247	0.2683	180	438.12	2.434	0.2694	0.2975	86.64%	81.06%
3	3	7	233	0.6969	0.7338	140	244.47	1.746	0.2452	0.2378	87.84%	84.86%

LSTM network				LSTM Training: BPTT algorithm (GPU)					LSTM Validation		
order	layers	units/layer	# parameters	training RMSE	training RMSE	epochs	time	time/epoch	validation RMSE	validation FIT	validation FIT
4	1	2	46	0.5475	0.4164	12000	841.05	0.0701	0.6236	0.7680	69.06%	51.11%
6	1	3	80	0.4142	0.3207	12000	821.17	0.0684	0.5876	0.6221	70.85%	60.40%
8	1	4	122	0.2371	0.1841	12000	1036.00	0.0863	0.4236	0.2769	78.99%	82.37%
10	1	5	172	0.0793	0.0790	12000	827.54	0.0690	0.3811	0.2150	81.09%	86.31%
14	1	7	296	0.0545	0.0505	12000	826.17	0.0688	0.3606	0.2628	82.11%	83.27%
8	2	2	86	0.2747	0.2142	12000	1690.19	0.1408	0.3254	0.2512	83.86%	84.00%
12	2	3	164	0.2131	0.1636	12000	3061.88	0.2552	0.3858	0.4607	80.86%	70.67%
16	2	4	266	0.1781	0.1543	12000	2879.37	0.2399	0.8467	0.4391	58.00%	72.04%
20	2	5	392	0.1867	0.1390	12000	2392.98	0.1994	0.8459	0.7204	58.04%	54.14%
12	3	2	126	0.2046	0.2049	12000	2865.79	0.2388	1.7057	0.3697	15.38%	76.46%
18	3	3	248	0.0755	0.0818	12000	2309.90	0.1925	0.3655	0.1999	81.87%	87.27%

Table 9.4 MIMO polynomial Wiener-Hammerstein system example: Comparison of several networks.

GRU network				GRU Training: BPTT algorithm (GPU)					GRU Validation			
order	**layers**	**units/layer**	**# parameters**	**training RMSE**	**training RMSE**	**epochs**	**time**	**time/epoch**	**validation RMSE**	**validation RMSE**	**validation FIT**	**validation FIT**
3	1	3	71	0.4960	0.3020	12000	1498.09	0.1248	1.1727	1.3030	41.82%	17.05%
4	1	4	106	0.1376	0.1128	12000	1297.46	0.1081	0.3307	0.2031	83.59%	87.07%
5	1	5	147	0.1617	0.1380	12000	819.22	0.0683	0.1929	0.2127	90.43%	86.46%
7	1	7	247	0.0564	0.0504	12000	810.15	0.0675	0.2219	0.2425	88.99%	84.56%
10	1	10	442	0.0741	0.0853	12000	1460.55	0.1217	0.4643	0.7117	76.96%	54.69%
6	2	3	143	0.2337	0.1584	12000	1578.26	0.1315	0.3226	0.3752	84.00%	76.11%
8	2	4	226	0.1353	0.1265	12000	1581.26	0.1318	0.3751	0.5469	81.39%	65.18%
10	2	5	327	0.0326	0.0309	12000	1591.46	0.1326	0.6165	0.2345	69.41%	85.07%
14	2	7	583	0.0253	0.0325	12000	1571.18	0.1309	0.3244	0.3920	83.91%	75.05%
6	3	2	114	0.1929	0.1647	12000	3638.46	0.3032	0.3943	0.2501	80.44%	84.08%
9	3	3	215	0.0829	0.0858	12000	3741.53	0.3118	0.3664	0.2032	81.83%	87.07%

NNARX network				NNARX Training: BP algorithm (GPU)					NNARX Validation			
order	**layers**	**neuron/layer**	**# parameters**	**training RMSE**	**training RMSE**	**epochs**	**time**	**time/iter**	**validation RMSE**	**validation RMSE**	**validation FIT**	**validation FIT**
3	1	5	87	5.0156	4.0211	1000	89.59	0.090	6.3442	3.6032	-214.72%	-129.39%
3	1	8	138	6.8757	6.9953	1000	93.47	0.093	7.2585	6.9879	-260.08%	-344.87%
3	1	10	172	7.8516	7.7647	1000	104.78	0.105	7.7406	9.3001	-284.00%	-492.07%
3	1	15	257	8.0630	9.8919	1000	112.59	0.113	7.4535	10.7121	-269.76%	-581.97%
3	2	3	65	8.2622	9.5509	1000	170.33	0.170	8.1210	9.6828	-302.87%	-516.44%
3	2	5	117	7.7178	9.4936	1000	120.02	0.120	6.6021	9.3484	-227.52%	-495.15%
3	2	7	177	9.9773	8.4243	1000	117.51	0.118	9.8954	8.5393	-390.90%	-443.64%
3	2	10	282	2.5683	5.5859	1000	105.30	0.105	2.0974	5.8676	-4.05%	-273.55%
3	3	3	77	5.4776	3.1718	1000	120.97	0.121	5.5254	2.8112	-174.11%	-78.97%
3	3	5	147	5.3081	5.7561	1000	112.76	0.113	5.2680	6.5923	-161.34%	-319.69%
3	3	7	233	7.6568	9.9372	1000	120.09	0.120	7.5069	11.4208	-272.40%	-627.09%

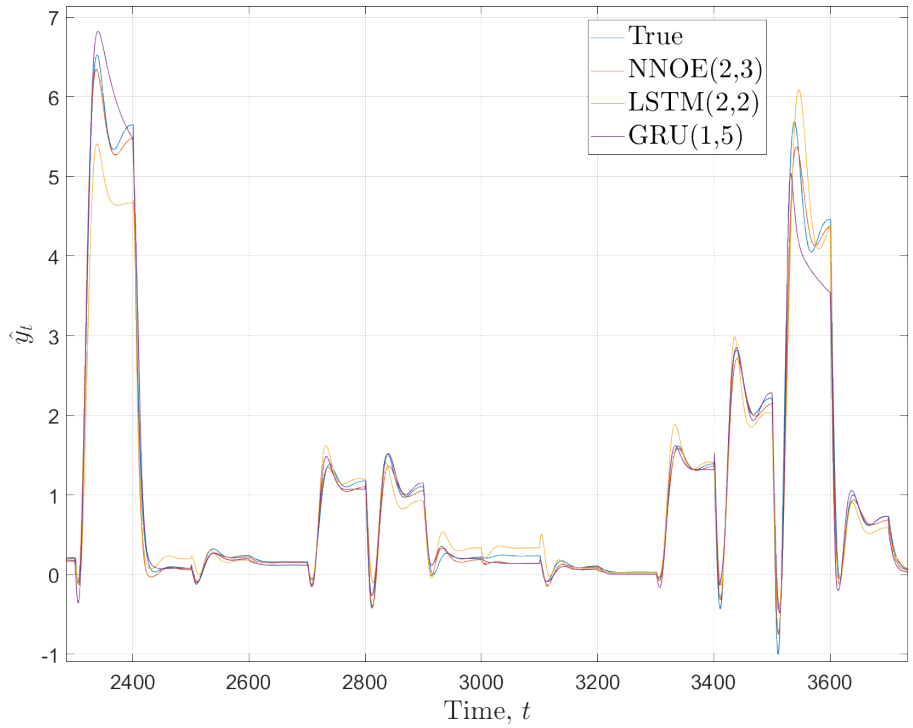

Fig. 9.1 MIMO polynomial Wiener-Hammerstein system example: Comparison of the best models in each class. First output.

We proceed by discretizing the dynamic system equation by Euler's method, which gives us the model

$$m z_{t-2}^2 \left(\frac{z_t - 2z_{t-1} + z_{t-2}}{T_s^2} - g \right) + k_m i_{t-2}^2 + k_0 = 0, \tag{9.39}$$

where T_s is the selected sampling rate.

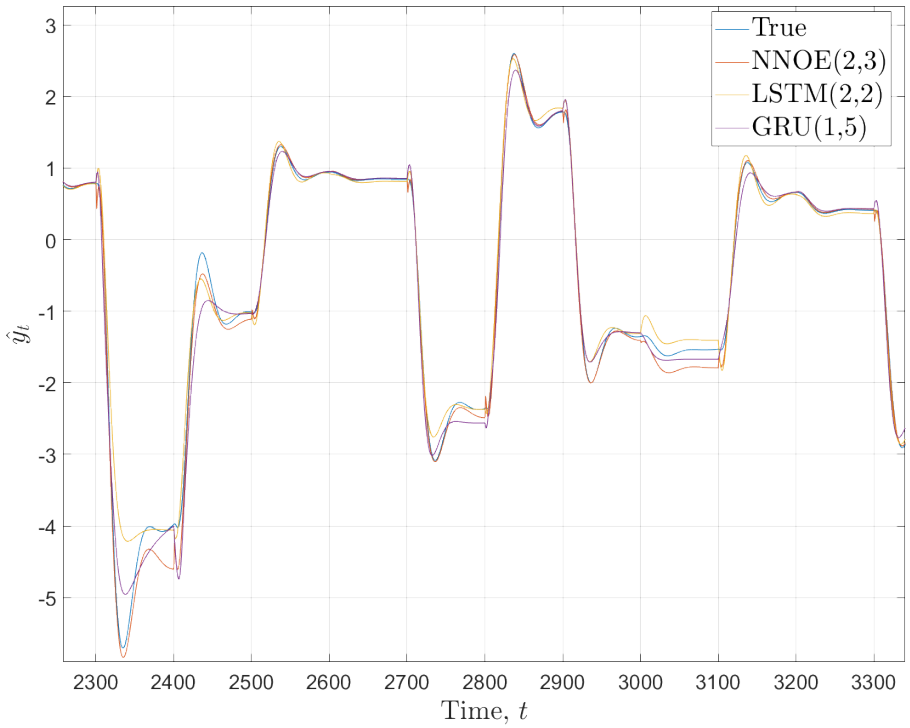

Fig. 9.2 MIMO polynomial Wiener-Hammerstein system example: Comparison of the best models in each class. Second output.

