University of Pavia

Department of Electrical, Computer
and Biomedical Engineering

Ph.D. Dissertation

Neural Networks and Sliding Modes: Control, Observation, and Application to Robotics

Advisor:
Prof. Antonella Ferrara

Candidate:
Nikolas Sacchi

A.Y. 2023/2024
CYCLE XXXVII

Abstract

The aim of this dissertation is to present and analyze several control and observation schemes that rely on the joint use of neural network and sliding modes. In particular, it presents a novel framework which exploits deep neural networks (DNN)s and integral sliding mode (ISM) to design control schemes able to control perturbed nonlinear systems with fully unknown dynamics. Differently from other methodologies present in the literature, the DNNs are not trained offline, but, inspired by the adaptive control framework, their weights are adjusted online while the system is being controlled via adaptation laws that are derived from Lyapunov stability analysis. Such a framework is then extended to the case in which the system must satisfy some state or input constraints, presenting three modifications: one which relies on a modified sliding variable, one that embeds model predictive control, and one that makes use of barrier functions.

The joint use of DNNs and sliding modes is also explored in the domain of fault diagnosis (FD). In particular, two FD schemes are presented. The former relies on the aforementioned DNN based ISM framework to build an unknown input observer (UIO) for the estimation of fault affecting a system. As for the latter, it consists of a deep reinforcement learning (DRL) agent that aims to estimate the sensor fault affecting the joints of a robotic manipulator. Such an estimate is used to clear the faulted signal and build a battery of second order sliding mode UIOs that allows to estimate the actuator fault acting on the robot joints.

Finally, application of DNN and sliding modes in the domain of physical human-robot interaction are presented. In particular, an adaptive version of the DNN based ISM control framework is developed to control a robotic manipulator so that it performs the so-called ergonomic handover, i.e., exchanges object with the human operator, adapting to her/his pose to reduce psychophysical stress. Moreover, a collision avoidance architecture that relies on convolutional neural networks for obstacle detection and ISM for obstacle avoidance is presented.

The control and observation methodologies present in this dissertation have been theoretically analyzed and their validity is assessed in simulation or experimentally obtaining more than satisfactorily results. The experiments are performed on a real Franka Emika Panda robot, present in the Intelligent Robotics Lab at the University of Pavia.

List of Acronyms

The list of abbreviations used in this dissertation is hereafter reported.

ABA Articulated Body Algorithm

ANN Artificial Neural Network

APF Artificial Potential Field

ASMC Adaptive Sliding Mode Control

BLF Barrier Lyapunov Function

CBF Control Barrier Function

CDA Convolutional Denoising Autoencoder

CLF Control Lyapunov Function

CNN Convolutional Neural Network

DDPG Deep Deterministic Policy Gradient

DH Denavit-Hartenberg

DoF Degrees of Freedom

DNN Deep Neural Network

DNN-ISM Deep Neural Network based Integral Sliding Mode

DRL Deep Reinforcement Learning

EL Euler-Lagrange

FD Fault Diagnosis

FE Fault Event

FHOCP Finite-Horizon Optimal Control Problem

HE Hand-Eye

HOSM Higher Order Sliding Mode

ICP Iterative Closest Point

IMU Inertial Measurement Unit

ISM Integral Sliding Mode

MARL Multi-Agent Reinforcement Learning

MDP Markov Decision Process

MIMO Multi Input Multi Output

ML Machine Learning

MLP Multi-Layer Perceptron

MPC Model Predictive Control

MSDs Muscoloskeletal Disorders

NN Neural Network

pHRI physical Human Robot Interaction

QP Quadratic Programming

RGBD Red Green Blue Depth

RL Reinforcement Learning

RMS Root Mean Square

RNEA Recursive Newton-Euler Algorithm

ROS Robot Operating System

SISO Single Input Single Output

SMC Sliding Mode Control

SOSM Second Order Sliding Mode

SPD Symmetric Positive Definite

TD3 Twin-Delayed DDPG

UIO Unknown Input Observer

URDF Unified Robot Description Format

VSC Variable Structure Control

VSS Variable Structure System

Notation

The sets $\mathbb{R}_{>0}$ and $\mathbb{R}_{\geq 0}$ represent positive and non-negative real numbers, respectively. Given a matrix $A \in \mathbb{R}^{n \times m}$, then $A^\top \in \mathbb{R}^{m \times n}$ denotes its transpose, $A^{(i)} \in \mathbb{R}^n$ its i-th column, while $\mathrm{vec}\,(A) \in \mathbb{R}^{nm}$ is the vectorization operation, defined as $\mathrm{vec}\,(A) = \begin{bmatrix}(A^{(1)})^\top & (A^{(2)})^\top & \cdots & (A^{(m)})^\top\end{bmatrix}^\top$. Given a vector $x \in \mathbb{R}^n$ and a matrix $B \in \mathbb{R}^{n \times n}$, then $\|x\|_B^2 = x^\top B x$. An identity matrix with n rows and columns is denoted as $I_n \in \mathbb{R}^{n \times n}$, a matrix of zeros with p rows and q columns as $0_{p \times q} \in \mathbb{R}^{p \times q}$, while a vector of m zeros as $0_m \in \mathbb{R}^m$. Given a time varying vector $v(t) \in \mathbb{R}^n$, $v_{[t_a, t_{a+N}]}$ denotes the discrete time samples of the vector, i.e, $[v(t_a), v(t_{a+1}), \ldots, v(t_{a+N})]$. Note that, throughout the dissertation, function dependencies may be omitted for sake of readability when obvious.

Contents

Contents ix

List of Figures xv

List of Tables xxi

I Introduction and Preliminaries 1

1 Introduction 3

1.1 Thesis Structure 5

1.2 List of peer-reviewed scientific publications 7

2 Preliminaries on Sliding Mode Control 9

2.1 Idea behind Sliding Mode Control 9

2.1.1 An illustrative example 10

2.2 Control-Affine Systems 13

2.2.1 Canonical forms 13

2.3 Elements of classical Sliding Mode Control 15

2.3.1 The Sliding Manifold 15

2.3.2 The Control Law 17

2.3.3 Existence and Reaching conditions 18

2.3.4 Solutions of the controlled system 21

2.3.5 Robustness Property and SMC Design 23

2.3.6 The Chattering problem 25

2.3.7 Approximability Property 26

2.4 Integral Sliding Mode 27

2.4.1 Existence and Robustness Property 28

2.4.2 Physical Interpretation of the Equivalent Control 31

2.4.3 ISM control example 31
2.5 Higher Order Sliding Mode Control 32
2.6 Adaptive Sliding Mode Control 35

3 Preliminaries on Neural Networks and Learning 37
3.1 Multi-Layer Perceptron 37
3.1.1 The Perceptron model 38
3.1.2 Universal approximation capabilities of ANNs 41
3.1.3 Depth vs Width 43
3.1.4 Learning the weights 44
3.2 Reinforcement Learning 46
3.2.1 Key concepts 47
3.2.2 Q-learning 50
3.2.3 Deep Q-learning 51
3.2.4 Actor-Critic 53

4 Preliminaries on Robotics 59
4.1 Basic Definitions 59
4.2 Pose of a rigid body 60
4.2.1 Change of orientation representation 62
4.3 Kinematics modeling 62
4.3.1 Forward Kinematics 63
4.3.2 Differential Kinematics 65
4.3.3 Inverse Kinematics 67
4.4 Dynamic Modeling 68

II Deep Neural Network based Integral Sliding Mode Control Framework 71

5 The DNN-ISM Framework 73
5.1 Problem Formulation 74
5.2 Approximating the Dynamics using DNNs 76
5.2.1 Approximation error of the Drift Dynamics DNN 79
5.2.2 Approximation error of the Control Effectiveness DNN 84
5.3 The DNN-ISM Control Strategy 86
5.3.1 Use of Parameter Projection and its effects 90
5.3.2 Sliding mode Existence 93

5.4 Practical Aspects . . . 96
5.4.1 Computational Complexity Analysis . . . 96
5.4.2 Chattering Reduction . . . 100
5.4.3 Weights initialization . . . 101
5.5 Simulations . . . 102
5.5.1 Duffing Oscillator . . . 102
5.5.2 Robotic Manipulator . . . 104
5.6 Real Robot Experiment . . . 105

6 DNN-ISM with State and Input Constraints 111
6.1 DNN-ISM with State Constraints Avoidance . . . 111
6.1.1 Problem Formulation . . . 111
6.1.2 The DNN-ISM scheme with avoidance capabilities . . . 113
6.1.3 Sliding mode existence . . . 115
6.1.4 Simulations . . . 117
6.2 DNN based MPC/ISM . . . 118
6.2.1 Problem Formulation . . . 119
6.2.2 The DNN-ISM based MPC scheme . . . 121
6.2.3 Simulations . . . 123
6.3 DNN-ISM with Barrier Functions . . . 124
6.3.1 Problem Formulation . . . 124
6.3.2 Preliminaries on BLFs, CLFs, and CBFs . . . 126
6.3.3 The DNN-ISM scheme with CBFs, CLFs, and BLFs . . . 128
6.3.4 Simulations . . . 131

III Fault Diagnosis via Neural Networks and Sliding Mode Observers 135

7 Fault Diagnosis via DNN-ISM based UIO 137
7.1 The considered faulted system . . . 138
7.2 ISM Unknown Input Observer . . . 138
7.3 DNN-ISM Unknown Input Observer . . . 139
7.4 Simulations . . . 142

8 SM based Fault Diagnosis with DRL add-ons for Redundant Manipulators 145
8.1 Problem Formulation . . . 146

8.1.1 Robot Model . . . 146
8.1.2 Faults Modeling . . . 146
8.1.3 Problem Statement . . . 147
8.2 Inverse Dynamics control . . . 147
8.3 The Fault Diagnosis Scheme . . . 149
8.3.1 Sensor fault diagnosis with DRL . . . 149
8.3.2 Actuator FD with SOSM UIOs . . . 151
8.4 Simulations . . . 153

IV Applications to Human-Robot Interaction 159

9 Human-Robot Ergonomic Handover via Adaptive DNN-ISM 161
9.1 Problem Formulation . . . 162
9.2 Adaptive DNN-ISM for ergonomic handover . . . 163
9.2.1 Reference generation . . . 164
9.2.2 Adaptive DNN-ISM control . . . 166
9.3 Experiment . . . 168

10 Vision-based Collision Avoidance with ISM control for Collaborative Robots 171
10.1 Problem Formulation . . . 171
10.2 The HE calibration scheme . . . 172
10.2.1 Position estimation . . . 172
10.2.2 Orientation estimation . . . 174
10.2.3 Pose estimation adjustment . . . 176
10.3 The collision avoidance scheme . . . 177
10.4 Experiment and results . . . 179

11 Conclusions and Future Research 183
11.1 Future Research . . . 184

A Parameter Projection 189

B Proofs 193
B.1 Proof of Theorem 5.1 . . . 193
B.2 Proof of Theorem 5.2 . . . 198
B.3 Proof of Theorem 6.1 . . . 200

B.4 Proof of Theorem 6.2 206
B.5 Proof of Theorem 6.3 208
B.6 Proof of Theorem 6.4 210
B.7 Proof of Theorem 7.1 211
B.8 Proof of Theorem 7.2 212
B.9 Proof of Theorem 7.3 213
B.10 Proof of Theorem 8.1 215
B.11 Proof of Theorem 9.1 215

C Franka Emika Panda Robot **219**
C.1 Technical Specifications 220
C.2 Dynamical Modeling 221
C.3 PyBullet Simulation 222
C.4 Controlling the Panda robot 223

Bibliography **225**

List of Figures

2.1 Illustrative example of the sliding manifold $\sigma = 0_2$ given by the intersection of $\kappa = 2$ switching surfaces 10

2.2 Graphical representation of the Duffing oscillator system. 11

2.3 Graphical interpretation of the Filippov's solution. 23

2.4 Example of an ideal sliding mode (left) and a sliding mode with chattering (right). 26

2.5 Time evolution of the system states 32

2.6 Time evolution of the sliding variable 32

2.7 Time evolution of equivalent control 32

3.1 Graphical representation of a biological neuron and its main components. 38

3.2 Mathematical model of the neuron. 39

3.3 Example of a network with one hidden layer. 39

3.4 Example of a network with $k = 2$ hidden layers that approximates a 2×2 matrix. 43

3.5 Graphical representation of the reinforcement learning framework. . 47

3.6 Block diagram of a generic actor-critic algorithm. 54

4.1 Example of open-chain manipulator with seven revolute joints, eight links, and a gripper as end-effector. 60

4.2 Graphical representation of the Denavit-Hartenberg convention, with the parameters highlighted in red. 64

5.1 Block diagram of the DNN-ISM control scheme. The blocks associated with the DNNs, the sliding variable, and the control law are highlighted in green, blue, and yellow, respectively. 87

5.2 Outlier box plot of the recorded times of the full updates of the neural networks. The box indicates the data samples between the 25-th and the 75-th percentile, with the vertical red line indicating the median of the sample distribution. The whiskers of the box plot go from the 5-th to the 25-th and from the 75-th to the 95-th percentile, while the red markers indicate the outliers. 100

5.3 Time evolution of the system states, sliding variable components, and discontinuous control gain during the simulation. 103

5.4 Time behavior of the sliding variable when the DNN-ISM is employed with the discontinuous control gain depicted in Figure 5.3, and when SMC is applied with constant gains $\rho = 0.15$, $\rho = 0.25$, $\rho = 0.5$, and $\rho = 0.75$. 104

5.5 Evolution of the joint positions (left column), joint velocities (middle column), and sliding mode components (right column) during the simulation. 106

5.6 Evolution of the nominal control u_n (left column), robustifying control law u_r (middle column), and full control law u (right column) during the simulation. 107

5.7 Evolution of the discontinuous control gain ρ during the simulation. 108

5.8 Evolution of the joint positions (left column), joint velocities (middle column), and sliding mode components (right column) during the experiment. 109

5.9 Evolution of the nominal control u_n (left column), robustifying control law u_r (middle column), and full control law u (right column) during the experiment. 110

6.1 Block diagram of the modified DNN-ISM control scheme for the avoidance of non-admissible states. The blocks related to the DNNs are colored in green, the blue ones are associated with the sliding variable, the yellow blocks are the components of the control law, while in red are denoted the blocks responsible for the change of control mode. 114

6.2 Time evolution of the joint positions, sliding variable components and control input when the planar version of the robot is controlled using the control scheme in Figure 6.1. 118

6.3 Block diagram of the MPC/ISM control architecture. 119

6.4 Block diagram of the DNN based MPC/ISM control architecture. The blocks associated with the control law are colored in yellow, the ones related to the sliding variable in blue, and the ones corresponding to the DNNs and their update in green. 121

6.5 Time evolution of the system states, control input, and sliding variable components during the simulation. The state and input constraints $\mathcal{X}$ and $\bar{\mathcal{U}}$ are depicted with green solid lines. 125

6.6 Zoom of the norm of the sliding variable during the adaptation transient, along the MPC activation threshold ς 125

6.7 Block diagram of the BLF based DNN-ISM control architecture with CBF/CLF based QP. The yellow blocks are related to the control law, the green ones to the DNNs and their adaptation, while the blue ones are associated with the sliding variable. 128

6.8 Graphical representation of the double tank system employed in the simulation. 131

6.9 Time evolution of the system states (first and second plots), norm of the full control input (third plot), and norm of the nominal control input resulting from the QP problem (6.58)(fourth plot). 133

6.10 Time evolution of the sliding variable components (first and second plots) and norm of the sliding variable vector, along with the bound imposed by the BLF (third plot). 134

7.1 Block diagram of the DNN-ISM based UIO. The blocks associated with the sliding variable are colored in blue, the ones related to the DNNs in green, while the ones in yellow are related to the UIO and its input. 140

7.2 Time evolution of the system and UIO states, the sliding variable components, the actual fault injected into the system, and its estimate. 143

8.1 Inverse dynamic control architecture. 148

8.2 Block diagram of the FD scheme. The blocks in yellow are related to the inverse dynamics, the ones in blue are associated with the estimation and compensation of the sensor faults, while the blocks responsible for the actuator faults estimation are colored in red. . . . 149

8.3 Evolution of the average cumulative reward during the training procedure. 154

8.4 Time evolution of the actual and estimated actuator faults for joints 1, 3, 4 and 6 in the case of scenario FE3. 155
8.5 Time evolution of the actual and estimated sensor faults for joints 2 and 6 in the case of scenario FE4. 156
8.6 Time evolution of the actual and estimated sensor and actuator faults for joints 1 and 7 in the case of scenario FE5. 157

9.1 Block diagram of the hand reaching strategy. The blocks responsible for the computation of the joints reference are colored in yellow. . . 164
9.2 Block diagram of the adaptive DNN-ISM control scheme. The yellow blocks are associated with the control law, the blue and the green ones are related to the sliding variable and the DNN, respectively. . 166
9.3 Instants in which the handover operation is performed. The robot adapts the pose of its end-effector so that it follows the orientation of the IMU sensor. 169
9.4 Time evolution of the pose error (first plot), norm of the sliding variable (second plot), and adaptive control gain (third plot). 170

10.1 Block diagram of the HE calibration scheme. The green blocks are associated with the elements that rely on CNNs, blocks responsible for the computation of the first guess are highlighted in yellow. . . . 173
10.2 Examples of the generated synthetic images with the bounding boxes for training the YOLOv3 detector. 174
10.3 Architecture of the CDA employed for orientation estimation. 174
10.4 Examples of two synthetic images, along with their ground truths, for the training of the CDA. 175
10.5 Examples of images in which (a) the robot presents axial symmetry (b) the robot configuration breaks axial symmetry. 176
10.6 Block diagram of the obstacle detection and avoidance scheme. The blocks responsible for the change of reference frame are highlighted in red, the ones that compute the joint position reference in yellow, and the low-level ISM controller in blue. 177
10.7 Vision sensor employed for the experiment. 179
10.8 Result of the HE calibration process. Depth points from the camera are in blue and green. The orange 3D model represents the first pose estimate, while the red points the result of the ICP algorithm. . . . 180

10.9 Instants of the experiment. In each frame, the point cloud $\mathcal{O}^c$ are colored in red and presented in the bottom left corner. 181
10.10 Time evolution of the distance between end-effector and target (blue), robot an human operator (red), and the state flag. 181

A.1 Graphical representation of the projection operators defined in (A.2) and (A.4) in the case $\theta \in \mathbb{R}^2$. 190

C.1 Franka Emika Panda robot present in the University of Pavia. . . . 219
C.2 Workspace of the Franka Emika Panda robot. 220
C.3 Graphical representation of the modified Denavit-Hartenberg (DH) parameters of the Franka Emika Panda robot (picture taken from the documentation). 221
C.4 Virtualization of the Franka Emika Panda robot in PyBullet. 223
C.5 Connection diagram of the Franka Emika Panda robot, the low-level controller, and the computer with Libfranka library. 224

List of Tables

3.1 Tabular representation of the optimal action value function in the case of discrete state and action spaces $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ and $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$. 51

8.1 Possible fault events (FEs) during the task execution. 147
8.2 RMS estimation errors for scenario FE3. 155
8.3 RMS estimation errors for scenario FE4. 156
8.4 RMS estimation errors for scenario FE5. 156

10.1 Example of the lookup table which associates the latent space vector z to a set of Euler angles. 175

C.1 Limits of the robot joints. 220
C.2 Modified DH parameters of the Franka Emika Panda robot. 222

Part I

Introduction and Preliminaries

Chapter 1

Introduction

In real world control applications, one has to deal with the problem of uncertainty. Sources of uncertainty may be, for example, exogenous signals (e.g., faults, measurement noise, etc.), or modeling mismatches. To cope with uncertainties, the literature of control theory has been enriched with a large variety of methodologies, which belong to the so-called Robust Control field [1, 2]. Usually, such techniques generate the control action while taking into consideration the worst realization of the uncertainty. Depending on the specific methodology employed, this could result into an-over conservative strategy, or into the generation of a control signal which could be detrimental for the plant.

However, there are some cases in which the system model is completely unavailable to the controller designer and, in such a condition, devising control laws with sufficient guarantees is not possible.

In the last decades, availability of data and the computational power have grown exponentially. For this reason, Machine Learning (ML), and in particular the concept of Artificial Neural Network (ANN), gained popularity in a wide variety of fields, included control theory. Even though the popularity of ML has increased quite recently, its origin can be traced back to the beginning of the XIX century, when the *least squares problem* has been formulated. Nevertheless, the history of the ML field, as it is intended today, starts more or less in the half of the last century, when the so-called *Perceptron* model has been introduced in [3]. The popularity of ANNs comes from the fact that they are extremely convenient, since they require a restricted number of hypotheses, and powerful, since they provide important theoretical results. Above all of them, there is the so-called *Universal Approximation Theorem*, proposed at the end of the eighties by George Cybenko in [4], which proves that, in general, an ANN is a function capable of approximating

any continuous function with a degree of approximation which is related to the ANN structure. Indeed, in works like [5] and [6], it has been shown how an ANN with a deeper architecture, referred to as Deep Neural Network (DNN), provides better results than a network with a single layer of perceptrons.

Clearly, also the field of control theory has been influenced by ML, and specifically by ANNs. Indeed, the universal approximation capabilities of this last one become a powerful tool that enables the design of control laws even in the case of unknown system dynamics. The ways in which ANNs have been employed are different. For example, [7] proposes a methodology that performs the so-called *identification* of the system dynamics relying on ANNs. Moreover, in the case in which a very large amount of data is available, one could train a network that estimates the plant dynamics and treat it as an observer, as done in [8]. Another possibility is to employ ANNs to directly estimate the optimal control law [9].

The main issue in many works involving ANNs is that the training phase is done offline and the approximation error, which is dependent by the network structure and by the quality of the gathered data, is not properly dealt with, preventing from an effective control design. Moreover, the fact that the ANNs are trained offline does not give any guarantees about the effects that the approximation has on the behavior of the system.

The aim of this dissertation is to explore the combined use of ANNs and robust control strategies, specifically Sliding Mode Control (SMC) [10], for control and observer design with performance guarantees.

Throughout the years, the use of neural networks and sliding mode control has been widely investigated. In particular, in works like [11, 12, 13] Radial Basis Function neural networks are combined with sliding mode control for designing controllers and observers, with applications to robotic manipulators, power converters, and lithium-ion batteries. In other works [14, 15, 16], neural networks with fuzzy logic have been employed to design controllers and observers for controlling and observing motor drives and active suspension systems. Other interesting works that analyze the combination of sliding mode controllers and neural networks are, among others, [17, 18, 19, 20].

The existing literature presents two main flaws. The first is that the neural networks are not used to estimate the complete model of the plant, but rather to compensate small model mismatches or directly the disturbance. Second, the majority of them present an offline training of the neural networks, which can lead to the problems described earlier in this introduction.

With the aim of coping with these aspects and enrich the existing literature, this dissertation covers the following topics:

- design of a novel DNN based Integral Sliding Mode (ISM) control framework, characterized by an online adaptation of the parameters, for controlling perturbed nonlinear system with fully unknown dynamics;
- extensions of the aforementioned control framework to the case in which the system is affected by state and input constraints;
- development of fault detection schemes that rely on the joint use of DNN and SMC observers, with application to industrial manipulators;
- joint use of DNNs and SMC for the development of control strategies for safe and ergonomic physical-human robot interaction, with experiments on a real collaborative robot.

1.1 Thesis Structure

This dissertation is divided into four parts and structured as follows:

(I) **Introduction and Preliminaries**: In this part, all the preliminary concepts behind what presented in this dissertation, are introduced. In particular, Chapter 2 introduces the basics about SMC, along with the main theorems, Chapter 3 presents some preliminaries on neural networks and introduces the notation that will be employed in the rest of the dissertation, while Chapter 4 recalls some fundamental concepts about robotics.

(II) **Deep Neural Network based Integral Sliding Mode Control Framework**: In this part, the Deep Neural DNN-ISM control framework, which can be seen as the core of this dissertation, is introduced. In particular, Chapter 5 presents the main concepts and theoretical results and it is based on the published works

- N. Sacchi, G.P. Incremona, and A. Ferrara. “Neural network-based practical/ideal integral sliding mode control.” IEEE Control Systems Letters 6 (2022): 3140-3145.
- E. Vacchini, N. Sacchi, G.P. Incremona, and A. Ferrara. “Design of a deep neural network-based integral sliding mode control for nonlinear systems under fully unknown dynamics.” IEEE Control Systems Letters 7 (2023): 1789-1794.

- N. Sacchi, E. Vacchini, G.P. Incremona, and A. Ferrara. "On neural networks application in integral sliding mode control." Journal of the Franklin Institute, Volume 361, Issue 13 (2024).

As for Chapter 6, it presents extensions of the aforementioned framework to the case in which there are state and input constraints, and it is based on the results published in

- N. Sacchi, E. Vacchini, A. Ferrara. "Neural network based integral sliding mode control of systems with time-varying state constraints." 2023 31st Mediterranean Conference on Control and Automation (MED). IEEE, 2023.
- N. Sacchi, E. Vacchini, G.P. Incremona, and A. Ferrara. "Model Predictive Control with Deep Neural Network Based Integral Sliding Modes Generation for a Class of Uncertain Nonlinear Systems." IFAC-PapersOnLine 58.5 (2024): 84-89.

(III) **Fault Diagnosis via Neural Networks and Sliding Mode Observers** In this part, two different fault detection schemes, which rely on the use of neural networks and Sliding Mode Control, are presented. Specifically, Chapter 7 presents an Unknown Input Observer developed on the Deep Neural Network based Integral Sliding Mode (DNN-ISM) framework introduced in Chapter 5 for the detection of faults acting on the input channel, and it is based on the work in

- N. Sacchi, G.P. Incremona, and A. Ferrara. "Actuator Fault Diagnosis With Neural Network-Integral Sliding Mode Based Unknown Input Observers." IFAC-PapersOnLine 56.2 (2023): 773-778.

Moreover, Chapter 8 presents the combination of a Deep Reinforcement Learning agent and a Second Order Sliding Mode Unknown Input Observer for the fault diagnosis of sensors and actuator faults acting on an industrial manipulator. This Chapter is based on the results contained in

- N. Sacchi, G.P. Incremona, and A. Ferrara. "Sliding mode based fault diagnosis with deep reinforcement learning add-ons for intrinsically redundant manipulators." International Journal of Robust and Nonlinear Control 33.15 (2023): 9109-9127.

(IV) **Applications to Human-Robot Interaction** In this part, neural networks and SMC are employed to develop architectures for safe and ergonomic human robot interaction. In particular, Chapter 9, which presents a strategy that exploits a modified version of the DNN-ISM framework to design a controller to perform ergonomic handover between a robot and a human operator, is based on the work in

- N. Sacchi, E. Vacchini, and A. Ferrara. "Human-Robot Ergonomic Handover via Deep Neural Network Based Adaptive Integral Sliding Mode Control." 2024 European Control Conference (ECC). IEEE, 2024.

As for Chapter 10, it presents a collision avoidance scheme which exploits a vision-based methodology for obstacle detection and uses Integral Sliding Mode control for maintaining a safety distance between the robot and the human operator.

Finally, some concluding remarks are gathered and possible future research paths are proposed in Chapter 11.

1.2 List of peer-reviewed scientific publications

In the following, the complete list of peer-reviewed publications produced during the Ph.D. course (October 2021 - September 2024) is reported.

Journal

- **N. Sacchi**, E. Vacchini, G. P. Incremona, and A. Ferrara. "On neural networks application in integral sliding mode control." Journal of the Franklin Institute, Volume 361, Issue 13 (2024).
- E. Vacchini, **N. Sacchi**, G.P. Incremona, and A. Ferrara. "Design of a deep neural network-based integral sliding mode control for nonlinear systems under fully unknown dynamics." IEEE Control Systems Letters 7 (2023): 1789-1794.
- **N. Sacchi**, G.P. Incremona, and A. Ferrara. "Neural network-based practical/ideal integral sliding mode control." IEEE Control Systems Letters 6 (2022): 3140-3145.
- **N. Sacchi**, G.P. Incremona, and A. Ferrara. "Sliding mode based fault diagnosis with deep reinforcement learning add-ons for intrinsically redundant

manipulators." International Journal of Robust and Nonlinear Control 33.15 (2023): 9109-9127.