According to the proposed approach, we must evaluate the Jacobian of the constraints (9.39). In particular, we need the partial derivatives:

$$\frac{\partial h_t}{\partial k_m} = i_{t-2}^2, \tag{9.40a}$$

$$\frac{\partial h_t}{\partial k_0} = 1, \tag{9.40b}$$

$$\frac{\partial h_t}{\partial z_t} = \frac{m}{T_s^2} z_{t-2}^2, \tag{9.40c}$$

$$\frac{\partial h_t}{\partial z_{t-1}} = -\frac{2m}{T_s^2} z_{t-2}^2, \tag{9.40d}$$

$$\frac{\partial h_t}{\partial z_{t-2}} = \frac{m}{T_s^2} z_{t-2}(2z_t - 4z_{t-1} + 3z_{t-2}) - 2mgz_{t-2}. \tag{9.40e}$$

We apply the proposed FL-CMO-based identification algorithm using $K = 1$. For comparison, we also identify the system by:

	k_m	k_0
True	$2.1039 \times 10^{-4}\,\mathrm{H\,m}$	$0.0\,\mathrm{N\,m}$
Estimated (FL-CMO)	$2.1037 \times 10^{-4}\,\mathrm{H\,m}$	$2.4544 \times 10^{-5}\,\mathrm{N\,m}$
Estimated (LS)	$2.1228 \times 10^{-4}\,\mathrm{H\,m}$	$-3.6130 \times 10^{-4}\,\mathrm{N\,m}$
Estimated (Adam BPTT)	$7.46 \times 10^{-4}\,\mathrm{H\,m}$	$-0.143\,\mathrm{N\,m}$

Table 9.5 Gray-box identification of magnetic levitation system: Comparison of estimated parameters estimated using FL-CMO, LS, and Adam.

- least-squares (LS). We minimize the one-step-ahead prediction error, which leads to a linear and convex LS problem. The LS approach is the standard method to estimate the physical parameters of dynamical systems when, as in this case, the system equation depends linearly on the unknown parameters.
- Adam BPTT. We formulate the unconstrained optimization problem of simulation error minimization (see Equation 9.6), and we solve it by using Adam gradient descent (i.e., we use BPTT) and evaluating the loss' gradient by automatic differentiation. We select a learning rate of 10^{-3} and optimize for 10^4 epochs.

We repeat the optimization 10 times for each method using different, normally distributed initialization for the optimization variables. In Table 9.5, we compare the results of all three methods by selecting the best result for each method. We notice that FL-CMO achieves the best identification accuracy. LS provides an acceptable estimate despite the estimated k_0 being one order of magnitude less accurate than the FL-CMO counterpart. Instead, the Adam BPTT method provides a very inaccurate estimation for both parameters.

Chapter 10

Polynomial optimization for linear SI

In this chapter, we consider generic discrete-time (DT) linear-time-invariant (LTI) systems:

$$G(q^{-1}) = \frac{\sum_{i=0}^{n} \beta_i q^{-i}}{1 + \sum_{i=1}^{n} \alpha_i q^{-i}}, \tag{10.1}$$

where $\theta = [\alpha^\top, \beta^\top]^\top \in \mathbb{R}^p$ are the parameters of the model and q^{-1} is the backward time-shift operator. The problem of minimizing the ℓ_2-norm of the simulation error (or output error) is

$$\underset{\theta \in \mathbb{R}^p}{\arg\min} \ \|\tilde{y}_t - \hat{y}_t(\theta)\|_2^2 \tag{10.2}$$

where $\hat{y}_t(\theta)$ is the output of the model when fed with the input u_t, and $\tilde{y}_t$ is the experiment measured output.

Following the same approach proposed in Chapter 9, Problem (10.2), we reformulate the identification problem as constrained optimization. Specifically, we propose solving the problem

$$\begin{aligned} &\underset{\theta \in \mathbb{R}^p, y \in \mathbb{R}^N}{\arg\min} \ \|\tilde{y}_t - y_t\|_2^2 \\ &\text{s.t.} \quad \mathrm{y_t} = -\sum_{\mathrm{j}=1}^{\mathrm{n}} \alpha_{\mathrm{j}} \mathrm{y_{t-j}} + \sum_{\mathrm{j}=0}^{\mathrm{n}} \beta_{\mathrm{j}} \mathrm{u_{t-j}}, \quad \mathrm{t} = \mathrm{n}+1, \ldots, \mathrm{N}. \end{aligned} \tag{10.3}$$

We notice that both Problem (10.2) and Problem (10.3) are polynomial optimization problems (POPs). The main difference between the two lies in that (10.2) is unconstrained, but the loss function is a polynomial of very high degree. In contrast,

(10.3) is characterized by loss function and equality constraints with low degree. In particular, the objective function is convex and quadratic, while equality constraints involve products between optimization variables and, therefore, are non-convex and bilinear. Moreover, as shown in [144], enforcing the stability of the identified model amounts to introducing additional polynomial inequality constraints into the optimization problem.

Beyond that, problems formulated as POP include: linear set-membership estimation and system identification [54, 16], data-driven control [10], hybrid system identification [145], model predictive control [146, 147], and optimal control [148]. More recent applications include localization in narrowband internet-of-things [149], routing games [150], and estimation of Lipschitz constants in deep neural networks [151].

The above-mentioned problems motivate why, in the remainder of this chapter, we devote our attention to a generic POP, i.e., a problem of the kind

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ & \text{s.t.} \\ & g_i(x) \leq 0 \quad i=1,\dots,l \\ & h_i(x) = 0 \quad i=1,\dots,m, \end{aligned} \tag{10.4}$$

where f, g_i, h_i are multivariate polynomials of the decision variable $x \in \mathbb{R}^n$.

10.1 State-of-the-art and contribution

POPs are non-convex and NP-hard problems; therefore, searching for their global minima is challenging. To this purpose, solutions based on semidefinite programming (SDP) relaxations were proposed in the literature. In a nutshell, one can build a hierarchy of SDP problems that converge to the global optimum under mild assumptions; see [152–154] for a complete overview. The SDP approach recasts POPs into convex optimization, which makes the problem tractable. The development of the related theory is mature, and several software implementations are available, such as SparsePOP [155] and TSSOS [156].

A drawback of the SDP approach is that the size of the involved matrices at d-degree of the hierarchy is proportional to $\binom{n+d}{n}$, where n is the number of variables of the original POP. This approach becomes computationally prohibitive for medium and large-scale models, especially when high d are necessary to achieve the optimal solution. Recent literature proposes SDP-accelerating strategies to tackle this problem. These strategies exploit the presence of specific sparsity structures, such as the running intersection property [157], chordal sparsity [158, 159], or term sparsity [156]. However, these sparsity structures are not always present in SDP relaxations of POPs, as discussed, e.g., in [158]; therefore, these fast implementations are feasible only for some classes of POPs.

Given the shortcomings in global optimization with SDP, one can relax POPs to search for local minima or stationary points. In the literature, substantial attention is devoted to the local minimization of non-convex problems, e.g., via gradient-based methods, due, e.g., to the rise of deep learning techniques; we refer the reader to [160] for a recent survey. Convergence of the algorithms is a delicate point in this framework, both in terms of guarantees and speed.

In this work, we propose a novel approach to local minimization of POPs based on the alternating direction method of multipliers (ADMM, [161]). ADMM is well-known in convex optimization as an effective algorithm to minimize composite functionals, even in non-differentiable terms. ADMM structure is easy to implement and prone to decentralization. Its convergence is proven and analyzed in, e.g., [161, 162]. As to non-convex optimization, the convergence of ADMM is more critical to analyze and is proven only for some classes of problems; for a statement of the main results, we refer the reader to [163, 164].

The application of ADMM to POPs is not straightforward since POPs are non-convex, constrained problems, while ADMM was initially conceived for convex, unconstrained problems. The first contribution of this work is a suitable reformulation of POPs. On top of that, we developed a novel ADMM-based procedure for POPs, which we call ADMM4POP. The second contribution is proof of the convergence of ADMM4POP in the case of equality constraints, which we account for in a relaxed way. Furthermore, we illustrate the implementation of ADMM4POP in some numerical examples and compare its practical effectiveness to state-of-the-art global/local minimization methods.

10.2 ADMM for polynomial optimization

We start by rewriting (10.4) as an unconstrained, composite problem with some specific properties. In this section, we illustrate the proposed reformulation.

The first step is to write (10.4) as a quadratic optimization problem (QOP). As discussed, e.g., in [165, 166], any POP can be transformed into a QOP by adding suitable slack variables. Given a POP, different equivalent QOP representations are possible, as illustrated in [165, Section 2.3].