Conference Proceeding

- A. Ferrara, G.P. Incremona, E. Vacchini and **N. Sacchi**. "Design of Neural Networks Based Sliding Mode Control and Observation: An Overview." 17th International Workshop on Variable Structure Systems (VSS). IEEE, 2024.
- E. Vacchini, **N. Sacchi**, M. Cucuzzella, and A. Ferrara. "Robust Sliding Manifold Design for Uncertain Linear Systems." 2024 European Control Conference (ECC). IEEE, 2024.
- **N. Sacchi**, E. Vacchini, and A. Ferrara. "Human-Robot Ergonomic Handover via Deep Neural Network Based Adaptive Integral Sliding Mode Control." 2024 European Control Conference (ECC). IEEE, 2024.
- **N. Sacchi**, E. Vacchini, G.P. Incremona, and A. Ferrara. "Model Predictive Control with Deep Neural Network Based Integral Sliding Modes Generation for a Class of Uncertain Nonlinear Systems." IFAC-PapersOnLine 58.5 (2024): 84-89.
- **N. Sacchi**, E. Vacchini, A. Ferrara. "Neural network based integral sliding mode control of systems with time-varying state constraints." 2023 31st Mediterranean Conference on Control and Automation (MED). IEEE, 2023.
- **N. Sacchi**, G.P. Incremona, and A. Ferrara. "Integral Sliding Modes Generation via DRL-Assisted Lyapunov-Based Control for Robot Manipulators." 2023 European Control Conference (ECC). IEEE, 2023.
- **N. Sacchi**, G.P. Incremona, and A. Ferrara. "Actuator Fault Diagnosis With Neural Network-Integral Sliding Mode Based Unknown Input Observers." IFAC-PapersOnLine 56.2 (2023): 773-778.

Chapter 2

Preliminaries on Sliding Mode Control

The aim of this chapter is to introduce the main concepts related to Sliding Mode Control, instrumental for the understanding of the DNN based approach presented later in Chapter 5.

2.1 Idea behind Sliding Mode Control

Sliding Mode Control is a nonlinear control technique belonging to the family of Variable Structure Control (VSC), which alters the dynamics of a system by means of a high frequency switching law [10, 21].

Consider a system characterized by a state vector $x \in \mathcal{X} \subset \mathbb{R}^n$, with $\mathcal{X}$ being a compact set containing the origin. Then, let $\sigma_i(x) = 0$, with $i \in \{1, 2, \dots, \kappa\}$, be some suitably predefined surfaces, each defined as subspace of the state space. The intersection of such surfaces is called *sliding manifold* (an illustrative example is provided in Figure 2.1). If the system state forced on $\sigma_i(x) = 0$, then a *sliding motion* is generated on that surface and this last one is referred to as *switching surface*. The idea behind SMC is to design a controller which, starting from an initial condition $x(t_0) = x_0$, with $t_0 \in \mathbb{R}_{\geq 0}$ being the initial time instant, steers the state onto the sliding manifold in finite time, generating a sliding motion on all the surfaces, and hence on the sliding manifold. If this happens, then the system is said to be in *sliding mode*. When a sliding mode is enforced, the controlled system behaves like a system of reduced order, referred to as *equivalent system*, and whose dynamics depends on the definition of the sliding manifold. Moreover, the system becomes insensitive with respect to a significant class of parameter uncertainties

and disturbances.

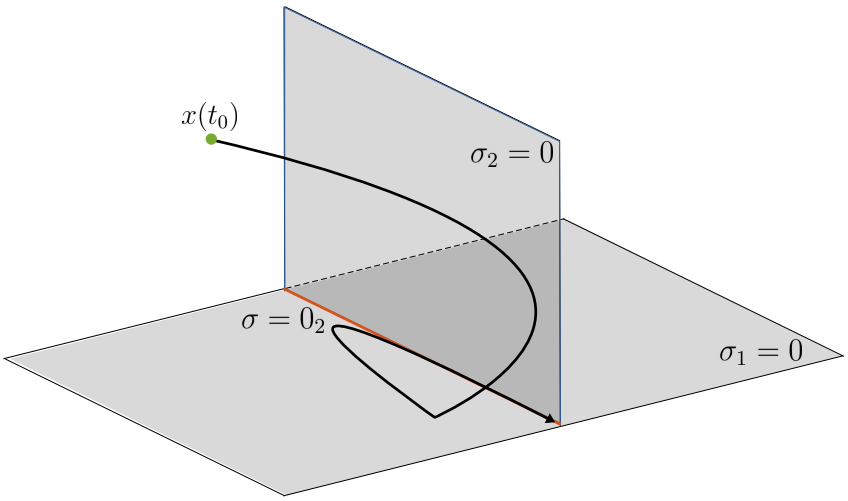


Figure 2.1: Illustrative example of the sliding manifold $\sigma = 0_2$ given by the intersection of $\kappa = 2$ switching surfaces

2.1.1 An illustrative example

In order to present some of the main concepts related to SMC, a simple example is introduced. In particular, let us consider the *Duffing oscillator*, depicted in Figure, 2.2 and whose dynamics is

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{g}{l}\sin(x_1) - \frac{\alpha}{ml^2}x_1 - \frac{k}{ml^2}x_2 + \frac{\tau}{ml^2} \end{bmatrix}, \tag{2.1}$$

where $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top \in \mathcal{X} \subset \mathbb{R}^2$ is the state vector, with x_1 and x_2 being the position and velocity, respectively. The other elements appearing in (2.1) are the gravity term $g \in \mathbb{R}$, the mass attached to the rod $m \in \mathbb{R}$, the length of the rod $l \in \mathbb{R}$, and the control torque $\tau \in \mathbb{R}$. As for $\alpha \in \mathbb{R}_{>0}$ and $k \in \mathbb{R}_{>0}$, they are respectively, the parameters regulating the restoring torque $-\alpha x_1$ and the damping torque $-kx_2$. Defining $u = \frac{\tau}{ml^2}$ and approximating $\sin(x_1) \approx x_1 - \frac{x_1^3}{6}$, the dynamics in (2.1) can be rewritten as

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \left(\frac{g}{l} - \frac{\alpha}{ml^2}\right)x_1 - \frac{g}{6l}x_1^3 - \frac{k}{ml^2}x_2 + u \end{bmatrix}. \tag{2.2}$$

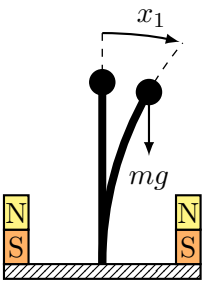


Figure 2.2: Graphical representation of the Duffing oscillator system.

For sake of simplicity in the analysis, it is possible to set $\frac{g}{l} - \frac{\alpha}{ml^2} = 1$, $\frac{g}{6l} = 1$, and $\frac{k}{ml^2} = \eta$ to simplify the above dynamics as

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1(1 - x_1^2) - \eta x_2 + u \end{bmatrix}. \tag{2.3}$$

The design of SMC consists in two phases, i.e., *manifold design* and *control law design*. The sliding manifold is designed so that, while in sliding mode, the dynamics of the controlled system exhibits some desired properties (e.g., asymptotic stability). As for the control law, it is chosen so that a sliding mode is enforced.

With the aim of steering the system state toward the origin, the sliding manifold can be selected as a linear combination of the state elements, i.e.,

$$\sigma(x) = cx_1 + x_2 = 0, \tag{2.4}$$

where $\sigma : \mathcal{X} \to \mathbb{R}$ is referred to as *sliding variable*.

To bring the systems states on the sliding manifold, the control law

$$u = -\rho \operatorname{sign}(\sigma), \tag{2.5}$$

can be used. In particular, the term $\rho \in \mathbb{R}_{>0}$ is the control gain, while the $\operatorname{sign}(\cdot)$ function is defined as

$$\operatorname{sign}(\sigma) = \begin{cases} 1 & \text{if } \sigma > 0 \\ -1 & \text{if } \sigma < 0 \end{cases}$$

To prove that the control law (2.5) successfully steers the system state on the sliding manifold $\sigma = 0$ and to properly select the control gain ρ, stability analysis must be performed [22]. Let $v : \mathcal{X} \to \mathbb{R}$ be a candidate Lyapunov function defined as

$$v = \frac{1}{2}\sigma^2. \tag{2.6}$$

Then, asymptotic stability of the equilibrium point $\sigma = 0$ is ensured if, for $\sigma \neq 0$, it holds that $\dot{v} < 0$. Moreover, finite-time convergence is achieved if it holds

$$\dot{v} \leq -\gamma\sqrt{v} = -\frac{\gamma}{\sqrt{2}}|\sigma|, \tag{2.7}$$

with $\gamma \in \mathbb{R}_{>0}$. Since $\dot{v} = \sigma\dot{\sigma}$, the above expression can be conveniently rewritten as

$$\sigma\dot{\sigma} \leq -\frac{\gamma}{\sqrt{2}}|\sigma|. \tag{2.8}$$

The left hand-side of the inequality can computed explicitly as

$$\sigma\dot{\sigma} = \sigma(c\,x_2 + x_1 - x_1^3 - \eta x_2 + u). \tag{2.9}$$

Since $x \in \mathcal{X}$, with $\mathcal{X}$ being compact, then it holds that $|c\,x_2 + x_1 - x_1^3 - \eta x_2| \leq \bar{F}$, with $\bar{F} \in \mathbb{R}_{>0}$. Then, substituting u as in (2.5), the above expression can be upper bounded as

$$\begin{aligned}\sigma\dot{\sigma} &\leq \sigma(\bar{F} - \rho\,\mathrm{sign}\,(\sigma)) \\ &\leq (\bar{F} - \rho)|\sigma|.\end{aligned} \tag{2.10}$$

Then, to satisfy (2.8), which is referred to as γ-*reaching condition*, it is sufficient to choose the control gain ρ so that the condition

$$(\bar{F} - \rho)|\sigma| = -\frac{\gamma}{\sqrt{2}}|\sigma|$$

is satisfied. Hence, finite-time convergence to $\sigma = 0$ is ensured if

$$\rho = \bar{F} + \frac{\gamma}{\sqrt{2}}. \tag{2.11}$$

By virtue of to the choice of the sliding variable (2.4), when at time $t_r > t_0$ the sliding mode $\sigma = 0$ is enforced, it holds that $x_2 = -c\,x_1$ and the dynamics of the controlled system has a reduced order, i.e.,

$$\begin{bmatrix}\dot{x}_1(t) \\ x_2(t)\end{bmatrix} = -c\,x_1(t)\begin{bmatrix}1 \\ 1\end{bmatrix}, \tag{2.12}$$

whose solution is

$$\begin{bmatrix}\dot{x}_1(t) \\ x_2(t)\end{bmatrix} = e^{-c(t-t_r)}\begin{bmatrix}x_1(t_r) \\ -c\,x_1(t_r)\end{bmatrix}. \tag{2.13}$$

From the above equation, it is evident how the properties of the system is sliding mode are strictly dependent on the choice of the sliding manifold, in the form of the parameter c. In particular, asymptotic stability of the origin is ensured if c is positive, with a rate of convergence that is equal to this last one.

2.2 Control-Affine Systems

In classical SMC theory, the class of *control-affine systems* is considered. A control-affine system is a system which is nonlinear with respect to the state and affine with respect to the control variable. Hence, as detailed in [22], it can be described by

$$\dot{x}(t) = f(x(t), t) + B(x(t), t)u(t), \tag{2.14}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the control input vector, $t \in \mathbb{R}_{\geq 0}$ is time, $f : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is the so-called *drift dynamics*, while $B : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n \times m}$ is the so-called *control effectiveness matrix*. In the domain of SMC theory is common to assume that both f and B are smooth and norm bounded vector fields. Moreover, the system state is assumed to be limited in a compact set $\mathcal{X} \subset \mathbb{R}^n$ containing the origin, i.e., $x \in \mathcal{X}$.

The class of control-affine systems is one of the most studied in the literature, as it includes the majority of electromechanical systems (e.g., motors [23] and robotic manipulators [24]).

2.2.1 Canonical forms

The analysis of VSC, and more specifically SMC, is simpler when the considered nonlinear system is expressed in one of the canonical forms [22, 25, 26]. To discuss them, it is fundamental to introduce the concept of *relative degree*.

Consider the nonlinear system (2.14) with associated output function

$$\begin{cases} \dot{x}(t) = f(x(t), t) + B(x(t), t)u(t) \\ y(t) = \psi(x(t), t) \end{cases} \tag{2.15}$$

where $y \in \mathbb{R}^m$ is the output vector, while $\psi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ is a smooth enough vector field. The definition for Single Input Single Output (SISO) and its generalization for Multi Input Multi Output (MIMO) systems are introduced.

Definition 2.1 (Relative degree of SISO systems). *Let the system* (2.15) *be a SISO system, i.e.,* $m = 1$*. Then, the relative degree* $r \in \mathbb{N}_{>0}$ *of the system is the minimum order of the time-derivative of the output, namely* $y^{(r)} = \frac{d^r y}{dt^r}$*, in which the control* u *explicitly appears.*

Definition 2.2 (Relative degree of MIMO systems). *Let the system* (2.15) *be a MIMO system, i.e.,* $m > 1$*. Then, each component* y_i*, with* $i \in \{1, 2, \ldots, m\}$ *must be considered. In particular, define* $r_i \in \mathbb{N}$ *the minimum order of the derivative of the output element, namely,* $y_i^{(r_i)} = \frac{d^{r_i} y_i}{dt^{r_i}}$ *in which any component of the*

control vector explicitly appears. The result is the so-called relative degree vector $\bar{r} = \begin{bmatrix} r_1 & r_2 & \cdots & r_m \end{bmatrix}^\top \in \mathbb{N}^m$ *and its 1-norm defines the total relative degree of the system, i.e.,* $r = \|\bar{r}\|_1 = \sum_{i=1}^m |r_i|$.

It is now possible to introduce the so-called canonical forms.

Reduced form Consider a control-affine system like the one in (2.14). If the control effectiveness matrix has a structure

$$B(x(t), t) = \begin{bmatrix} 0_{(n-m)\times m} \\ \bar{B}(x(t), t) \end{bmatrix}, \tag{2.16}$$

with $\bar{B} : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{m\times m}$, then the state vector can be split into two components, namely $x_1 \in \mathcal{X}_1 \subset \mathbb{R}^{n-m}$ and $x_2 \in \mathcal{X}_2 \subset \mathbb{R}^m$, and the system can be rewritten as

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} f_1(x(t), t) \\ f_2(x(t), t) + \bar{B}(x(t), t)u(t) \end{bmatrix}, \tag{2.17}$$

where $f_1 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n-m}$ and $f_2 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ are the two components of the drift dynamics of (2.14).

Controllability form In this form, the control-affine system (2.15) can be split into m subsystems, each of them being a perturbed chain of integrators. Consider the state vector expressed as

$$x(t) = \begin{bmatrix} x_1^\top(t) & x_2^\top(t) & \dots & x_m^\top(t) \end{bmatrix}^\top,$$

where $x_i \in \mathbb{R}^{n_i}$, for $i \in \{1, 2, \dots, m\}$, with $n_i \in \mathbb{N}$ being the degree of the i-th subsystem such that $\sum_{i=1}^m = n$. Each subsystem is characterized by the dynamics

$$\dot{x}_i = A_i x_i(t) + f_i(x(t), t) + b_i(x(t), t)u(t), \tag{2.18}$$

where $A_i \in \mathbb{R}^{n_i\times n_i}$, $f_i(x(t), t) \in \mathbb{R}^{n_i}$, and $b_i(x(t), t) \in \mathbb{R}^{n_i\times m}$ are defined as

$$f_i(x(t), t) = \begin{bmatrix} 0_{n_i-1} \\ f_{i,0}(x(t), t) \end{bmatrix}, \quad b_i(x(t), t) = \begin{bmatrix} 0_{(n_i-1)\times m} \\ b_{i,0}^\top(x(t), t) \end{bmatrix}, \quad A_i = \begin{bmatrix} 0_{n_i-1} & I_{n_i-1} \\ 0 & 0_{n_i-1}^\top \end{bmatrix},$$

with $f_{i,0}(x(t), t) \in \mathbb{R}$ and $b_{i,0}(x(t), t) \in \mathbb{R}^m$. Then, the overall system is written as

$$\dot{x}(t) = Ax(t) + \bar{f}(x(t), t) + \bar{B}(x(t), t)u(t), \tag{2.19}$$

where $A = \operatorname{diag}\{A_i\}_{i=1}^m$, $\bar{f}(x(t), t) = \begin{bmatrix} f_1^\top(x(t), t) & f_2^\top(x(t), t) & \dots & f_m^\top(x(t), t) \end{bmatrix}^\top$, and $\bar{B}(x(t), t) = \begin{bmatrix} b_1^\top(x(t), t) & b_2^\top(x(t), t) & \dots & b_m^\top(x(t), t) \end{bmatrix}^\top$.

Normal form Consider a control-affine system in the form (2.15), characterized by a relative degree vector $\bar{r} = \begin{bmatrix} r_1 & r_2 & \cdots & r_m \end{bmatrix}^\top$ and relative degree $r \leq n$. Let $z_{i,j}$, with $(i,j) \in \{1,2,\ldots,m\} \times \{1,2,\ldots,r_i - 1\}$, denote the $(j-1)$-th derivative of the i-th output variable $y_i = \psi_i(x(t),t)$. Then, the normal canonical form of the system is

$$\begin{cases} \dot{z}_{i,j} = z_{i,j+1} & \text{if } j \in \{1,2,\ldots,r_i - 1\} \\ \dot{z}_{i,r_i} = \alpha_i(z,\eta) + \sum_{k=1}^{m} \beta_{i,k}(z,\eta)\, u_k(t) & \text{if } j = r_i \\ \dot{\eta} = \gamma(z,\eta) \end{cases} \tag{2.20}$$

where $z \in \mathbb{R}^{m \times (r-1)}$ is the matrix collecting the *external variables*, $\eta \in \mathbb{R}^{n-r}$ is the vector of the so-called *internal variables*. Their dynamics is defined by $\alpha_i, \beta_{i,k} : \mathbb{R}^{m\times(r-1)} \times \mathbb{R}^{n-r} \to \mathbb{R}$ and $\gamma : \mathbb{R}^{m\times(r-1)} \times \mathbb{R}^{n-r} \to \mathbb{R}^{n-r}$. In particular, $\gamma(\cdot,\cdot)$ is a design function that describes the internal behavior of the system when input and initial conditions have been chosen to keep the output identically zero. For this reason, the term $\gamma(0_{m\times(r-1)},\eta)$ is called *zero-dynamics*.

2.3 Elements of classical Sliding Mode Control

2.3.1 The Sliding Manifold

As already mentioned in Section 2.1, the sliding manifold is the subspace of state space toward which the systems states are steered by the SMC controller. In the following, a more formal definition, along with some design examples, is provided.

Consider a control-affine system (2.14) and let $\sigma_i : \mathcal{X} \to \mathbb{R}$, with $i \in \{1,2,\ldots,m\}$, be the sliding variables. Then, for each σ_i, it is possible to define the corresponding sliding surface as the set

$$\{x \in \mathcal{X} : \sigma_i(x(t)) = 0\}. \tag{2.21}$$

Since the sliding manifold is the intersection of all the m sliding surfaces, it is possible to define the a new sliding variable $\sigma : \mathcal{X} \to \mathbb{R}^m$ that collects the sliding variables associated with the sliding surfaces, i.e.,

$$\sigma(x(t)) = \begin{bmatrix} \sigma_1(x(t)) & \sigma_2(x(t)) & \ldots & \sigma_m(x(t)) \end{bmatrix}^\top, \tag{2.22}$$

and then define the sliding manifold as

$$\{x \in \mathcal{X} : \sigma_i(x(t)) = 0, \forall i \in \{1,2,\ldots,m\}\}. \tag{2.23}$$

The sliding manifold $\sigma(x(t)) = 0_m$ can be designed as any nonlinear function of the states x [27, 28]. It is common do define it as the linear combination of the system states

$$\{x \in \mathcal{X} : \sigma(x(t)) = Cx(t) = 0\}, \tag{2.24}$$

where $C \in \mathbb{R}^{m \times n}$ is a design matrix.

In the case in which the system can be expressed in one of the canonical forms, then some specific structures for the sliding variable are known. In the following, the canonical forms introduced in Section 2.2.1 are considered.

Reduced form Since the vector of the system states can be split in two components $x_1 \in \mathcal{X}_1 \subset \mathbb{R}^{n-m}$ and $x_2 \in \mathcal{X}_2 \subset \mathbb{R}^m$, one can conveniently define the sliding manifold as the linear combination

$$\sigma(x(t)) = \sigma(x_1(t), x_2(t)) = C_1 x_1(t) + C_2 x_2(t), \tag{2.25}$$

where $C_1 \in \mathbb{R}^{m \times (n-m)}$ and $C_2 \in \mathbb{R}^{m \times m}$, with the latter being non-singular. When the system is in sliding mode, it exhibits the reduced order dynamics

$$\begin{bmatrix} x_2(t) \\ \dot{x}_1(t) \end{bmatrix} = \begin{bmatrix} -C_2^{-1} C_1 x_1(t) \\ f_1\left(\begin{bmatrix} x_1^\top(t) & -(C_2^{-1} C_1 x_1(t))^\top \end{bmatrix}^\top, t\right) \end{bmatrix}, \tag{2.26}$$

where f_1 is defined in (2.17). In addition, if f_1 is a linear function of the system states, i.e.,

$$f_1(x(t), t) = F_1 x_1(t) + F_2 x_2(t), \tag{2.27}$$

with $F_1 \in \mathbb{R}^{(n-m)\times(n-m)}$ and $F_2 \in \mathbb{R}^{(n-m)\times m}$, the reduced order dynamics becomes

$$\begin{bmatrix} x_2(t) \\ \dot{x}_1(t) \end{bmatrix} = \begin{bmatrix} -C_2^{-1} C_1 x_1(t) \\ (F_1 - F_2 C_2^{-1} C_1) x_1 \end{bmatrix}. \tag{2.28}$$

Then, if the pair F_1, F_2 is controllable, the desired dynamics of the system in sliding mode can be imposed by suitably defining the matrix $C_2^{-1} C_1$ using classical control techniques, e.g., pole, assignment, optimal control, etc. [26].

Controllability form Since the system in this form can be split in m different subsystems, one can design a sliding surface for each subsystem given by the sliding variable $\sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}$

$$\sigma_i(x_i(t)) = c_i^\top x_i(t), \tag{2.29}$$

for all $i \in \{1, 2, \ldots, m\}$, where $c_i \in \mathbb{R}^{n_i}$ is a design vector. When a sliding motion is exhibited the i-th surface, i.e., $\sigma_i(x_i) = 0$, the associated subsystem exhibits a reduced order dynamics

$$\begin{cases} \dot{x}_{i,j} = x_{i,j+1}, & \text{if } j \in \{1, 2, \ldots, n_i - 1\} \\ x_{i,n_i} = -\frac{1}{c_{i,n_i}} \sum_{j=1}^{n_i-1} c_{i,j} x_{i,j} & \text{if } j = n_i, \end{cases} \tag{2.30}$$

for $i \in \{1, 2, \ldots, m\}$. Hence, the vectors c_i must be chosen so that the polynomial defining x_{i,n_i} is Hurwitz.

Normal form For a system in this form, the stability is related to the one of the zero dynamics, which coincides with the equivalent system. Hence, it is convenient to design m different sliding variables $\sigma_i : \mathbb{R}^{r-1} \to \mathbb{R}$ as the linear combination of the external variables, i.e,

$$\sigma_i(z_i(t)) = c_i^\top z_i, \tag{2.31}$$

for all $i \in \{1, 2, \ldots, m\}$, with $c_i \in \mathbb{R}^{r-1}$.

2.3.2 The Control Law

As anticipated at the beginning of this chapter, the control laws generated by SMC are characterized by high frequency switching behavior. Depending on the type of system, different control laws have been proposed in the literature. In the following, some of the available options are presented.

Relay form For this type of controller, the i-th component of the control input vector $u(t)$ may assume two distinct values, depending on the sliding variable associated with i-th sliding variable. Let $\sigma : \mathcal{X} \to \mathbb{R}^m$ be the sliding variable vector in (2.22), then the control law is defined as

$$u_i(t) = \begin{cases} u_i^+(x(t), t) & \text{if } \sigma_i(x(t)) > 0, \\ u_i^-(x(t), t) & \text{if } \sigma_i(x(t)) < 0, \end{cases} \tag{2.32}$$

with $i \in \{1, 2, \ldots, m\}$. The values $u_i^+ \in \mathbb{R}$ and $u_i^- \in \mathbb{R}$ must be defined during the design phase.

Unit Vector approach Suitable in the case of MIMO systems, the unit vector approach [27] exploits a control law that relies on a unit vector which indicates the

direction toward the sliding manifold. In particular, the control law $u(t) \in \mathbb{R}^m$ is given by

$$u(t) = K(x(t), t)\frac{\sigma(x(t))}{\|\sigma(x(t))\|_2}, \tag{2.33}$$

where $\sigma : \mathcal{X} \to \mathbb{R}^m$ is the sliding variable vector in (2.22) and $K(x(t), t) \in \mathbb{R}^{m\times m}$ is a design gain matrix.

State Feedback Control with Switching Gain The control law is given by

$$u(t) = \Gamma(x(t))x(t), \tag{2.34}$$

where $\Gamma : \mathcal{X} \to \mathbb{R}^{m\times n}$ is a state-dependent gain matrix, whose elements depend on the sliding variable vector $\sigma : \mathcal{X} \to \mathbb{R}^m$, defined as in (2.22). In particular

$$\Gamma_{i,j}(x(t)) = \begin{cases} \gamma_{i,j}^{+} & \text{if } \sigma_i(x)x_j > 0 \\ \gamma_{i,j}^{-} & \text{if } \sigma_i(x)x_j < 0 \end{cases} \tag{2.35}$$

for $i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, n\}$. In this case, the design parameters are the values $\gamma_{i,j}^{+} \in \mathbb{R}$ and $\gamma_{i,j}^{-} \in \mathbb{R}$.

Augmented control The control law is obtained by combining the so-called *equivalent control* $u_{\text{eq}} \in \mathbb{R}^m$, i.e., the control law obtained by imposing the first derivative of sliding variable vector to zero (refer to the corresponding paragraph in Section 2.3.4 for more details). In particular

$$u_i(t) = u_{\text{eq},i}(t) + u_{r,i}(t), \tag{2.36}$$

with $i \in \{1, 2, \ldots, m\}$, where $u_{r,i} \in \mathbb{R}$ is a discontinuous control like the relay type or the unit vector.

2.3.3 Existence and Reaching conditions

A fundamental aspect that one must consider when dealing with SMC is the existence of the sliding mode. In simple words, a sliding mode exists if in the vicinity of the sliding manifold $\sigma(x(t)) = 0_m$, the tangent vector of the controlled system trajectory is always directed toward the sliding manifold. Moreover, let $t_r \in \mathbb{R}_{\geq 0}$ denote the so-called *reaching time*, i.e., the time in which the system states reach the sliding manifold starting from some initial conditions $x(t_0) = x_0 \in \mathcal{X}$. Then, if the condition $\sigma(x(t)) = 0_m$ is satisfied for $t \geq t_r$, then the sliding mode is said to be *ideal*. Note that, the existence of a reaching time allows to distinguish between two

phases of the system motion: the so-called *reaching phase*, during which the system states are approaching the manifold, and the *sliding phase*, in which the system evolves on the sliding manifold.

From a theoretical point of view, the existence problem of sliding mode can be treated as a stability problem. In particular, it must be ensured that the sliding manifold is attractive for some initial initial conditions $x(t_0) = x_0 \in \mathcal{X}$. To prove the attractiveness of the manifold one can rely on *Lyapunov's second method* [25].

In the case of SISO systems, a simple, but often effective, choice to prove the existence of a sliding mode is

$$v(x(t)) = \frac{1}{2}\sigma^2(x(t)), \tag{2.37}$$

for which we aim to obtain that the first time-derivative satisfies

$$\dot{v}(x(t)) = \sigma(x(t))\dot{\sigma}(x(t)) < 0. \tag{2.38}$$

The condition (2.38) is referred to as *reaching condition* [27]. In order to attain a sliding mode in a finite time t_r, it is fundamental that the the γ-*reaching condition*

$$\dot{v}(x(t)) \leq -\gamma\sqrt{2v(x(t))}, \tag{2.39}$$

which translates in to

$$\sigma(x(t))\dot{\sigma}(x(t)) \leq -\gamma|\sigma(x(t))|, \tag{2.40}$$

with $\gamma \in \mathbb{R}_{>0}$, is satisfied. Condition (2.40) expresses the fact that the sliding variable and its first time-derivative $\dot{\sigma}$ always have opposite sign, meaning that, when $\sigma < 0$, it will grow since $\dot{\sigma} > \gamma$ and, on the contrary, if $\sigma > 0$, this last one will decrease because $\dot{\sigma} < -\gamma$.