In the remainder of this chapter, given two vectors v, w of equal dimension, we write $v \leq w$ to denote the componentwise inequality. Moreover, we use the notation $A \succeq 0$ to indicate that a matrix A is positive semidefinite.

For our purpose, we consider any transformation that provides a QOP of this form:

$$\begin{aligned} \min_{x \in \mathbb{R}^{\tilde{n}}} \quad & \frac{1}{2} x^\top A x + a^\top x \\ & \text{s.t.} \\ & Bx - b \leq 0 \\ & Cx - c = 0 \\ & x_i x_j = x_k, \quad (i,j,k) \in \mathscr{B} \end{aligned} \tag{10.5}$$

where $x \in \mathbb{R}^{\tilde{n}}$, $\tilde{n} \geq n$, is the augmented vector of decision variables; $A \in \mathbb{R}^{\tilde{n},\tilde{n}}$ is symmetric and $A \succeq 0$; $B \in \mathbb{R}^{l,\tilde{n}}$, $C \in \mathbb{R}^{\tilde{m},\tilde{n}}$, $a \in \mathbb{R}^{\tilde{n}}$, $b \in \mathbb{R}^{l}$, and $c \in \mathbb{R}^{\tilde{m}}$. We define $\tilde{m}$ as the number of equality constraints after the slack variables introduction. Moreover, $\mathscr{B} \subseteq \{1, \ldots, \tilde{n}\}^3$ is the set of 3−tuples (i,j,k) that contains all the indices such that $x_i x_j = x_k$.

We provide some remarks before illustrating the transformation to obtain (10.5).

Remark A. If the transformation returns a QOP where A is not positive semidefinite, it is sufficient to introduce suitable quadratic constraints that reduce $\frac{1}{2} x^\top A x + a^\top x$ to a linear function, i.e., $A = 0$.

Remark B. Without loss of generality, we assume that for each $(i,j,k) \in \mathscr{B}$, the indices i, j and k are mutually different. If not, it is sufficient to add new slack variables. For example, in the presence of the constraint $x_i^2 = x_k$, i.e., $(i,i,k) \in \mathscr{B}$, we can define a new variable x_j and the constraint $x_j = x_i$ to obtain $x_i x_j = x_k$.

Remark C. Without loss of generality, we assume that each index $i, j, k \in \{1, \dots, \tilde{n}\}$ appears at most once in each tuple $(i, j, k) \in \mathscr{B}$. This assumption is without loss of generality because if an overlap occurs, we can introduce an additional variable x_h, for each repeated variable x_i, together with constraints of the kind $x_h = x_i$.

For simplicity, we describe the transformation algorithm from (10.4) to (10.5) through an illustrative example.

Example 10.2.1. *Let us consider the POP*

$$\min_{x \in \mathbb{R}^2} x_1^2 x_2.$$

By defining $x_3 = x_1^2$, *we obtain*

$$\min_{x \in \mathbb{R}^3} x_3 x_2 \quad \text{s.t. } \mathrm{x}_1^2 = \mathrm{x}_3.$$

Following Remark A, we notice that $A = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ *is not positive semi-definite. Therefore, we define* $x_4 = x_3 x_2$. *Moreover, according to Remark B, we define* $x_5 = x_1$ *to transform the constraint* $x_1^2 = x_3$ *into* $x_1 x_5 = x_3$. *In this way,* $\mathscr{B} = \{(3,2,4), (1,5,3)\}$. *To fulfill the non-overlap assumption described in Remark C, we add* $x_6 = x_3$.

In conclusion, we obtain the QOP

$$\begin{aligned} \min_{x \in \mathbb{R}^5} \quad & x_4 \\ & s.t. \\ & x_5 = x_1, \qquad x_6 = x_3 \\ & x_i x_j = x_k, \quad (i, j, k) \in \mathscr{B} = \{(3,2,4), (1,5,6)\}. \end{aligned}$$

Let us define the sets

$$\mathscr{P} \doteq \{x \in \mathbb{R}^{\tilde{n}} : x_i x_j = x_k, \quad \forall (i, j, k) \in \mathscr{B}\}. \tag{10.6}$$

and

$$\mathscr{D} \doteq \{x \in \mathbb{R}^{\tilde{n}} | Bx - b \leq 0, Cx - c = 0\}. \tag{10.7}$$

We notice that $\mathscr{P}$ is a non-convex semialgebraic set that describes the original problem's non-convexity. Instead, $\mathscr{D}$ is a convex set that includes all the linear constraints of (10.5). Given these definitions, we rewrite (10.5) as

$$\min_{x\in\mathscr{D}} \quad \frac{1}{2}x^\top Ax + a^\top x + \mathbb{1}_{\mathscr{P}}(x) \tag{10.8}$$

where $\mathbb{1}_{\mathscr{P}}(x)$ denotes the indicator function of $\mathscr{P}$, i.e.,

$$\mathbb{1}_{\mathscr{P}}(x) = \begin{cases} 0 & \text{if } x \in \mathscr{P} \\ \infty & \text{otherwise.} \end{cases} \tag{10.9}$$

Next, we prove we can solve the POP formulated as in (10.8) through ADMM.

In general, we can apply ADMM to composite problems of kind $\min_{x\in\mathbb{R}^n} F(x) + G(x)$; see, e.g., [161] and Section 4.2 for details. For this chapter's self-consistency, we provide the generic iteration of the ADMM algorithm. We denote by x_k, z_k, and u_k the estimates of the variables x, z, and u, respectively, at a generic k-th iteration of the algorithm.

ADMM consists of a loop iterating over three steps:

1. x-update:

$$\begin{aligned} x_{\tau+1} &= \arg\min_{x\in\mathbb{R}^n} L_\rho(x, z) \\ &= \arg\min_{x\in\mathbb{R}^n} F(x) + \frac{\rho}{2}\left\|x - z_\tau + u_\tau\right\|_2^2. \end{aligned} \tag{10.10a}$$

2. z-update:

$$\begin{aligned} z_{\tau+1} &= \arg\min_{z\in\mathbb{R}^n} L_\rho(x, z) \\ &= \arg\min_{z\in\mathbb{R}^n} G(z) + \frac{\rho}{2}\left\|x_{\tau+1} - z + u_\tau\right\|_2^2. \end{aligned} \tag{10.10b}$$

3. u-update:

$$u_{\tau+1} = u_\tau + x_{\tau+1} - z_{\tau+1}. \tag{10.10c}$$

We aim to apply ADMM to (10.8). To this purpose, we decompose (10.8) as in (4.23), by setting $F(x) = \frac{1}{2}x^\top Ax + a^\top x$ and $G(z) = \mathbb{1}_{\mathscr{D}}(z)$. Then, we have

$$\begin{aligned} \min_{x\in\mathscr{D}, z\in\mathbb{R}^{\tilde{n}}} \quad & \frac{1}{2}x^\top Ax + a^\top x + \mathbb{1}_{\mathscr{D}}(z) \\ \text{s.t.} \quad & x = z \end{aligned} \tag{10.11}$$

and we implement the algorithm as in (10.10a)-(10.10b)-(10.10c). While step (10.10c) is trivial, (10.10a) and (10.10b) require more computations.

10.2.1 x-update step

According to (10.10a), the x-update applied on (10.11) is

$$x_{\tau+1} = \arg\min_{x\in\mathscr{D}} \frac{1}{2}x^\top Ax + a^\top x + \frac{\rho}{2}\|x - z_\tau + u_\tau\|_2^2. \tag{10.12}$$

This is equivalent to

$$\begin{aligned} x_{\tau+1} = \arg\min_{x\in\mathbb{R}^{\tilde{n}}} \quad & \frac{1}{2}x^\top (A + \rho I_{\tilde{n}})x + x^\top [a + \rho(u_\tau - z_\tau)] \\ \text{s.t.} \quad & \\ & Bx - b \leq 0, \\ & Cx - c = 0 \end{aligned} \tag{10.13}$$

where $I_{\tilde{n}}$ is the identity matrix of dimension $\tilde{n}$. This convex quadratic program (QP) can be solved efficiently to global optimality.

Moreover, if no inequality constraints are present, the solution to (10.13) reduces to the solution to the linear system of the Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{bmatrix} A + \rho I & C^\top \\ C & 0 \end{bmatrix} \begin{bmatrix} x_{\tau+1} \\ \mu \end{bmatrix} = \begin{bmatrix} \rho(z_\tau - u_\tau) - a \\ c \end{bmatrix} \tag{10.14}$$

where μ is the Lagrange multiplier; see [167] for details. In Sections 10.3 and 10.4, we also discuss the relaxation of the equality constraints.

10.2.2 z-update step

According to (10.10b), the z-update applied on (10.11) is

$$z_{\tau+1} = \arg\min_{z\in\mathbb{R}^{\tilde{n}}} \mathbb{1}_{\mathscr{P}}(z) + \frac{\rho}{2}\|x_{\tau+1} - z + u_\tau\|_2^2. \tag{10.15}$$

This is equivalent to

$$z_{\tau+1} = \arg\min_{z\in\mathscr{P}} \|z - (x_{\tau+1} + u_\tau)\|_2^2. \tag{10.16}$$

Since $\mathscr{P}$ is non-convex, the solution to (10.16) is not straightforward. However, we can exploit the decoupling assumption discussed in Remark C to split the problem into smaller and affordable sub-problems. More precisely, on the one hand, the quadratic cost in (10.16) is separable in its components; on the other hand, according to the decoupling assumption, we can partition $\mathscr{P}$ into separated subsets, each of them representing a constraint $x_i x_j = x_k$. Let $t = 1,\dots,|\mathscr{B}|$ be the ordered indices of the elements of $\mathscr{B}$, i.e., each $(i,j,k) \in \mathscr{B}$ is labeled with a $t \in \{1,\dots,|\mathscr{B}|\}$. Then, we can write

$$\mathscr{P} = \mathscr{P}_1 \times \mathscr{P}_2 \times \cdots \times \mathscr{P}_{|\mathscr{B}|} \tag{10.17}$$

where

$$\mathscr{P}_t \doteq \{(x_i, x_j, x_k) : x_i x_j = x_k\} \text{ for each } t = 1,\dots,|\mathscr{B}|. \tag{10.18}$$

Hence,

$$z_{\tau+1} = \arg\min_{z\in\mathscr{P}_1\times\cdots\times\mathscr{P}_{|\mathscr{B}|}} \sum_{t=1}^{|\mathscr{B}|} \|z_t - (x_{\tau+1} + u_\tau)_t\|_2^2 \tag{10.19}$$

where $w_t \doteq (w_i, w_j, w_k)$ for any vector $w \in \mathbb{R}^{\tilde{n}}$ and (i,j,k) corresponding to the index t.

In conclusion, (10.19) can be split into $|\mathscr{B}|$ sub-problems with only three variables, i.e.,

$$(z_{\tau+1})_t = \arg\min_{z_t\in\mathscr{P}_t} \|z_t - (x_{\tau+1} + u_\tau)_t\|_2^2. \tag{10.20}$$

In other terms, $(z_{\tau+1})_t$ is the ℓ_2 projection of $v_t = (v_i, v_j, v_k) \doteq (x_{\tau+1} + u_\tau)_t \in \mathbb{R}^3$ onto $\mathscr{P}_t$. The problems in (10.20) are non-convex; however, we can solve them in

closed form. In fact, we have

$$\begin{aligned}(z_{\tau+1})_t = \arg\min_{z_t\in\mathbb{R}^3} &(z_i - v_i)^2 + (z_j - v_j)^2 + (z_k - v_k)^2 \\ \text{s.t.} \quad & z_i z_j = z_k.\end{aligned} \tag{10.21}$$

Then, we plug $z_k = z_i z_j$ into the cost function and remove the corresponding constraint:

$$(z_{\tau+1})_t = \arg\min_{z_t\in\mathbb{R}^3} (z_i - v_i)^2 + (z_j - v_j)^2 + (z_i z_j - v_k)^2. \tag{10.22}$$

Now, we can explicitly find the minima by evaluating the first order conditions: given $f_v(z_i, z_j) = (z_i - v_i)^2 + (z_j - v_j)^2 + (z_i z_j - v_k)^2$, we solve

$$\nabla f_v(z_i, z_j) = 2\begin{pmatrix} z_i - v_i + z_i z_j^2 - z_j v_k \\ z_j - v_j + z_i^2 z_j - z_i v_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{10.23}$$

From the first equation, we obtain

$$z_i - v_i + z_i z_j^2 - z_j v_k = 0 \Rightarrow z_i = \frac{v_i + z_j v_k}{1 + z_j^2} \tag{10.24}$$

Notice that this division is always properly defined since we assume that z_j assumes real values; therefore, $1 + z_j^2$ is never null.

By replacing (10.24) into the second equation of (10.23), we get

$$\begin{aligned} &z_j - v_j + \left(\frac{v_i + z_j v_k}{1 + z_j^2}\right)^2 z_j - \frac{v_i + z_j v_k}{1 + z_j^2} v_k = \\ &= \frac{z_j^5 + c_4 z_j^4 + c_3 z_j^3 + c_2 z_j^2 + c_1 z_j + c_0}{\left(1 + z_j^2\right)^2} = 0 \end{aligned} \tag{10.25}$$

where

$$\begin{aligned} &c_0 = -v_j - v_i v_k, \quad c_1 = v_i^2 - v_k^2 + 1, \\ &c_2 = v_i v_k - 2v_j, \quad c_3 = 2, \quad c_4 = -v_j. \end{aligned} \tag{10.26}$$

As $1 + z_j^2 \neq 0$, we obtain the candidate solutions by finding the zeros of a univariate polynomial of degree 5, e.g., by computing the eigenvalues of the companion

matrix

$$\left[\begin{array}{c|c} 0 & I_4 \\ \hline -c_0 & -c_1 \ \dots \ -c_4 \end{array}\right]. \tag{10.27}$$

In this way, we get five candidate solutions, which are either real or complex and conjugate, and at least one is real. We evaluate the cost function at each real candidate solution, which finally provides the desired minimum.

We notice that the z-update requires the computation of the eigenvalues of a 5×5 matrix for each bilinear constraint, which may be burdensome if $|\mathscr{B}|$ is large. However, the problem can be fully parallelized by exploiting the separability of $\mathscr{B}$; therefore, the z-update can be solved very effectively in the presence of suitable hardware. We summarize the proposed algorithm, referred to as ADMM4POP, in Algorithm 2.

Algorithm 2 ADMM4POP

Initialization: $z_0, u_0 \in \mathbb{R}^{\tilde{n}}$
for $\tau = 1, \dots, T_{stop}$ **do**
  $x_{\tau+1} = \arg\min_{x \in \mathscr{D}} \frac{1}{2} x^\top A x + a^\top x + \frac{\rho}{2} \|x - z_\tau + u_\tau\|_2^2,$
  via convex QP or KKT conditions, see Section 10.2.1
  for $t = 1, \dots, |\mathscr{B}|$ **do**
    $(z_{\tau+1})_t = \arg\min_{z_t \in \mathbb{R}^3} (z_i - v_i)^2 + (z_j - v_j)^2 + (z_i z_j - v_k)^2$
    by explicit computation, see Section 10.2.2
  end for
  $u_{\tau+1} = u_\tau + x_{\tau+1} - z_{\tau+1}$
end for

Concerning the stopping time T_{stop}, as suggested in [161], we stop the algorithm when the norms of both primal and dual residuals $\|x_\tau - z_\tau\|$ and $\|\rho(z_{\tau-1} - z_\tau)\|$ are sufficiently small.

10.3 Convergence guarantee

Proving the convergence of ADMM in non-convex problems is a challenging task. In [168], the authors prove the convergence to a stationary point of single-block ADMM for a family of non-convex problems. Later, in [163, 164], extensions to

multi-block problems are proposed. In [163], the terms of the cost functional that are non-convex must be differentiable to prove the convergence. This is not the case of (10.8), where the indicator function of a non-convex set occurs, which is non-convex and non-smooth. Since the development in [168] accounts for non-convex and non-smooth terms, we leverage their results to develop the convergence analysis of ADMM4POP.

We remark that the analysis in [168] is developed for unconstrained problems; thus, for simplicity, in this work, we recast problem (10.5) into unconstrained optimization as well, while we leave the proof of convergence of the constrained version for future extended work. To this end, in (10.5), we assume no inequality constraints $Bx \leq b$ are present. In contrast, the equality constraints $Cx = c$ are accounted for in a relaxed way, i.e., by adding the term $\gamma\|Cx-c\|_2^2$ to the objective function for some $\gamma > 0$. Finally, we keep the bilinear constraints by representing them through the indicator function, see (10.8). In conclusion, we consider

$$\min_{x\in\mathbb{R}^{\tilde{n}}} \quad \frac{1}{2}x^\top Ax + a^\top x + \gamma\|Cx-c\|_2^2 + \mathbb{1}_P(x). \tag{10.28}$$

As $\gamma\|Cx-c\|_2^2$ is quadratic and convex, (10.28) has the form of (10.8); then, we can apply ADMM4POP to it. We refer to this approach as relaxed ADMM4POP.

For completeness, we summarize a result from [168], which is the basis for our proof of convergence. Let us consider the composite problem (4.23) and its augmented Lagrangian (4.24) under the following conditions.

Assumption 8.

A1. F is semi-algebraic, twice continuously differentiable on $\mathbb{R}^n$ with a bounded, positive semidefinite Hessian $\nabla^2 F$;

A2. G is a semialgebraic, proper, lower semicontinuous function;

A3. there exists $0 < \zeta < \rho$ such that $F(x) - \frac{1}{2\zeta}\|\nabla F(x)\|^2$ is lower bounded;

A4. the design parameter ρ is designed so that $\rho I_n - \sqrt{2}\nabla^2 F(x) \succeq 0$.

We recall that a function is proper if it is finite somewhere and it does tend to $-\infty$; see, e.g., [168, Section 2].

Theorem 10.3.1. (Convergence of non-convex ADMM, [168, Theorems 1-2-4])

Let us suppose that Assumption 8 holds. Then, the sequence (x_k, z_k, u_k) *generated by ADMM converges to a point* $(x^\star, z^\star, u^\star)$*, and* $x^\star$ *is a stationary point of* $F(x)+G(x)$.

Theorem 10.3.2. (Convergence of ADMM4POP)

Let us consider (10.28)*. The sequence* (x_k, z_k, u_k) *generated by relaxed ADMM4POP converges to a point* $(x^\star, z^\star, u^\star)$*, and* $x^\star$ *is a stationary point of* (10.28).