Theorem 2.1 (Finite time reaching). *Consider a control-affine system* (2.14) *with* $m = 1$*, initial conditions* $x(t_0) = x_0$ *and sliding variable* $\sigma : \mathcal{X} \to \mathbb{R}$*. If the condition* (2.40) *holds, then the reaching time* $t_r \geq t_0$ *is bounded above as*

$$t_r \leq t_0 + \frac{|\sigma(x_0)|}{\gamma}. \tag{2.41}$$

Proof. First, recall that, since $m = 1$, during the reaching phase, the sliding variable keeps the same sign as its initial condition, i.e., $\text{sign}\,(\sigma(x(t))) = \text{sign}\,(\sigma(x_0))$ all $t \in [t_0, t_r)$. Then, two sub-cases can be distinguished which depend on the value of $\sigma(x_0)$:

1. If $\sigma(x_0) > 0$, then $\sigma(x(t)) > 0$ for all $t \in [t_0, t_r)$ and it holds that

$$\dot{\sigma}(x(t)) \leq -\gamma.$$

Integrating with respect tot time between t_0 and t_r, one obtains

$$\int_{t_0}^{t_r} \dot{\sigma}(x(t))dt \leq -\int_{t_0}^{t_r} \gamma dt,$$
$$\sigma(x(t_r)) - \sigma(x(t_0)) \leq -\gamma(t_r - t_0).$$

Since the reaching time is defined as the time instant in which the sliding mode is enforced, then $\sigma(x(t_r)) = 0$. Hence, rearranging the terms and substituting $\sigma(x(t_0)) = \sigma(x_0)$, one obtains

$$t_r \leq t_0 + \frac{\sigma(x_0)}{\gamma} = t_0 + \frac{|\sigma(x_0)|}{\gamma}. \tag{2.42}$$

2. If $\sigma(x_0) < 0$, then $\sigma(x(t)) < 0$ for all $t \in [t_0, t_r)$ and one has that

$$\dot{\sigma}(x(t)) \geq \gamma.$$

Similarly to what done in the previous sub-case, one can integrate with respect tot time between t_0 and t_r to obtain

$$\int_{t_0}^{t_r} \dot{\sigma}(x(t))dt \geq \int_{t_0}^{t_r} \gamma dt,$$
$$\sigma(x(t_r)) - \sigma(x(t_0)) \geq \gamma(t_r - t_0).$$

Recalling $\sigma(x(t_r)) = 0$, rearranging the terms and substituting $\sigma(x(t_0) = \sigma(x_0)$ lead to

$$t_r \leq t_0 - \frac{\sigma(x_0)}{\gamma},$$

which, since $\sigma(x_0) < 0$, can be rewritten as

$$t_r \leq t_0 + \frac{|\sigma(x_0)|}{\gamma}.$$

□

For MIMO systems, the reaching condition can be obtained easily. The MIMO version of (2.37) is the quadratic Lyapunov function

$$v(x(t)) = \frac{1}{2}\sigma^\top(x(t))\sigma(x(t)), \tag{2.43}$$

for which the aim is to obtain that the first-time derivative satisfies

$$\dot{v}(x(t)) = \sigma^\top(x(t))\dot{\sigma}(x(t)). \tag{2.44}$$

As for the reaching condition, in the case of a MIMO system, it can be chosen to decide the convergence rate of the sliding variable having

$$\dot{\sigma}(x(t)) = -Q\text{sign}\left(\sigma(x(t))\right) - K\phi(\sigma(x(t))), \tag{2.45}$$

where $Q \in \mathbb{R}^{m\times m}$ and $K \in \mathbb{R}^{m\times m}$ are two positive definite diagonal matrices, while $\phi(\sigma(x(t))) = \begin{bmatrix}\phi_1(\sigma_i(x(t))) & \phi_2(\sigma_2(x(t))) & \cdots & \phi_m(\sigma_m(x(t)))\end{bmatrix}^\top$, with $\phi_i : \mathbb{R} \to \mathbb{R}$ satisfying

$$\sigma_i(x(t))\phi_i(\sigma_i(x(t))) > 0, \tag{2.46}$$

for $\sigma_i(x(t)) > 0$ and $i \in \{1, 2, \ldots, m\}$. The first term in (2.45) expresses a constant rate, while the second a variable one. A common choice is to make the second term an exponential rate by setting $\phi(\sigma_i(x(t))) = \sigma_i(x(t))$.

2.3.4 Solutions of the controlled system

The control law that keeps the system states on the sliding manifold is a discontinuous one. The dynamics of a system controlled via VSC, or more in general via VSC, is given by

$$\dot{x}(t) = \varphi(x(t), u(t), t), \tag{2.47}$$

where $\varphi : \mathcal{X} \times \mathbb{R}^m \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is a piece-wise continuous function, which makes the system in (2.47) behave like a continuous nonlinear system in different regions of the state space, with a discontinuous change of dynamics at the boundaries of such regions. Due to such a change of dynamics, a system controlled via VSC is generally referred to as Variable Structure System (VSS).

Since the dynamics of a VSS is dictated by nonlinear differential equations with discontinuous right hand side, classical results on solutions of differential equations do not hold. However, the evolution of a system on the sliding manifold is unique and approaches to achieve an analytical solution have been proposed. In the following, the *Filippov's method* [29] and the so-called *equivalent control approach* [10, 27] are presented.

Filippov's method This method provides a good interpretation but it is not used in practical application. Hence, only the SISO case is analyzed. Consider the SISO system of the n-th order

$$\dot{x}(t) = \varphi(x(t), u(t), t),$$

with $\varphi : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, control law $u(t) \in \mathbb{R}$ of the relay type (2.32). Then, it is possible to show that the controlled system on $\sigma(x(t)) = 0$ behaves so that it

follows the manifold. In particular, the the dynamics of the system on the manifold is coincides with a vector $\varphi_\sigma : \mathbb{R}^n \times [0,1] \to \mathbb{R}^n$ belonging to the convex hull of two vectors $\varphi^+ : \mathbb{R}^n \to \mathbb{R}^n$ and $\varphi^- : \mathbb{R}^n \to \mathbb{R}^n$, defined as

$$\varphi^+(x(t)) = \varphi(x(t), u^+, t), \quad \varphi^-(x(t)) = \varphi(x(t), u^-, t),$$

where u^+ and u^- are defined as in (2.32). The dynamics of the system is given by

$$\dot{x}(t) = \varphi_\sigma(x(t), \alpha) = \alpha\varphi^+(x(t)) + (1-\alpha)\varphi^-(x(t)), \tag{2.48}$$

where $\alpha \in [0,1]$. A graphical representation of (2.48) is provided in Figure 2.3. The value of α can be found solving for α the equation

$$\frac{\partial \sigma(x(t))}{\partial x} \cdot \varphi_\sigma(x(t), \alpha) = 0, \tag{2.49}$$

which highlights the fact that, since the gradient $\frac{\partial \sigma(x(t))}{\partial x}$ is orthogonal to the level lines, the vector $\varphi_\sigma(x(t), \alpha)$ follows the level line $\sigma(x(t)) = 0$ that defines the manifold. Substituting the definition of $\varphi_\sigma(x(t), \alpha)$, one has

$$\frac{\partial \sigma(x(t))}{\partial x} \cdot \Big(\alpha\varphi^+(x(t)) + (1-\alpha)\varphi^-(x(t))\Big), \tag{2.50}$$

which, rearranging the terms to isolate α, leads to

$$\alpha \frac{\partial \sigma(x(t))}{\partial x} \cdot \Big(\varphi^+(x(t)) - \varphi^-(x(t))\Big) = \frac{\partial \sigma(x(t))}{\partial x} \sigma(x(t)) \cdot \varphi^-(x(t)). \tag{2.51}$$

Then, the value of α for the computation of (2.48) is

$$\alpha = \frac{\frac{\partial \sigma(x(t))}{\partial x} \cdot \varphi^-(x(t))}{\frac{\partial \sigma(x(t))}{\partial x} \cdot (\varphi^+(x(t)) - \varphi^-(x(t)))}. \tag{2.52}$$

It is important to remark that the solution provided by (2.48) is valid only when the system is in sliding mode and exhibits a dynamics described by a differential equation with discontinuous right hand side. During the reaching phase, classical results on solutions of differential equations can be adopted as the controller is a continuous function outside the sliding manifold.

Equivalent Control Method Consider the control-affine system in (2.14) with $m > 1$, with $u(t)$ chosen as the relay form controller in (2.32). Then, assume that a sliding mode is enforced on the sliding manifold $\sigma(x(t)) = 0_m$ from the time instant $t_r \geq t_0$. Since in sliding mode, the state trajectories are forced on the sliding manifold, $\dot{\sigma}(x(t)) = 0_m$ for all $t \geq t_r$. From this observation, the idea behind of the equivalent control method is to find the so-called *equivalent controller* $u_{\text{eq}} \in \mathbb{R}^m$,

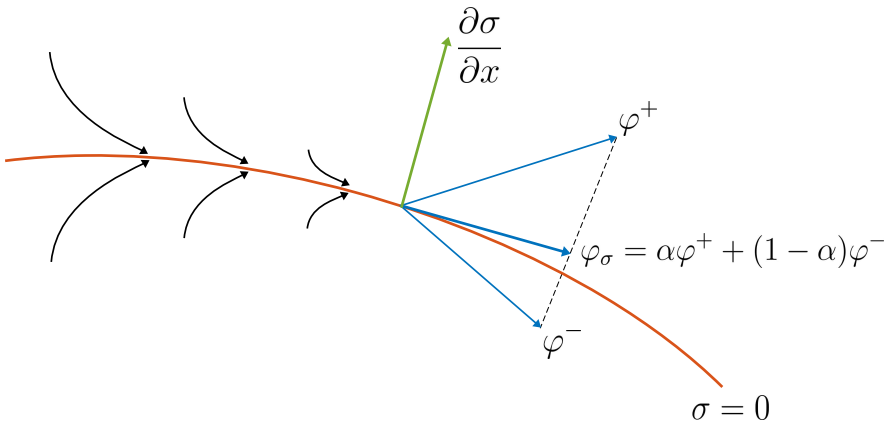


Figure 2.3: Graphical interpretation of the Filippov's solution.

i.e., the continuous control law that keeps the system states on the sliding manifold. To do so, one can impose $\dot{\sigma}(x(t)) = 0$ and solve for $u(t)$. Computing the time derivative of the sliding variable leads to

$$\dot{\sigma}(x(t)) = \frac{\partial \sigma(x(t))}{\partial x}\dot{x}(t) = \frac{\partial \sigma(x(t))}{\partial x}\left[f(x(t),t) + B(x(t),t)u(t)\right] = 0_m. \tag{2.53}$$

If one assumes that

$$\frac{\partial \sigma(x(t))}{\partial x}B(x(t),t) > 0_m, \tag{2.54}$$

for all $(x,t) \in \mathcal{X} \times \mathbb{R}_{\geq 0}$, then the equivalent controller can be computed as

$$u_{\text{eq}}(t) = -\left(\frac{\partial \sigma(x(t))}{\partial x}B(x(t),t)\right)^{-1}\left(\frac{\partial \sigma(x(t))}{\partial x}f(x(t),t)\right). \tag{2.55}$$

Then, substituting (2.55) in (2.14) it is possible to define the dynamics of the controlled system while in sliding mode as

$$\begin{aligned}\dot{x}(t) &= f(x(t),t) + B(x(t),t)u_{\text{eq}}(t)\\ &= f(x(t),t) - B(x(t),t)\left(\frac{\partial \sigma(x(t))}{\partial x}B(x(t),t)\right)^{-1}\left(\frac{\partial \sigma(x(t))}{\partial x}f(x(t),t)\right)\\ &= \left(I_n - B(x(t),t)\left(\frac{\partial \sigma(x(t))}{\partial x}B(x(t),t)\right)^{-1}\frac{\partial \sigma(x(t))}{\partial x}\right)f(x(t),t).\end{aligned} \tag{2.56}$$

2.3.5 Robustness Property and SMC Design

The main context of application of SMC is the one of systems affected by uncertainties and external disturbances. Consider the control-affine system affected by

uncertainties given by

$$\dot{x}(t) = f(x(t),t) + \Delta f(x(t),\theta(t),t) + (B(x(t) + \Delta B(x(t),\theta(t),t))) \, u(t), \tag{2.57}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ and $u \in \mathbb{R}^m$ represent the state and the input of the system, respectively, $f : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is the nominal drift dynamics, and $B : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n \times m}$ is the nominal control effectiveness matrix. As for $\Delta f : \mathcal{X} \times \Theta \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and $\Delta B : \mathcal{X} \times \Theta \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n \times m}$, they represent the model uncertainties, they are dependent on a time-varying unknown parameter vector $\theta \in \Theta \subset \mathbb{R}^p$, with Θ being a compact set, and they satisfy the so-called *matching condition*.

Definition 2.3 (Matching condition). *A vector $v \in \mathbb{R}^n$ is said to be matched if*

$$v \in \operatorname{span}\{B(x(t),t)\},$$

for all $x \in \mathcal{X}$ and $t \in \mathbb{R}_{\geq 0}$.

The uncertainties can be then rewritten as

$$\Delta f(x(t),\theta(t),t) + \Delta B(x(t),\theta(t),t) = B(x(t),t) w(x(t),\theta(t),t), \tag{2.58}$$

with $w \in \mathbb{R}^m$ being the vector of matched uncertainties, then, it is possible to rewrite (2.57) as if the uncertainties act as a disturbance entering though the input channel as

$$\dot{x}(t) = f(x(t),t) + B(x(t),t) \left(u(t) + w(x(t),\theta(t),t)\right). \tag{2.59}$$

Moreover, the perturbation vector $w(x(t),\theta(t),t)$ satisfies the following assumption.

Assumption 2.1 (Bounded perturbation). *There exists a known scalar function $\kappa : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ such that the norm of perturbation vector $w(x(t),\theta(t),t)$ is bounded above as*

$$\|w(x(t),\theta(t),t)\| \leq \kappa(x(t),t), \tag{2.60}$$

for all $x \in \mathcal{X}$ and $t \in \mathbb{R}_{\geq 0}$.

It has been proven that, when the states lie on the sliding manifold, the system is robust to disturbances that enters the system in the same channel of the input (see, for example, [30], for more details). In the following, a possible design for the control law that ensures a sliding mode $\sigma(x(t)) = 0_m$ in finite time, making the system robust to $w(x(t),\theta(t),t)$, is provided.

Let Assumption 2.1 holds. Then, it is possible to design an augmented controller like the one presented in Section 2.3.2, i.e.,

$$u(t) = u_{\text{eq}}(t) + u_r(t), \tag{2.61}$$

with $u_{\text{eq}} \in \mathbb{R}^m$ being the equivalent control law and $u_r \in \mathbb{R}^m$ a relay type controller. The objective of the discontinuous control law u_r is to make the system robust to $w(x(t), \theta(t), t)$, while u_{eq} is designed to impose the behavior of the system once the perturbations are rejected.
In particular, u_{eq} can be designed solving $\dot{\sigma}(x(t))$ in the case of $w(x(t), \theta(t), t) = 0_m$, while u_r is chosen to ensure that the system achieves a sliding mode $\sigma(x(t)) = 0_m$.