Proof. Let us consider $F(x) = \frac{1}{2}x^\top Ax + a^\top x + \gamma\|Cx-c\|_2^2$ and $G(z) = \mathbb{1}_P(z)$. In the following, we prove that all the points of Assumption 8 are fulfilled, which is sufficient to prove the thesis.

By construction of A, it is straightforward to prove that $\frac{1}{2}x^\top Ax + a^\top x + \gamma\|Cx-c\|_2^2$ fulfills A1.

Regarding A2, by definition, $\mathbb{1}_P$ is proper. Moreover, it is semialgebraic because $\mathscr{P}$ is a semialgebraic set, and the indicator function of a semialgebraic set is semialgebraic, see, e.g., [169]. Finally, the indicator function of a set is lower semicontinuous if the set is closed; then, we need to show that $\mathscr{P}$ is closed. For this purpose, let t be the index of (i,j,k) as defined in Section 10.2 and let us define

$$\mathscr{Q}_t \doteq \left\{z \in \mathbb{R}^{\tilde{n}} : z_i z_j = z_k\right\}. \tag{10.29}$$

We remark that $\mathscr{Q}_t$ differs from $\mathscr{P}_t$ defined in (10.17). In particular, it holds $\mathscr{P} = \bigcup_{t\in\mathscr{B}} \mathscr{Q}_t$. Since $\mathscr{Q}_t = \left\{z : -z_i z_j + z_k \leq 0\right\} \cup \left\{z : z_i z_j - z_k \leq 0\right\}$ and since $z_i z_j - z_k$ and $-z_i z_j + z_k$ are continuous functions, their sub-level sets are closed. As a consequence, $\mathscr{Q}_t$ is an intersection of closed sets; thus, it is closed. In turn, $\mathscr{P}$ is a union of closed sets; hence it is closed. Then, we conclude that $\mathbb{1}_{\mathscr{P}}$ is lower semicontinuous.

Finally, A3 holds because, in our problem, $F(x) - \frac{1}{2\zeta}\|\nabla F(x)\|^2$ is convex if we suitably design $\rho > \zeta > \|A + \gamma C^\top C\|_2$. In particular, if we set $\rho > \sqrt{2}\|A+\gamma C^\top C\|_2$, A4 is satisfied as well. □

10.4 Numerical examples

In this section, we propose two numerical experiments to test the effectiveness of the proposed ADMM4POP and to validate the theoretical convergence results. For ADMM4POP, we implement both the constrained and relaxed versions, which is guaranteed to converge by Theorem 10.3.2. However, the constrained version is also convergent in our experiments.

We compare ADMM4POP to two state-of-the-art methods: the interior-point method (IPM) and the SDP approach [152]. IPM is a local algorithm like ADMM4POP. All the experiments are performed with MATLAB R2021b on a PC with AMD Ryzen 7 1700 8-Core CPU @ 3.00 GHz and 16 GB RAM. We implement IPM through the function *fmincon* in the MATLAB Optimization Toolbox, while we use the SparsePOP package [155] to implement the SDP method.

10.4.1 Academic example

In the first experiment, we consider the following POP:

$$\begin{aligned} \min_{x\in\mathbb{R}^3} \quad & x_1^2x_2^2 + x_1^2 + q_1x_1 + x_2^2 + x_2x_3 + q_2x_2 + x_3^2 + q_3x_3 \\ & \text{s.t.} \\ & x_2x_3 + x_1 = 10 \end{aligned}$$

where $q_1 \in [4,6]$, $q_2 \in [-8,-6]$, and $q_3 \in [1,3]$ are generated uniformly at random. We reformulate the problem as in (10.5). First, we replace $x_4 = x_2x_3$ and $x_5 = x_1x_2$. Then, we define $x_6 = x_2$ to fulfill the splitting condition of Remark C. In conclusion, we obtain

$$\begin{aligned} \min_{x\in\mathbb{R}^6} \quad & x_5^2 + x_1^2 + q_1x_1 + x_2^2 + x_4 + q_2x_2 + x_3^2 + q_3x_3 \\ & \text{s.t.} \\ & x_4 + x_1 = 10, \qquad x_6 = x_2 \\ & x_ix_j = x_k, \ (i,j,k) \in \mathscr{B} = \{(2,3,4),(1,6,5)\} \end{aligned}$$

which is in the form (10.8). In particular, verifying that the quadratic cost function is convex is straightforward.

The stop threshold for ADMM4POP is set so that the obtained accuracy is comparable to that of IPM. We randomly initialize all the variables with standard Gaussian distribution. We set $\rho = 2$, and $\gamma = 10^3$ for relaxed ADMM4POP. In this experiment, SDP with relaxation order two always achieves the global minimum. Thus, it is used as a reference.

In Table 10.1, we report the results over 500 random runs in terms of accuracy and runtime statistics. We evaluate the error as the ℓ_2 distance from the SDP solution, considered ground truth. Both ADMM4POP and IPM substantially achieve the desired global minimum. As expected, the error is slightly larger in the relaxed version of ADMM4POP. However, the gap is irrelevant in practice, and the global minimizer has been identified with sufficient accuracy. As to the runtime, we can see that ADMM4POP is significantly faster than IPM and SDP.

	Error	Runtime (ms)		
	mean	mean	min	max
SDP	–	63.5	59.0	88.5
IPM	2.5×10^{-5}	12.3	7.7	21.8
ADMM4POP cns	6.5×10^{-5}	1.6	1.5	7.9
ADMM4POP rel	4.2×10^{-4}	1.7	1.6	8.9

Table 10.1 Academic ADMM4POP example: Constrained POP; statistics over 500 random runs. ADMM4POP is implemented in constrained (cns) and relaxed (rel) versions.

10.4.2 LTI system identification

In the second experiment, we deal with a system identification problem. We consider the DT LTI system $y_{t+1} = \alpha y_t + \beta u_{t+1}$, $t = 0, \dots, T$, and we aim at recovering the parameters α and β given the knowledge of the input $u \in \mathbb{R}^T$ and noisy measurements of the output $w \doteq y + \eta$, where $\eta \in \mathbb{R}^T$ represents the noise. In particular, we minimize the simulation error ℓ_2-norm, i.e., we solve

$$\begin{aligned} \min_{\alpha,\beta\in\mathbb{R},y\in\mathbb{R}^T} \quad & \|y - w\|_2^2 \\ & \text{s.t.} \\ & y_{t+1} = \alpha y_t + \beta u_{t+1}, \qquad t = 1, \dots, T-1. \end{aligned}$$

As illustrated in Section 10.2, we define the slack variables $\psi_t = \alpha$ and $\xi_t = \psi_t y_t$ for $t = 1, \dots, T-1$, to decouple the bilinear constraints. Thus,

$$\theta = (\alpha, \beta, y_1, \dots, y_T, \psi_1, \dots, \psi_{T-1}, \xi_1, \dots, \xi_{T-1}) \in \mathbb{R}^{2+T+2(T-1)}$$

is the total vector of variables, and

$$\mathscr{B} = \{(2+h, 2+T+h, 1+2T+h), h = 1, \dots, T-1\}.$$

	Error	Runtime (s)		
	Mean	mean	min	max
SDP	6.1×10^{-4}	10.16	5.87	22.99
IPM	3.2×10^{-4}	3.97	1.34	36.73
ADMM4POP cns	3.6×10^{-4}	2.53	0.76	21.45
ADMM4POP rel	3.9×10^{-4}	0.62	0.40	1.61

Table 10.2 ADMM4POP for linear SI: Statistics over 500 random runs. ADMM4POP is implemented in constrained (cns) and relaxed (rel) versions.

We perform 500 random runs, with $\alpha, \beta \in (-1, -0.5) \cup (0.5, 1)$ generated uniformly at random; η is a white Gaussian noise with variance 10^{-4}. The variables are initialized with Gaussian distribution. In each run, we measure $T = 500$ output samples. Thus, the total number of variables is significantly larger than in the first experiment. For ADMM4POP, we set $\rho = 1$ and $\gamma = 10$. For the SDP approach, we set the relaxation order to 2, which is observed to be sufficient to achieve the global minimum.

In Table 10.2, we collect the results. We define the error as the ℓ_2 distance between the estimates and the true parameters α, β. All the considered algorithms identify the correct parameters with similar accuracy. Regarding the runtime, ADMM4POP is faster than the other algorithms, with a more evident improvement in the relaxed version.

We remark that, in the proposed experiments, we perform the decoupled projections (10.20) sequentially, which leaves room for further enhancement via parallelization, in particular in view of larger dimension problems.

The main drawback of the proposed ADMM4POP algorithm when performing LTI system identification is that the number of slack optimization variables to be introduced grows quickly when the system's order increases, limiting the ap-

proach's applicability. Future work will envisage developing strategies to overcome this limitation and theoretically analyzing convergence in inequality-constrained problems.

Chapter 11

Conclusions

This thesis investigates the system identification (SI) of dynamical nonlinear input-output (NIO) systems.

We start by formulating the SI problem using a simulation error minimization (SEM) approach. We illustrate the origins of the vanishing and exploding gradient issues through a simple example and suggest a constrained optimization formulation to prevent these issues.

Given the high computational demands of standard methods in constrained optimization, we develop novel, efficient, first-order optimization algorithms for equality-constrained problems. To achieve this, we recast the optimization problem into an equivalent control problem and show that we can derive new optimization algorithms by appropriately defining controllers. We utilize proportional-integral (PI) control and feedback linearization (FL) to develop two novel algorithms, named PI-CMO and FL-CMO, respectively.