To this end, one can chose the Lyapunov function as detailed in [31], as

$$v(x(t)) = \frac{1}{2}\sigma^\top(x(t))\sigma(x(t)),$$

with first time-derivative

$$\begin{aligned}\dot{v}(x(t)) &= \sigma^\top(x(t))\dot{\sigma}(x(t)) \\ &= \sigma^\top(x(t))\frac{\partial\sigma(x(t))}{\partial x}\left[f(x(t),t) + B(x(t),t)\Big(u(t) + w(x(t),\theta(t),t)\Big)\right].\end{aligned}$$

Then, substituting $u(t)$ as in (2.61), and having $u_{\text{eq}}(t)$ as in (2.55), one has that

$$\dot{v}(x(t)) = \sigma^\top(x(t))\frac{\partial\sigma(x(t))}{\partial x}B(x(t),t)\Big(u_r(t) + w(x(t),\theta(t),t)\Big). \tag{2.62}$$

In order to ensure $v(x(t)) < 0$, one can exploit the knowledge about the perturbation bound in Assumption 2.1 and design the discontinuous control as

$$u_r(t) = -(\kappa(x(t),t) + \alpha)\frac{B^\top(x(t),t)\frac{\partial v(x(t))}{\partial x}}{\left\|B^\top(x(t),t)\frac{\partial v(x(t))}{\partial x}\right\|}, \tag{2.63}$$

where $\alpha \in \mathbb{R}_{>0}$ is a design constant which defines the reaching time, as previously detailed.

2.3.6 The Chattering problem

The generation of an ideal sliding mode requires an infinite control frequency, which is not obtainable when SMC is implemented in the practice. Since the implementation is done on digital controllers, the process signals are always sampled with a certain (possibly small) sampling time. This introduces the so-called *chattering*, i.e., high frequency oscillation in the state trajectory, having a negative impact on the performance of the control. As highlighted in Figure 2.4, if chattering is present, then the system trajectories do not stay exactly on the sliding manifold $\sigma(x(t)) = 0_m$, but switches around it in a boundary layer $\|\sigma(x(t))\| \leq \epsilon$, with $\epsilon \in \mathbb{R}_{>0}$. In such a condition, the sliding mode is said to be *practical*. The amplitude of the chattering effect is usually proportional to the gain of the discontinuous controller [10] and, in order to alleviate its effect, chattering alleviation techniques have been proposed in the literature (see, e.g., [32, 33], among others).

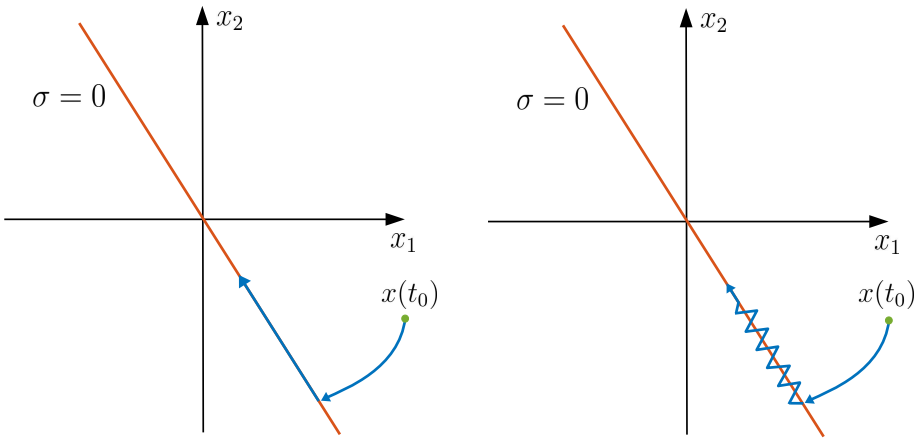


Figure 2.4: Example of an ideal sliding mode (left) and a sliding mode with chattering (right).

2.3.7 Approximability Property

As anticipated in Section 2.3.6, it is impossible to have an infinite frequency switch of the control law, causing a sliding motion only around the sliding manifold $\sigma(x(t)) = 0_m$ and not exactly on that. The results of the studies related to the behavior of the system near the sliding manifold are gathered under the so-called *approximability property* [34, 35], which is recalled in the following. Consider the multi-input system

$$\dot{x}^\star(t) = f(x^\star(t), t) + B(x^\star(t), t)u(t), \tag{2.64}$$

where $u(t)$ is the ideal SMC law which ensures $\sigma(x^\star(t)) = 0_m$. Let now $\tilde{u}(t)$ be a control input which includes non-idealities. The system becomes

$$\dot{x}(t) = f(x(t), t) + B(x(t), t)\tilde{u}(t), \tag{2.65}$$

and the sliding mode is achieved in a ϵ-neighborhood of the manifold, i.e.,

$$\Omega_\epsilon := \{x(t) \in \mathcal{X} : \|\sigma(x(t))\| \leq \epsilon\}, \tag{2.66}$$

with $\epsilon \in \mathbb{R}_{>0}$. Then, the following theorem holds.

Theorem 2.2 (Approximability Property [26]). *If it is true that*

1. *there exists a solution $x(t)$ to (2.65) in the interval $t \in [t_0, T]$, and $x(t)$ belongs to the ϵ-neighborhood of the sliding manifold (2.66);*

2. *there exists a Lipschitz constant L for the equivalent dynamics*

$$\dot{x}^\star(t) = f(x^\star(t),t) - B(x^\star(t),t)\left(\frac{\partial\sigma(x(t))}{\partial x}B(x^\star(t),t)\right)^{-1}\frac{\partial\sigma(x(t))}{\partial x}f(x^\star(t),t);$$

3. *there exist bounded partial derivatives of* $B(x(t),t)\left(\frac{\partial\sigma(x(t))}{\partial x}B(x(t),t)\right)^{-1}$;

4. *there exist two constants* $M, N \in \mathbb{R}_{>0}$ *such that*

$$\|f(x(t),t) + B(x(t),t)\tilde{u}(t)\| \leq M + N\|x\|;$$

then, for any pair of solutions of solutions $x(t)$ *and* $x^\star(t)$ *with initial conditions* $\|x(t_0) - x^\star(t_0)\| \leq \alpha_1\epsilon$, *with* $\alpha_1 \in \mathbb{R}_{>0}$, *there exists a constant* $\alpha_2 \in \mathbb{R}_{>0}$ *such that* $\|x(t) - x^\star(t)\| \leq \alpha_2\epsilon$, *for all* $t \in [t_0, T]$. *Moreover, if system* (2.64) *achieves a sliding mode asymptotically, then the theorem holds for* $T \to \infty$.

2.4 Integral Sliding Mode

The robustness to matched perturbations is guaranteed only when the system is in sliding mode, i.e., for $t \geq t_r$, making the system sensitive to disturbances during the reaching phase. To mitigate this problem, it would be possible to use high gain switching controller to have a smaller t_r. However, this approach could cause damages to the plant actuators [36].

To cope with such a problem, Utkin and Shi introduced the concept of *integral sliding mode* (ISM) in [36]. In such a work, they propose a novel SMC strategy which, thanks to a modification of the sliding variable, eliminates the reaching phase, achieving a robust motion on the whole state space. Since the complete state space can be seen as the sliding manifold, the system in sliding mode is not characterized by order reduction.

Consider the nonlinear multi-input system

$$\dot{x}(t) = f(x(t),t) + B(x(t),t)u(t) + h(x(t),t), \tag{2.67}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$, with $\mathcal{X}$ being a compact set containing the origin, represent the system states, $u \in \mathbb{R}^m$ is the control input vector, $f : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is the drift dynamics, $B : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n\times m}$ is the control effectiveness matrix, while $h : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is the vector of disturbances. Note that, even though ISM has been extended to deal with unmatched disturbances (see, e.g., [37] and [38]), in the following, the term $h(x(t),t)$ satisfies the matching condition (Definition 2.3).

Moreover, the disturbance is bounded according to the following assumption.

Assumption 2.2. *There exists a known positive scalar function* $\bar{h} : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ *that bounds the disturbance term as*

$$h(x(t), t)) \in \mathcal{H}, \qquad \mathcal{H} \coloneqq \{v(x(t), t) \in \mathbb{R}^n : \|v(x(t), t)\| \leq \bar{h}(x(t), t)\},$$

for all $x \in \mathcal{X}$ *and* $t \in \mathbb{R}_{\geq 0}$.

In the ISM control framework, the control law has a structure similar to the one of an augmented-type controller, being

$$u(t) = u_n(t) + u_r(t), \tag{2.68}$$

where $u_r(t) \in \mathbb{R}^m$ is a *switching controller* whose objective is to make the system robust against $h(x(t), t)$, while $u_n(t) \in \mathbb{R}^m$ is the so-called *nominal controller*, which is designed to stabilize the system (2.67) as if it was not affected by h.

As for the sliding variable $\sigma : \mathcal{X} \to \mathbb{R}^m$, which is now referred to as *integral sliding variable*, it is defined by the sum of two components

$$\sigma(x(t)) = \sigma_0(x(t)) - z(x(t)), \tag{2.69}$$

where $\sigma_0 : \mathcal{X} \to \mathbb{R}^m$ is the SMC conventional sliding variable chosen, for example, in one of the ways described in Section 2.3.1 and satisfying the following assumption.

Assumption 2.3. *The conventional sliding variable* $\sigma_0 : \mathcal{X} \to \mathbb{R}^m$ *is chosen so that the matrix*

$$\frac{\partial \sigma_0(x(t))}{\partial x} B(x(t), t) \in \mathbb{R}^{m \times m},$$

with $\frac{\partial \sigma_0}{\partial x} \in \mathbb{R}^{m \times n}$ *being the Jacobian of* σ_0 *with respect to* x*, is positive definite for all* $x \in \mathcal{X}$ *and* $t \in \mathbb{R}_{\geq 0}$.

As for $z : \mathcal{X} \to \mathbb{R}^m$, it is the so-called *transient function* and it is designed on the system controlled by the nominal controller as

$$z(x(t)) = \sigma_0(x_0) + \int_{t_0}^{t} \frac{\partial \sigma_0(x(\tau))}{\partial x} \Big(f(x(\tau), \tau) + B(x(\tau), \tau) u_n(\tau) \Big) d\tau. \tag{2.70}$$

Such a choice of the transient variable ensures that, for $t = t_0$, it holds that $\sigma(x(t_0)) = \sigma_0(x_0) - z(x_0) = \sigma_0(x_0) - \sigma_0(x_0) = 0_m$. Hence, the system is on the sliding manifold from the initial time instant.

2.4.1 Existence and Robustness Property

The following theorem presents the existence conditions of an ISM for $t \geq t_0$.

Theorem 2.3 (ISM Existence). *Given the nonlinear perturbed multi-input system* (2.67) *with matched uncertainty* $h(x(t),t)$ *satisfying Assumption 2.2, integral sliding variable* $\sigma(x(t))$ *defined as in* (2.69)*, and controlled via an ISM controller* $u(t)$ *in* (2.68)*, with discontinuous controller* $u_r(t)$ *chosen according to the unit vector approach in* (2.33)*, i.e,*

$$u_r(t) = -\rho(x(t),t)\frac{\sigma(x(t),t)}{\|\sigma(x(t),t)\|}. \tag{2.71}$$

Then, if the control gain $\rho(x(t),t) \in \mathbb{R}_{>0}$ *is chosen so that it satisfies*

$$\rho(x(t),t) > \frac{\left\|\frac{\partial \sigma_0(x(t))}{\partial x}\right\| \bar{h}(x(t),t)}{\underline{\lambda}\left(\frac{\partial \sigma_0(x(t))}{\partial x} B(x(t),t)\right)} + \eta, \tag{2.72}$$

where $\bar{h}(x(t),t)$ *is the function introduced in Assumption 2.2,* $\eta \in \mathbb{R}_{>0}$ *is a design constant, while* $\underline{\lambda}(\cdot)$ *denotes the smallest in norm eigenvalue of its argument, then a sliding mode* $\sigma(x(t)) = 0_m$ *is enforced for* $t \geq t_0$*.*

Proof. Let $v : \mathcal{X} \to \mathbb{R}$ be the Lyapunov function chosen as

$$v(x(t)) = \frac{1}{2}\sigma^\top(x(t))\sigma(x(t)).$$

Then, its first time-derivative can be computed as

$$\begin{aligned}
\dot{v}(x(t)) &= \sigma^\top(x(t))\dot{\sigma}(x(t)) \\
&= \sigma^\top(x(t))\Big[\dot{\sigma}_0(x(t)) - \dot{z}(x(t))\Big] \\
&= \sigma^\top(x(t))\left[\frac{\partial \sigma_0(x(t))}{\partial x}\dot{x}(t) - \dot{z}(x(t))\right] \\
&= \sigma^\top(x(t))\left[\frac{\partial \sigma_0(x(t))}{\partial x}\Big(f(x(t),t) + B(x(t),t)(u_n(t)+u_r(t)) + h(x(t),t)\Big) + \right. \\
&\quad \left. - \frac{\partial \sigma_0(x(t))}{\partial x}\Big(f(x(t),t) + B(x(t),t)u_n(t)\Big)\right] \\
&= \sigma^\top(x(t))\frac{\partial \sigma_0(x(t))}{\partial x}\Big(B(x(t),t)u_r(t) + h(x(t),t)\Big)
\end{aligned}$$

Substituting $u_t(t)$ as in (2.71) and omitting the time dependence of the state for sake of readability, it holds that

$$\dot{v}(x) = \sigma^\top(x)\frac{\partial \sigma_0(x)}{\partial x}B(x,t)h(x,t) - \rho(x,t)\sigma^\top(x)\frac{\partial \sigma_0(x)}{\partial x}B(x,t)\frac{\sigma(x)}{\|\sigma(x)\|}.$$

If Assumption 2.3 hold, then $\dot{v}$ can be upper bounded as

$$\dot{v}(x) \leq -\rho(x,t)\frac{\|\sigma(x)\|}{2}\underline{\lambda}\left(\frac{\partial \sigma_0(x)}{\partial x}B(x,t) + B^\top(x,t)\left(\frac{\partial \sigma_0(x)}{\partial x}\right)^\top\right) +$$

$$+ \|\sigma(x)\| \left\| \frac{\partial \sigma_0(x)}{\partial x} \right\| \|h(x,t)\|.$$

Thanks to Assumption 2.2, the norm of the perturbation is bounded, leading to

$$\dot{v}(x) \leq -\rho(x,t)\frac{\|\sigma(x)\|}{2}\underline{\lambda}\left(\frac{\partial \sigma_0(x)}{\partial x}B(x,t)\right) + \|\sigma(x)\| \left\| \frac{\partial \sigma_0(x)}{\partial x} \right\| \bar{h}(x,t).$$

Then, choosing $\rho(x,t)$ as in (2.72), one has

$$\dot{v}(x(t)) \leq -\eta \|\sigma(x(t))\| < 0,$$

which ensures that a sliding mode $\sigma(x(t)) = 0_m$ is achieved in finite time. Moreover, since the integral sliding variable components are selected so that $\sigma(x(t_0)) = 0_m$, one can conclude that the sliding mode is enforced for $t \geq t_0$. □

To highlight the robustness capabilities of the ISM controller, the equivalent control law is computed. First, it is worth recalling that the equivalent controller is the continuous control law that keeps the system states on the sliding manifold and it is computed imposing

$$\dot{\sigma}(x(t)) = \frac{\partial \sigma_0(x(t))}{\partial x}\Big(B(x(t),t)u_r(t) + h(x(t),t)\Big) = 0_m, \tag{2.73}$$

and then solving for $u_r(t)$ obtaining

$$u_{\text{eq}}(t) = -\left(\frac{\partial \sigma_0(x(t))}{\partial x}B(x(t),t)\right)^{-1}\frac{\partial \sigma_0(x(t))}{\partial x}h(x(t),t). \tag{2.74}$$

Under the assumption of matched perturbations one can express

$$h(x(t),t) = B(x(t),t)w(t)$$

with $w : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$, and then rewrite (2.74) as

$$\begin{aligned} u_{\text{eq}}(t) &= -\left(\frac{\partial \sigma_0(x(t))}{\partial x}B(x(t),t)\right)^{-1}\frac{\partial \sigma_0(x(t))}{\partial x}B(x(t),t)w(t). \\ &= -B^{-1}(x(t),t)\left(\frac{\partial \sigma_0(x(t))}{\partial x}\right)^{-1}\frac{\partial \sigma_0(x(t))}{\partial x}B(x(t),t)w(t) \\ &= -B^{-1}(x(t),t)B(x(t),t)w(t) \\ &= -w(t) \end{aligned}$$

Hence, it is possible to conclude that, applying the discontinuous control $u(t)$ in (2.68), with $u_r(t)$ in (2.71) has the same effect to apply an ideal control law equal to the opposite of the disturbance.