Subsequently, we extend the CMO approach to deal with problems with linear inequality constraints. In this scenario, we propose a solution that modifies the PI control law, the modified PI-CMO algorithm.

We analyzed the convergence of all proposed algorithms; in particular, we showed that all methods are globally and exponentially convergent when the optimization problem is convex, and FL-CMO is also locally convergent in non-convex cases. Moreover, we compared their performance with related methods.

We applied the FL-CMO algorithm to the training of NIO models. Specifically, we formulated the identification problem as constrained optimization and employed adaptive Euler-Heun integration of the FL-CMO differential equation. We performed a theoretical analysis connecting our approach with the standard unconstrained formulation, illustrating that points satisfying the first-order optimality conditions of the constrained formulation are stationary points of the unconstrained formulation. Next, we addressed the primary computational bottleneck of the algorithm's iterations by leveraging the problem structure and employing sparse Q-less QR factorization.

We conducted extensive numerical experiments on both black-box and gray-box SI problems. The results of black-box problems demonstrated that our approach significantly outperforms standard methods for training neural models. Additionally, we found that adequately trained neural NIO models can be more effective than standard neural models like LSTM and GRU. The results from the gray-box experiment indicate that our proposed approach outperforms standard solutions based on gradient algorithms.

The findings of this study enhance our theoretical understanding of the interplay between mathematical optimization and control theory. Furthermore, applying the proposed algorithm to nonlinear SI has resulted in an effective identification method. We have shown that shifting from complex unconstrained optimization to well-structured constrained optimization provides several advantages. First, it facilitates the development of novel algorithms with guaranteed convergence. Second, it offers a framework for systematically addressing vanishing and exploding gradients, prevalent issues in SI and machine learning. The latter advantage is particularly relevant since many undesirable behaviors in dynamic systems learning, such as slow convergence and inaccurate results, can be traced back to the vanishing gradient.

We expect that applying the proposed approach in complex real-world applications will yield more effective and efficient learning. The reduced computational effort paves the way for further research into energy-efficient and environmentally friendly learning. Additionally, the increased accuracy of estimates enables the design of more effective feedback control systems.

However, this study does have its limitations. The assumptions of differentiability and the linearity of inequality constraints limit the applicability of the proposed optimization algorithms. Moreover, the computational cost of the training based

on FL-CMO grows quadratically with the amount of data, making it unsuitable for large-scale problems.

Future work on identification through FL-CMO will aim to develop a mini-batch version of the algorithm to improve its applicability to large datasets. Regarding the CMO theoretical framework, we intend to create new algorithms tailored for different classes of optimization problems, including non-smooth optimization, inequality-constrained convex optimization, and mixed-integer programming.

In conclusion, this research has demonstrated the potential of control theory in developing novel optimization algorithms for constrained minimization, which we have successfully applied to the problem of learning dynamical systems.

References

[1] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, 1999.

[2] D. J. Wagg, K. Worden, R. J. Barthorpe, and P. Gardner. Digital twins: state-of-the-art and future directions for modeling and simulation in engineering dynamics applications. *ASCE-ASME J. Risk Uncertain. Eng. Syst. B: Mech. Eng.*, 6(3):030901, 2020.

[3] M. Milanese and M. Taragna. H_∞ set membership identification: A survey. *Automatica*, 41(12):2019–2032, 2005.

[4] J. A. Rossiter. *Model-based predictive control: a practical approach*. CRC press, 2017.

[5] J. M. Hokanson, G. Iaccarino, and A. Doostan. Simultaneous identification and denoising of dynamical systems. *SIAM J. Sci. Comput.*, 45(4):A1413–A1437, 2023.

[6] A. Karimi, K. Van Heusden, and D. Bonvin. Non-iterative data-driven controller tuning using the correlation approach. In *Proc. Eur. Control Conf. (ECC)*, pages 5189–5195. IEEE, 2007.

[7] M. C. Campi, A. Lecchini, and S. M. Savaresi. Virtual reference feedback tuning: a direct method for the design of feedback controllers. *Automatica*, 38(8):1337–1346, 2002.

[8] M. C. Campi and S. M. Savaresi. Direct nonlinear control design: The virtual reference feedback tuning (VRFT) approach. *IEEE Trans. Autom. Control*, 51(1):14–27, 2006.

[9] C. Novara, M. Canale, M. Milanese, and M. C. Signorile. Set membership inversion and robust control from data of nonlinear systems. *Int. J. Robust Nonlinear Control*, 24(18):3170–3195, 2014.

[10] M. Abuabiah, V. Cerone, S. Pirrera, and D. Regruto. A non-iterative approach to direct data-driven control design of MIMO LTI systems. *IEEE Access*, 11:121671–121687, 2023.

[11] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung. Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3):657–682, 2014.

[12] Q. Zhang. Nonlinear system identification with output error model through stabilized simulation. *Proc. Symp. Nonlinear Control Syst. (NOLCOS)*, 37:501–506, 2004.

[13] L. Piroddi. Simulation error minimisation methods for narx model identification. *Int. J. Model. Identif. Control*, 3(4):392–403, 2008.

[14] T. Söderström and P. Stoica. Some properties of the output error method. *Automatica*, 18(1):93–99, 1982.

[15] M. Farina and L. Piroddi. Convergence properties of an iterative prediction approach to nonlinear sem parameter estimation. In *IEEE Conf. Decis. Control (CDC)*, pages 7226–7231. IEEE, 2010.

[16] V. Cerone, J.-B. Lasserre, D. Piga, and D. Regruto. A unified framework for solving a general class of conditional and robust set-membership estimation problems. *IEEE Trans. Autom. Control*, 59(11):2897–2909, 2014.

[17] V. Cerone, S. M. Fosson, S. Pirrera, and D. Regruto. Alternating direction method of multipliers for polynomial optimization. In *Proc. Eur. Control Conf. (ECC)*, pages 1–6, 2023.

[18] I. Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.

[19] F. M. Salem. *Recurrent Neural Networks*. Springer, 2022.

[20] D. T. Mirikitani and N. Nikolaev. Recursive bayesian recurrent neural networks for time-series modeling. *IEEE Trans. Neural Netw.*, 21(2):262–274, 2009.

[21] C.-C. Peng and G. D. Magoulas. Nonmonotone BFGS-trained recurrent neural networks for temporal sequence processing. *Appl. Math. Comput.*, 217(12):5421–5441, 2011.

[22] X. Liu, S. Liu, J. Sha, J. Yu, Z. Xu, X. Chen, and H. Meng. Limited-memory bfgs optimization of recurrent neural network language models for speech recognition. In *Proc IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 6114–6118. IEEE, 2018.

[23] A. Bemporad. Linear and nonlinear system identification under ℓ_1-and group-lasso regularization via L-BFGS-B. *arXiv preprint arXiv:2403.03827*, 2024.

[24] K. S. Narendra and K. Parthasarathy. Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Trans. Neural Netw.*, 2(2):252–262, 1991.

[25] R. J. Williams. A learning algorithm for continually running fully recurrent neural netwokrs. *Neural Comput.*, 1:256–263, 1989.

[26] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78(10):1550–1560, 1990.

[27] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *J. Marchine Learn. Res.*, 18:1–43, 2018.

[28] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 5(2):157–166, 1994.

[29] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 1310–1318. Pmlr, 2013.

[30] A. Ramaswamy. Gradient clipping in deep learning: A dynamical systems perspective. In *Int. Conf. Pattern Recognit. Appl. Methods (ICPRAM)*, volume 1, pages 107–114, 2023.

[31] L. Medsker and L. C. Jain. *Recurrent neural networks: design and applications*. CRC press, 1999.

[32] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön. Deep learning and system identification. *IFAC-PapersOnLine*, 53(2):1175–1181, 2020.

[33] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Comput. Surv.*, 55(6):1–36, 2022.

[34] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig. Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. *J. Artif. Intell. Soft Comput. Res.*, 9(4):235–245, 2019.

[35] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing LSTM language models. In *Proc. Int. Conf. Learn. Represent. (ICRL)*, 2018.

[36] A. Blanco, M. Delgado, and M. C. Pegalajar. A real-coded genetic algorithm for training recurrent neural networks. *Neural Netw.*, 14(1):93–105, 2001.

[37] E. Bas, E. Egrioglu, and E. Kolemen. Training simple recurrent deep artificial neural network for forecasting using particle swarm optimization. *Granular Computing*, 7(2):411–420, 2022.

[38] A. D. Adeoye and A. Bemporad. An inexact sequential quadratic programming method for learning and control of recurrent neural networks. *IEEE Trans. Neural Netw. and Learning Systems*, 2024.

[39] M. Milanese, J. Norton, H. Piet-Lahanier, and É. Walter. *Bounding Approaches to System Identification*. Springer US, 2013.

[40] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 3rd edition, 2016.

[41] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 2006.

[42] A. O'dwyer. *Handbook of PI and PID controller tuning rules*. World Scientific, 2009.

[43] A. Isidori. *Nonlinear Control Systems*. Springer London, 1995.

[44] V. Cerone, S. M. Fosson, S. Pirrera, and D. Regruto. A new framework for constrained optimization via feedback control of lagrange multipliers. *arXiv preprint arXiv:2403.12738*, 2024.

[45] V. Cerone, S. M. Fosson, S. Pirrera, and D. Regruto. A feedback control approach to convex optimization with inequality constraints. In *Proc. IEEE Conf. Decis. Control (CDC)*. IEEE, 2024.

[46] V. Cerone, S. M. Fosson, S. Pirrera, and D. Regruto. A constrained optimization approach to system identification of nonlinear input-output models. *Manuscript in preparation*, 2024.

[47] F. Giri E.W. Bai. *Block-oriented Nonlinear System Identification*. Lecture Notes in Control and Information Sciences n404. Springer, 1 edition, 2010.

[48] C. K. I. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[49] C. H. Moog, Y. Zheng, and P. Liu. Input-output equivalence of nonlinear systems and their realizations. In *Proc. IFAC World Congr.*, volume 35, pages 265–270, 2002.

[50] G. Conte, C. H. Moog, and A. M. Perdon. *Nonlinear control systems: An algebraic setting*, volume 242. Springer, 1999.

[51] P. Bernard, V. Andrieu, and D. Astolfi. Observer design for continuous-time dynamical systems. *Annual Reviews in Control*, 53:224–248, 2022.

[52] J. DiStefano and C. Cobelli. On parameter and structural identifiability: Nonunique observability/reconstructibility for identifiable systems, other ambiguities, and new definitions. *IEEE Transactions on Automatic Control*, 25(4):830–833, 1980.

[53] A. Isidori. *Nonlinear Control Systems*. Springer London, 1995.

[54] V. Cerone, D. Piga, and D. Regruto. Set-membership error-in-variables identification through convex relaxation techniques. *IEEE Trans. Autom. Control*, 57(2):517–522, 2012.

[55] P. Van Overschee and B. L. De Moor. *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media, 2012.

[56] T. P. Bohlin. *Practical grey-box process identification: theory and applications*. Springer Science & Business Media, 2006.

[57] O. Prot and G. Mercère. Combining linear algebra and numerical optimization for gray-box affine state-space model identification. *IEEE Trans. Autom. Control*, 65(8):3272–3285, 2019.

[58] L. Ljung. Approaches to identification of nonlinear systems. In *Proc. 29th Chin. Control Conf.*, pages 1–5, 2010.

[59] M. Schoukens and K. Tiels. Identification of block-oriented nonlinear systems starting from linear approximations: A survey. *Automatica*, 85:272–292, 2017.

[60] V. Cerone, D. Piga, and D. Regruto. Set-membership identification of block-structured nonlinear feedback systems. In *Proc. 48h IEEE Conf. Decis. Control held jointly 28th Chin. Control Conf.*, pages 3643–3649. IEEE, 2009.

[61] V. Cerone, V. Razza, and D. Regruto. One-shot set-membership identification of generalized Hammerstein–Wiener systems. *Automatica*, 118:109028, 2020.

[62] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. B: Stat. Methodol.*, 58(1):267–288, 1996.

[63] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci.*, 113(15):3932–3937, 2016.

[64] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[65] C. Novara. Sparse identification of nonlinear functions and parametric set membership optimality analysis. *IEEE Trans. Autom. Control*, 57(12):3236–3241, 2012.

[66] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[67] J. A. K. Suykens, B. L. R. De Moor, and J. Vandewalle. Nonlinear system identification using neural state space models, applicable to robust control design. *Int. J. Control*, 62(1):129–152, 1995.

[68] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[69] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45(11):2673–2681, 1997.

[70] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

[71] K. Smagulova and A. P. James. A survey on LSTM memristive neural network architectures and applications. *Eur. Phys. J. Spec. Top.*, 228(10):2313–2324, 2019.

[72] T. Lin, B. G. Horne, P. Tino, and C. L. Giles. Learning long-term dependencies in narx recurrent neural networks. *IEEE Trans. Neural Netw.*, 7(6):1329–1338, 1996.

[73] J. Sjöberg, H. Hjalmarsson, and L. Ljung. Neural networks in system identification. In *Proc. IFAC Symp. Syst. Identif. (SYSID)*, volume 27, pages 359–382. Elsevier, 1994.

[74] M. Nørgaard, O. Ravn, and N. K. Poulsen. NNSYSID-toolbox for system identification with neural networks. *Mathematical and computer modelling of dynamical systems*, 8(1):1–20, 2002.

[75] H.T. Siegelmann, B. G. Horne, and C. L. Giles. Computational capabilities of recurrent narx neural networks. *IEEE Trans. Syst. Man Cybern. B (Cybern.)*, 27(2):208–215, 1997.

[76] H. K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 3rd edition, 2002.

[77] A. Isidori. *Nonlinear Control Systems II*. Springer London, 1999.

[78] J.-J. E. Slotine, W. Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[79] E. D. Sontag. *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer Science & Business Media, 2013.

[80] C. I. Byrnes and A. Isidori. A frequency domain philosophy for nonlinear systems, with applications to stabilization and to adaptive control. In *IEEE Conf. Decis. Control (CDC)*, pages 1569–1573, 1984.

[81] A. Isidori. The zero dynamics of a nonlinear system: From the origin to the latest progresses of a long successful story. *Europ. J. Control*, 19(5):369–378, 2013.

[82] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[83] J. J. Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*, pages 105–116. Springer, 2006.

[84] G. Braun, A. Carderera, C. W. Combettes, H. Hassani, A. Karbasi, A. Mokhtari, and S. Pokutta. Conditional gradient methods. *arXiv preprint arXiv:2211.14103*, 2022.

[85] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends® Mach. Learn.*, 3(1):1–122, 2011.

[86] Y. Shen, Z. Wen, and Y. Zhang. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optim. Methods Softw.*, 29(2):239–263, 2014.

[87] A. P. Liavas and N. D. Sidiropoulos. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Trans. Signal Process.*, 63(20):5450–5463, 2015.

[88] L. Yang, T. K. Pong, and X. Chen. Alternating direction method of multipliers for nonconvex background/foreground extraction. *arXiv preprint arXiv:1506.07029*, 1(5):5, 2015.

[89] A. Bhaya and E. Kaszkurewicz. A control-theoretic approach to the design of zero finding numerical methods. *IEEE Trans. Autom. Control*, 52(6):1014–1026, 2007.

[90] J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[91] L. Calogero, M. Pagone, and A. Rizzo. Enhanced quadratic programming via pseudo-transient continuation: An application to model predictive control. *IEEE Control Syst. Lett.*, 8:1661–1666, 2024.

[92] S. Lojasiewicz. Sur les trajectoires du gradient d'une fonction analytique. *Seminari di geometria*, 1983:115–117, 1982.

[93] F. Bullo. *Contraction Theory for Dynamical Systems*. Kindle Direct Publishing, 1.1 edition, 2023.

[94] F. Santambrogio. Euclidean, metric, and Wasserstein gradient flows: an overview. *Bull. Math. Sci.*, 7:87–154, 2017.

[95] P.-A. Absil, R. Mahony, and B. Andrews. Convergence of the iterates of descent methods for analytic cost functions. *SIAM J. Optim.*, 16(2):531–547, 2005.

[96] C. A. Botsaris. Differential gradient methods. *J. Math. Anal. Appl.*, 63(1):177–198, 1978.

[97] J. A. Amaya. A differential equations approach to function minimization. *Rev. Mat. Apl.*, 9:1–7, 1987.

[98] W. Su, S. Boyd, and E. J. Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. *J. Mach. Learn. Res.*, 17(153):1–43, 2016.

[99] M. Muehlebach and M. Jordan. A dynamical systems perspective on nesterov acceleration. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 4656–4662. PMLR, 2019.

[100] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM J. Optim.*, 26(1):57–95, 2016.

[101] B. Hu and L. Lessard. Control interpretations for first-order optimization methods. In *Proc. Amer. Control Conf. (ACC)*, pages 3114–3119, 2017.

[102] G. França, D. P. Robinson, and R. Vidal. ADMM and accelerated ADMM as continuous dynamical systems. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 1559–1567, 2018.

[103] G. Qu and N. Li. On the Exponential Stability of Primal-Dual Gradient Dynamics. *IEEE Control Syst. Lett.*, 3(1):43–48, 2019.

[104] Y. Tang, G. Qu, and N. Li. Semi-global exponential stability of augmented primal–dual gradient dynamics for constrained convex optimization. *Syst. & Control Lett.*, 144:104754, 2020.

[105] K. Tanabe. An algorithm for constrained maximization in nonlinear programming. *J. Oper. Res. Soc. Jpn*, 17:184–201, 1974.

[106] J. B. Rosen. The gradient projection method for nonlinear programming. part I. linear constraints. *J. Soc. Ind. Appl. Math.*, 8(1):181–217, 1960.

[107] H. Yamashita. A differential equation approach to nonlinear programming. *Math. Program.*, 18(1):155–168, 1980.

[108] K. Tanabe. A geometric method in nonlinear programming. *J. Optim. Theory Appl.*, 30:181–210, 1980.

[109] Y. G. Evtushenko and V. G. Zhadan. Stable barrier-projection and barrier-newton methods in nonlinear programming. *Optim. Methods Softw.*, 3(1-3):237–256, 1994.

[110] S. Wang, X. Q. Yang, and K. L. Teo. A unified gradient flow approach to constrained nonlinear optimization problems. *Comput. Optim. Appl.*, 25:251–268, 2003.