2.4.2 Physical Interpretation of the Equivalent Control

As recalled in Section 2.4.1, the equivalent controller is a continuous control that maintains the system in sliding mode, having the same effect of the switching controller. As detailed in [10] and [36], it is possible to demonstrate that, in the practice, this last one corresponds to the sum of the equivalent controller plus a high-frequency term that cause the chattering. In particular, starting from the first-time derivative of the sliding variable in (2.73), with $h(x(t),t) = B(x(t),t)w(t)$, and rearranging the terms, one obtains

$$\begin{aligned} u_r(t) &= \left(\frac{\partial \sigma_0(x(t))}{\partial x} B(x(t),t)\right)^{-1} \left[\dot{\sigma}(x(t)) - \frac{\partial \sigma_0(x(t))}{\partial x} B(x(t),t)w(t)\right] \\ &= -w(t) + \left(\frac{\partial \sigma_0(x(t))}{\partial x} B(x(t),t)\right)^{-1} \dot{\sigma}(x(t)) \\ &= u_{\text{eq}}(t) + \left(\frac{\partial \sigma_0(x(t))}{\partial x} B(x(t),t)\right)^{-1} \dot{\sigma}(x(t)) \end{aligned}$$

Since the equivalent controller corresponds to the low-frequency component of the discontinuous controller, it is possible to approximate it using a first-older filter characterized by the differential equation

$$\mu \dot{\hat{u}}_{\text{eq}}(t) = u_r(t) - \hat{u}_{\text{eq}}(t), \tag{2.75}$$

with initial condition $\hat{u}_{\text{eq}}(t_0) = 0_m$ and filtering constant $\mu \in (0,1)$), chosen as not to distort the harmonic content of $h(x(t),t)$. The solution of (2.75) is given by

$$\hat{u}_{\text{eq}}(t) = \frac{1}{\mu} \int_{t_0}^{t} e^{-\frac{(t-\tau)}{\mu}} u_r(\tau). \tag{2.76}$$

2.4.3 ISM control example

Consider the Duffing oscillator dynamics in (2.3), with $\eta = 1$. Then, it is possible to describe the system in the control-affine form (2.67), with drift dynamics $f(x(t),t) \in \mathbb{R}^2$ and control effectiveness matrix $B(x(t),t) \in \mathbb{R}^2$ defined as

$$f(x(t),t) = \begin{bmatrix} x_2(t) \\ x_1(t)(1 - x_1^2(t)) - x_2(t) + u(t) \end{bmatrix} \qquad B(x(t),t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

As for the disturbance $h(x(t),t)$, in this example it has been chosen as

$$h(x(t),t) = B(x(t),t)w(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} 0.15 \sin(2t)\cos(0.1t)$$

which satisfies Assumption 2.2, with $\bar{h} = 0.2$. Then, the integral sliding variable $\sigma(x(t)) \in \mathbb{R}$ is as in (2.69), with $\sigma_0(x(t)) = (x_1 - x_{1,d}) + (x_2 - x_{2,d}) \in \mathbb{R}$, with $x_{1,d}$

and $x_{2,d}$ being a desired equilibrium point, and transient variable $z(x(t)) \in \mathbb{R}$ as in (2.70). Choosing the nominal control law as $u_n(t) = -4x_1(t) + x_1^3(t) - 2x_2(x(t)) \in \mathbb{R}$ and the switching controller as $u_r(t) = -0.25\text{sign}(\sigma(t)) \in \mathbb{R}$, whose gain is chosen accordingly with Theorem 2.3, lead to the results in Figures 2.5 and 2.6. Moreover, as shown in Figure 2.7, applying a first order filter to u_r, with $\mu = 0.01$, it is possible to provide an approximation of the equivalent control.

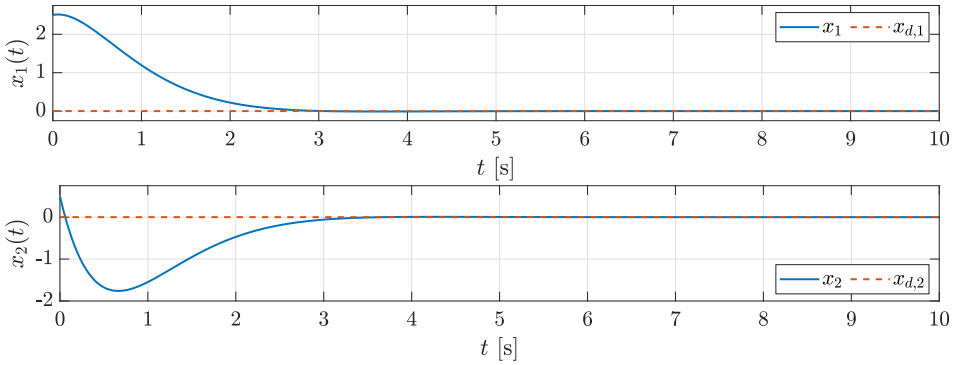


Figure 2.5: Time evolution of the system states

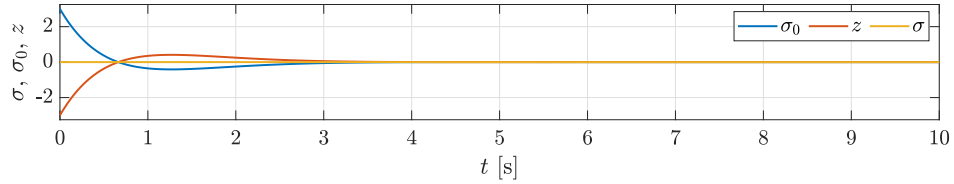


Figure 2.6: Time evolution of the sliding variable

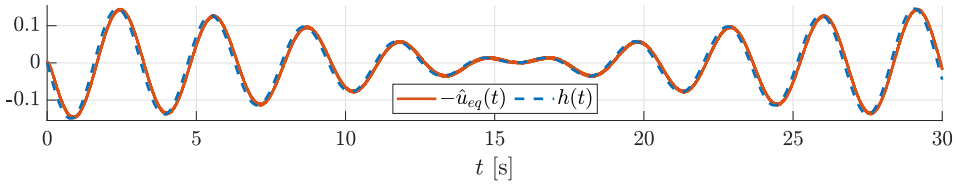


Figure 2.7: Time evolution of equivalent control

2.5 Higher Order Sliding Mode Control

As detailed in Section 2.3.6, due to its discontinuous nature, SMC can produce the so-called chattering effect, which, in some cases, can be disruptive for the controlled plant. To overcome such an effect, several techniques have been proposed.

For example, on-line estimation of the equivalent control has been employed to reduce the discontinuous control component [39]. Another method which has been widely adopted consists in moving the discontinuity (instrumental to ensure a sliding mode in finite time) into some derivative of the control law, letting the control signal fed to the system to be continuous. Such an approach, called Higher Order Sliding Mode (HOSM) control, has been extensively studied in the literature [40, 41, 42, 43, 44, 45] and it is characterized by the fact that it steers to zero not only the sliding variable, but also on its time derivatives.

Consider the nonlinear multi-input control-affine in (2.15)

$$\dot{x}(t) = f(x(t), t) + B(x(t), t)u(t)$$

and let $\sigma : \mathcal{X} \to \mathbb{R}^n$ be the sliding variable as in (2.22). Then, the system is said to be in *r-sliding mode* if, for $t \geq t_r \geq t_0$, its states reaches the so-called *r-sliding manifold*, defined as

$$\begin{aligned}\Big\{x(t) \in \mathcal{X}, u(t) \in \mathbb{R}^m : \sigma(x(t)) &= L_f\sigma(x(t)) + \sum_{i=1}^{m} L_{B^{(i)}}\sigma(x(t))u \\ &= \dots \\ &= L_f^{(r-1)}\sigma(x(t)) + \sum_{i=1}^{m} L_{B^{(i)}}L_f^{(r-2)}\sigma(x(t))u(t) = 0\Big\}\end{aligned}$$

where $L_f^{(r-1)}\sigma(x(t))$ is the $(r-1)$-th order Lie derivative of $\sigma(x(t))$ along the vector field $f(x(t))$. Note that the order of the sliding mode controller that enforces an r-sliding mode is r.

Consider, for sake of simplicity, the SISO system

$$\begin{cases}\dot{x}(t) = f(x(t), t) + b(x(t), t)u(t) \\ y(t) = \sigma(x(t), t)\end{cases} \tag{2.77}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$, $u \in \mathbb{R}$, and $\sigma : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a sufficiently smooth output function. The dynamics components $f : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and $b : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ are assumed unknown, but the relative degree of the system is available and equal to $r \in \mathbb{N}$, meaning that the term u appears explicitly in the r-th time derivative of y, having

$$\frac{\partial y^{(r)}}{\partial u} \neq 0 \implies \frac{\partial \sigma^{(r)}}{\partial u} \neq 0.$$

Then, one has that

$$\sigma^{(r)}(x(t)) = h(x(t), t) + g(x(t), t)u(t), \tag{2.78}$$

where $h : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ and $g : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ are suitable functions that satisfy the following assumption.

Assumption 2.4. *There exist some known constants $H, \underline{g}, \bar{g} \in \mathbb{R}_{>0}$ such that*

$$|h(x(t), t)| \leq H,$$
$$0 < \underline{g} \geq g(x(t)) \leq \bar{g}, \text{ or } -\bar{g} \geq g(x(t)) \leq -\underline{g} < 0,$$

meaning that function $g(x(t), t)$ has constant known sign.

Then, the original dynamical system implies the differential inclusion [46]

$$\sigma^{(r)}(x(t), t) = f_1 + f_2 u(t), \tag{2.79}$$

with $f_1 \in [-H, H]$ and $f_2 \in [\underline{g}, \bar{g}]$. It has been shown that, in order to make the r-sliding manifold associated with (2.79) finite-time attractive, and hence generate a r-sliding mode, any sliding mode controller of order r of the form

$$u(t) = k\, \psi \left(\sigma(x(t)), \dot{\sigma}(x(t)), \ldots, \sigma^{(r-1)}(x(t)) \right), \tag{2.80}$$

with $\psi(\cdot)$ being a discontinuous function and $k \in \mathbb{R}_{>0}$, can be used (see, e.g., [40, 42, 43, 44]). For example, in [40] two controllers have been proposed for $r = 1, 2$, i.e.,

$$u(t) = -k \,\text{sign}\left(\sigma(x(t))\right), \tag{2.81a}$$
$$u(t) = -k \,\text{sign}\left(\dot{\sigma}(x(t)) + |\sigma(x(t))|^{\frac{1}{2}} \text{sign}\left(\sigma(x(t))\right)\right). \tag{2.81b}$$

It is worth noticing that, if a sliding controller of order r is applied to a system with relative degree $d < r$, then the chattering phenomenon can be alleviated. In fact, if one introduces time derivatives of the control input $\dot{u}, u^{(2)}, \ldots, u^{(r-d-1)}$ as auxiliary variables and sets a new control variable $w(t) = u^{(r-d)}(t)$, then the relative degree of the resulting system with respect to w is r. Then, the control actually fed to the system u is a $(r-d-1)$-smooth function of time with $d < r-1$, a Lipschitz function with $d = r-1$, and a bounded discontinuous switching function with $d = r$ [26, 40].

As one can understand from (2.81), a sliding controller of order r requires the knowledge of $\sigma(x(t))$ and its derivatives $\dot{\sigma}(x(t)), \sigma^{(2)}(x(t)), \ldots, \sigma^{(r-1)}(x(t))$. If such a knowledge is not available, one could rely, for example, on the so-called *Levant's differentiator* [40]. In the case of $r = 2$, there are also methodologies that do not require the derivatives of the sliding variable (see, e.g., [33, 47]).

2.6 Adaptive Sliding Mode Control

The field of adaptive control theory studies controllers which include parameters that are not kept constant, but they must be adapted according to the system response [48]. In general, parameters are adapted so that the robustness of the method is maintained. Hence, adaptation laws are derived relying on Lyapunov analysis and making use of theoretical tools like *parameter projection*, which is described in Appendix A.

The main difference between *adaptive control* and *robust control* is that the former usually does not require a priory knowledge about the time-varying parameters, while the latter, which is the category to which SMC belongs, aims to guarantee stability of a system relying on some knowledge (usually bounds) on the uncertainties affecting the system. The two types of control have been often combined, generating powerful control techniques, as in the case of Adaptive Sliding Mode Control (ASMC). Fore some examples, one can refer to [49, 50, 51, 52, 53, 54], among others.