[111] J. Schropp and I. Singer. A dynamical systems approach to constrained minimization. *Numer. Funct. Anal. Optim.*, 21(3-4):537–551, 2000.

[112] Y. S. Xia and J. Wang. On the stability of globally projected dynamical systems. *J. Optim. Theory Appl.*, 106:129–150, 2000.

[113] A. Allibhoy and J. Cortés. Control barrier function-based design of gradient flows for constrained nonlinear programming. *IEEE Trans. Autom. Control*, 2023.

[114] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, NY, USA., 1982.

[115] D. Feijer and F. Paganini. Stability of primal-dual gradient dynamics and applications to network optimization. *Automatica*, 46(12):1974–1981, 2010.

[116] A. Cherukuri, E. Mallada, and J. Cortés. Asymptotic convergence of constrained primal–dual dynamics. *Syst. & Control Lett.*, 87:10–15, 2016.

[117] M. Mönnigmann. Efficient calculation of bounds on spectra of Hessian matrices. *SIAM J. Sci. Comput.*, 30(5):2340–2357, 2008.

[118] A. F. Filippov. Differential equations with discontinuous right-hand side. *Matematicheskii sbornik*, 93(1):99–128, 1960.

[119] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. https://cvxr.com/cvx, 2014.

[120] P. M. J. Van Den Hof, P.S.C. Heuberger, and J. Bokor. System identification with generalized orthonormal basis functions. *Automatica*, 31(12):1821–1834, 1995.

[121] B. Wahlberg and P. M. Mäkilä. On approximation of stable linear dynamical systems using laguerre and kautz functions. *Automatica*, 32(5):693–708, 1996.

[122] J. Schropp and I. Singer. A dynamical systems approach to constrained minimization. *Numer. Funct. Anal. Optim.*, 21:537–551, 2000.

[123] Peter Dorato, Vito Cerone, and Chaouki Abdallah. *Linear-Quadratic Control: An Introduction*. Simon & Schuster, Inc., USA, 1994.

[124] Mingyi Hong, Meisam Razaviyayn, and Jason Lee. Gradient primal-dual algorithm converges to second-order stationary solution for nonconvex distributed optimization over networks. In *Proc. Inter. Conf. Machine Learn. (ICML)*, volume 80, pages 2009–2018, 2018.

[125] A. Ferrara, G. P. Incremona, and M. Cucuzzella. *Advanced and optimization based sliding mode control: Theory and applications*. SIAM, 2019.

[126] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2013.

[127] A. Aggarwal and C.A. Floudas. Synthesis of general distillation sequences-nonsharp separations. *Comput. Chemical Eng.*, 14(6):631–653, 1990.

[128] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm Evol. Comput.*, 56:100693, 2020.

[129] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 2nd edition, 1999.

[130] Marcello Farina and Luigi Piroddi. Simulation error minimization identification based on multi-stage prediction. *Int. J. Adapt. Control Signal Process.*, 25(5):389–406, 2011.

[131] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[132] G. Wanner and E. Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

[133] J. Sundnes. *Adaptive Time Step Methods*, pages 61–77. Springer Nature Switzerland, 2024.

[134] S. Laue, M. Mitterreiter, and J. Giesen. Computing higher order derivatives of matrix and tensor expressions. In *Proc. Adv. Neural Infor. Process. Syst. (NeurIPS)*, volume 31, 2018.

[135] G. M. Stewart. *Matrix algorithms: volume 1: basic decompositions*. SIAM, 1998.

[136] J. Wang, A. Sano, T. Chen, and B. Huang. Identification of hammerstein systems without explicit parameterisation of non-linearity. *Int. J. Control*, 82(5):937–952, 2009.

[137] F. Chollet et al. Keras. https://github.com/keras-team/keras, 2015.

[138] P. M. Nørgård, O. Ravn, L. K. Hansen, and N. K. Poulsen. The nnsysid toolbox - a matlab toolbox for system identification with neural networks. In *IEEE Symp. Comput.-Aided Control Syst. Des.*, pages 374–379. IEEE, 1996.

[139] A. Bemporad. Recurrent neural network training with convex loss and regularization functions by extended kalman filtering. *IEEE Trans. Autom. Control*, pages 1–8, 2022.

[140] J.-P. Noel and M. Schoukens. Hysteretic benchmark with a dynamic nonlinearity. In *Workshop on nonlinear system identification benchmarks*, pages 7–14, 2016.

[141] A. F. Esfahani, P. Dreesen, K. Tiels, J.-P. Noël, and J. Schoukens. Polynomial state-space model decoupling for the identification of hysteretic systems. In *Proc. IFAC World Congr.*, volume 50, pages 458–463. Elsevier, 2017.

[142] M. Forgione and D. Piga. dynonet: A neural network architecture for learning dynamical systems. *Int. J. Adapt. Control Signal Process.*, 35(4):612–626, 2021.

[143] J. Belz, T. Münker, T. O. Heinz, G. Kampmann, and O. Nelles. Automatic modeling with local model networks for benchmark processes. In *Proc. IFAC World Congress*, volume 50, pages 470–475, 2017.

[144] V. Cerone, D. Piga, and D. Regruto. Enforcing stability constraints in set-membership identification of linear dynamic systems. *Automatica*, 47(11):2488–2494, 2011.

[145] C. Feng, C. M. Lagoa, and M. Sznaier. Hybrid system identification via sparse polynomial optimization. In *Proc. Amer. Control Conf. (ACC)*, pages 160–165, 2010.

[146] M. Canale, V. Cerone, D. Piga, and D. Regruto. Approximation of model predictive control laws for polynomial systems. *Asian J. Control*, 16(5):1425–1436, 2014.

[147] E. Harinath, L. C. Foguth, J. A. Paulson, and R. D. Braatz. Nonlinear model predictive control using polynomial optimization methods. In *Proc. Amer. Control Conf. (ACC)*, pages 1–6, 2016.

[148] D. Rodrigues and A. Mesbah. Efficient global solutions to single-input optimal control problems via approximation by sum-of-squares polynomials. *IEEE Trans. Autom. Control*, 67(9):4674–4686, 2022.

[149] S. Sedighi, K. V. Mishra, M. R. B. Shankar, and B. Ottersten. Localization with one-bit passive radars in narrowband internet-of-things using multivariate polynomial optimization. *IEEE Trans. Signal Process.*, 69:2525–2540, 2021.

[150] Y. Zhu and K. Savla. Information design in nonatomic routing games with partial participation: Computation and properties. *IEEE Trans. Control Netw. Syst.*, 9(2):613–624, 2022.

[151] T. Chen, J.-B. Lasserre, V. Magron, and E. Pauwels. Semialgebraic optimization for Lipschitz constants of ReLU networks. In *Adv. Neural Inform. Process. Syst. (NeurIPS)*, volume 33, pages 19189–19200, 2020.

[152] J.-B. Lasserre. *An introduction to polynomial and semi-algebraic optimization*. Cambridge Univ. Press, 2015.

[153] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Program.*, 96(2):293–320, 2003.

[154] N.-Z. Shor. Quadratic optimization problems. *Sov. J. Comput. Syst. Sci*, 26, 1987.

[155] H. Waki, S. Kim, M. Kojima, M. Muramatsu, and H. Sugimoto. SparsePOP: a sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Trans. Math. Softw.*, 35, 2008.

[156] J. Wang, V. Magron, and J.-B. Lasserre. TSSOS: A moment-SOS hierarchy that exploits term sparsity. *SIAM J. Optim.*, 31(1):30–58, 2021.

[157] T. Weisser, J.-B. Lasserre, and K.-C. Toh. Sparse-BSOS: a bounded degree SOS hierarchy for large scale polynomial optimization with sparsity. *Math. Programm. Comput.*, 10:1–32, 2018.

[158] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn. Chordal decomposition in operator-splitting methods for sparse semidefinite programs. *Math. Progamm.*, 2019.

[159] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. Fast ADMM for sum-of-squares programs using partial orthogonality. *IEEE Trans. Autom. Control*, 64(9):3869–3876, 2019.

[160] M. Danilova, P. Dvurechensky, A. Gasnikov, E. Gorbunov, S. Guminov, D. Kamzolov, and I. Shibaev. *Recent Theoretical Advances in Non-Convex Optimization*, pages 79–163. Springer Int., 2022.

[161] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Foundations and Trends, 2011.

[162] M. Hong and Z. Luo. On the linear convergence of the alternating direction method of multipliers. *Math. Programm.*, 162:165–199, 2017.

[163] M. Hong, Z. Q. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM J. Optim.*, 26(1):337–364, 2016.

[164] Y. Wang, W. Yin, and J. Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *J. Sci. Comput.*, 78(1):29–63, jan 2019.

[165] M. Mevissen and M. Kojima. SDP relaxations for quadratic optimization problems derived from polynomial optimization problems. *Asia Pac. J. Oper. Res.*, 27:15–38, 02 2010.

[166] S. Elloumi, A. Lambert, and A. Lazare. Semidefinite programming relaxations through quadratic reformulation for box-constrained polynomial optimization problems. In *Proc. Int. Conf. Control Decis. Inf. Technol. (CoDIT)*, pages 1498–1503. IEEE, 2019.

[167] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[168] G. Li and T. K. Pong. Global convergence of splitting methods for nonconvex composite optimization. *SIAM J. Optim.*, 25(4):2434–2460, 2015.

[169] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Math. Oper. Res.*, 35(2):438–457, 2010.