Consider a MIMO control-affine (2.67) system with control input $u(t) \in \mathbb{R}^m$ being defined according to the unit-vector approach

$$u(t) = -\rho(t)\frac{\sigma}{\|\sigma\|},$$

where $\rho(t) \in \mathbb{R}_{>0}$ is the discontinuous control gain, whose value must be sufficiently large to compensate the perturbation $h(x(t), t)$ and the effect of the rest of the dynamics. Hence, as mentioned in the previous sections, a knowledge on the bound of such a term is required. If, fore some reason, such a knowledge is not available, then the discontinuous control gain can be made adaptive according to one of the strategies cited above. For the reader's convenience, in the following, two adaptation strategies are presented.

Strategy 1 [52] The discontinuous control gain is described by a dynamics which depends on the norm of the sliding variable, i.e.,

$$\dot{\rho}(t) = \frac{1}{\alpha}\left\|\sigma(x(t))\right\|, \tag{2.82}$$

with initial condition $\rho(t_0) = \rho_0 \in \mathbb{R}_{>0}$ chosen in the design phase. The parameter $\alpha \in \mathbb{R}_{>0}$ acts as an adaptation rate and it is chosen by the designer. The basic idea behind the above adaptation scheme is that, if the sliding mode is not enforced ($\|\sigma(x(t))\| > 0$), then the gain is increased. As soon as the sliding mode is enforced

(assume at time $t_r \geq t_0$), the gain is no longer increased. The final value of the gain, denoted as $\rho_\infty \in \mathbb{R}_{>0}$, can be computed as

$$\begin{aligned}\rho_\infty &= \rho_0 + \int_{t_0}^{\infty} \dot{\rho}(\tau) d\tau \\ &= \rho_0 + \int_{t_0}^{t_r} \frac{\|\sigma\|}{\alpha} d\tau + \int_{t_r}^{\infty} \frac{\|\sigma\|}{\alpha} d\tau \\ &= \rho_0 + \int_{t_0}^{t_r} \frac{\|\sigma\|}{\alpha} d\tau \geq \rho_0.\end{aligned}$$

Strategy 2 [53] It is possible to lower the discontinuous control gain when the value of of the sliding variable is below a certain threshold $\varepsilon_\sigma \in \mathbb{R}_{>0}$ by having

$$\dot{\rho}(t) \begin{cases} \frac{\|\sigma(x(t))\|}{\alpha} \text{sign}\left(\|\sigma(x(t))\| - \varepsilon_\sigma\right) & \text{if } \rho(t) \geq \mu, \\ \mu & \text{if } \rho(t) < \mu, \end{cases} \tag{2.83}$$

where $\mu \in \mathbb{R}_{>0}$ is a small constant whose aim is to ensure that $\rho(t) > 0$, for all $t \geq t_0$.

Chapter 3

Preliminaries on Neural Networks and Learning

This chapter introduces the main concepts about Artificial Neural Networks and learning, instrumental for the development of the strategies presented in this dissertation. In particular, the mathematical formulation of multi-layer perceptron model is presented, along with the universal approximation theorem. Then, different conventional learning techniques are presented and discussed.

3.1 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a type of Artificial Neural Network (ANN). ANNs are mathematical models initially developed to understand the behavior and the information process capabilities of the nervous system, which can be seen as a dense network of neurons. A biological neuron, depicted in Figure 3.1, is a cell comprised by three main elements: the *soma* (the cell body), the *dendrites*, and the *axon*. In particular, the aim of dendrites is to receive signals from other neurons, while the axon, which is connected to the dendrites of other neurons, transmits the signal exiting the neuronal cell. The behavior of a neuron can be summarized as follows. The dendrites modulate the incoming signals from other neurons via transfer weighting coefficients and bring them into them into the cell body. This last one combines the signals coming from the various dendrites generating a composite signal which, as soon as it reaches a threshold, is transmitted through the axon. It is important to notice that, due to cell nonlinearities, the composite signal inside the soma is a nonlinear function of the combination of the signals entering the cell.

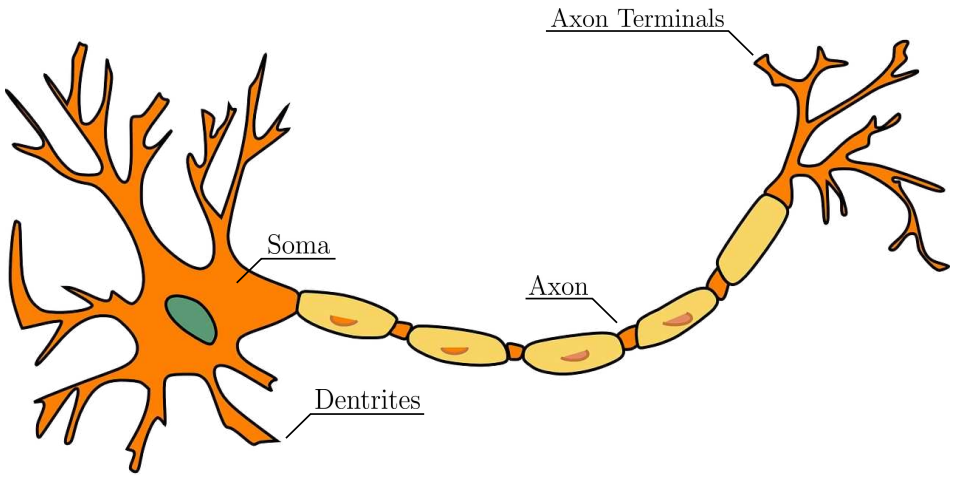


Figure 3.1: Graphical representation of a biological neuron and its main components.

3.1.1 The Perceptron model

Having in mind the behavior of biological neuron, the so-called *perceptron* model, proposed in [3] and depicted in Figure 3.2, is now described. Consider an input vector $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^n$, a weight vector $w = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}^\top \in \mathbb{R}^n$, a scalar bias $b \in \mathbb{R}$, and an activation function $\varsigma : \mathbb{R} \to \mathbb{R}$. Then, the output $y \in \mathbb{R}$ of the neuron is given by

$$y = \varsigma(w^\top x + b) = \varsigma\left(b + \sum_{i=1}^{n} x_i w_i\right). \tag{3.1}$$

If one augments the input vector and incorporates the weights and the bias in a single vector vector, it is possible to obtain two vectors $x_h \in \mathbb{R}^{n+1}$ and $v \in \mathbb{R}^{n+1}$, defined as

$$x_h = \begin{bmatrix} x^\top & 1 \end{bmatrix}^\top, \quad v = \begin{bmatrix} w^\top & b \end{bmatrix}^\top. \tag{3.2}$$

Hence, the expression of the neuron output given by (3.1) can be reformulated in a more compact form as

$$y = \varsigma(v^\top x).$$

Even though a single neuron could be effective in very simple problems [3], an higher number of neurons is required when the complexity of the problem increases. In particular, more complex ANNs can be built by organizing neurons in *layers*, i.e., groups of neurons which do not communicate with each other, but all receive the same input vector and whose output will be the input to the subsequent layer.

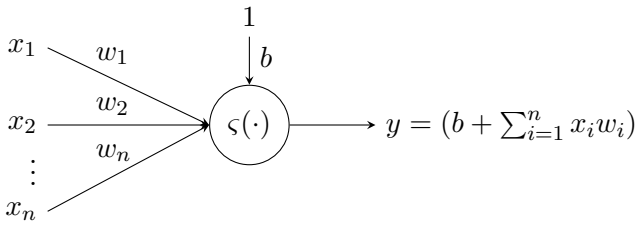


Figure 3.2: Mathematical model of the neuron.

Such an architecture is called MLP [4] and an example is provided in Figure 3.3. In the research reported in this dissertation, MLP models have been extensively used. Hence, their mathematical formulation it is now provided.

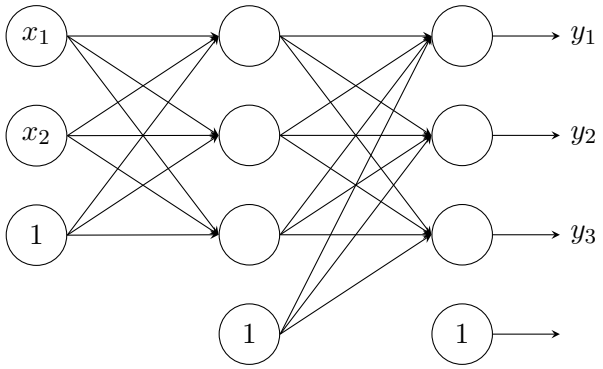


Figure 3.3: Example of a network with one hidden layer.

Consider a MLP composed by an input layer, $k \in \mathbb{N}_{\geq 1}$ hidden layers, and one output layer. The j-th layer is characterized by $L_j - 1$ activation functions and a scalar identity map which is used to introduce bias in the subsequent layer. Specifically, $L_0 - 1$ and $L_{k+1} - 1$ denote the size of inputs and outputs of the network. Moreover, each layer is characterized by a weight matrix $W_j \in \mathbb{R}^{(L_j - 1) \times (L_{j+1} - 1)}$ and by a bias vector $b_j \in \mathbb{R}^{L_{j+1} - 1}$, defined such that

$$W_j^\top \coloneqq \begin{bmatrix} w_{j,1}^\top \\ w_{j,2}^\top \\ \vdots \\ w_{j,L_{j+1}-1}^\top \end{bmatrix}, \quad b_j \coloneqq \begin{bmatrix} b_{j,1} & b_{j,2} & \cdots & b_{j,L_{j+1}-1}, \end{bmatrix}$$

where $w_{j,i} \in \mathbb{R}^{L_j - 1}$ and $b_{j,i} \in \mathbb{R}$ are, respectively, the weight vector and bias of the i-th neuron in the j-th layer, with $i \in \{1, 2, \dots L_{j+1} - 1\}$. Hence, it is possible to

define the output of the j-th layer with the quantity $\Phi_j \in \mathbb{R}^{L_{j+1}-1}$, defined as

$$\Phi_j \coloneqq \begin{cases} W_j^\top \varsigma_j(\Phi_j) + b_j & \text{for } j \in \{1, 2, \dots, k\} \\ W_0^\top x + b_0 & \text{for } j = 0, \end{cases} \tag{3.3}$$

where $\varsigma_j : \mathbb{R}^{L_j-1} \to \mathbb{R}^{L_j-1}$ is the vector of the activation functions of the j-th layer and it is defined so that

$$\varsigma_j(\Phi_{j-1}) = \begin{bmatrix} \varsigma_{j,1}(\Phi_{j-1,1}) & \varsigma_{j,2}(\Phi_{j-1,2}) & \dots & \varsigma_{j,L_j-1}(\Phi_{j-1,L_j-1}) \end{bmatrix},$$

with $\varsigma_{j,i} : \mathbb{R} \to \mathbb{R}$ being the activation function of the i-th neuron of the j-th layer, while $\Phi_{j-1,i} \in \mathbb{R}$ is the i-th element of the vector $\Phi_{j-1} \in \mathbb{R}^{L_j-1}$.

Recalling the definition of the augmented state vector x_h in (3.2), it is possible to simplify the the notation of the expression of the ANN by introducing, for each layer j, the function vector $\phi_j : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j}$ and the the matrix $V_j \in \mathbb{R}^{L_j \times L_{j+1}}$. In particular, the former one is defined so that, given a vector $\alpha \in \mathbb{R}^{L_j}$

$$\phi_j(\alpha) \coloneqq \begin{bmatrix} \varsigma_j(\alpha_{[1:L_j-1]}) \\ \mathrm{id}(\alpha_{L_j}) \end{bmatrix}, \tag{3.4}$$

where $\mathrm{id} : \mathbb{R} \to \mathbb{R}$ is athe identity function $\mathrm{id}(a) = a$, for any $a \in \mathbb{R}$. As for the matrix V_j, it is defined so that it incorporates both the weight matrix W_j and the bias vector b_j as

$$V_j^\top \coloneqq \begin{bmatrix} W_j^\top & b_j \\ 0_{L_j-1}^\top & 1 \end{bmatrix},$$

for the layers up to the second to last one, i.e., $j \in \{0, 1, \dots, k-1\}$. Since the last layer does not need to produce the output element for the bias, the columns of V_k are not augmented. In fact, this last one is defined as

$$V_k^\top \coloneqq \begin{bmatrix} W_k^\top & b_k \end{bmatrix} \in \mathbb{R}^{(L_{k+1}-1)\times L_k}, \tag{3.5}$$

with $L_{k+1} - 1$ being the number of outputs of the ANN. Then, it is possible reformulate the output of the j-th layer, expressed originally as in (3.3) in a more convenient way, i.e.,

$$\Phi_j = \begin{cases} V_j^\top \phi_j(\Phi_{j-1}) & \text{for } j \in \{1, 2, \dots, k\} \\ V_0^\top x_h & \text{for } j = 0, \end{cases} \tag{3.6}$$

where $\phi_j(\Phi_{j-1})$ is computes as in (3.4) with $\alpha = \Phi_{j-1}$. Note that, now $\Phi_j \in \mathbb{R}^{L_{j+1}}$. For sake of completeness, the *forward-pass*, i.e., the evaluation of the output of each

layer of the ANN, is reported. In particular, one has that, given an input $x \in \mathbb{R}^n$, it holds that

$$\Phi_0 = V_0^\top x_h = \begin{bmatrix} W_0^\top & b_0 \\ 0_n^\top & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} w_0^\top x + b_0 \\ 1 \end{bmatrix}$$

$$\Phi_1 = V_1^\top \phi_1(\Phi_0) = \begin{bmatrix} W_1^\top & b_1 \\ 0_{L_1-1}^\top & 1 \end{bmatrix} \begin{bmatrix} \varsigma_1\left(\Phi_{0,[1:L_1-1]}\right) \\ \Phi_{0,L_1} \end{bmatrix} = \begin{bmatrix} W_1^\top \varsigma_1\left(\Phi_{0,[1:L_1-1]}\right) + b_1 \\ 1 \end{bmatrix}$$

$$\Phi_2 = V_2^\top \phi_2(\Phi_1) = \begin{bmatrix} W_2^\top & b_2 \\ 0_{L_2-1}^\top & 1 \end{bmatrix} \begin{bmatrix} \varsigma_2\left(\Phi_{1,[1:L_2-1]}\right) \\ \Phi_{1,L_2} \end{bmatrix} = \begin{bmatrix} W_2^\top \varsigma_2\left(\Phi_{1,[1:L_2-1]}\right) + b_2 \\ 1 \end{bmatrix}$$

$$\vdots$$

$$\Phi_k = V_k^\top \phi_k(\Phi_{k-1}) = \begin{bmatrix} W_k^\top & b_k \end{bmatrix} \begin{bmatrix} \varsigma_k\left(\Phi_{k-1,[1:L_k-1]}\right) \\ \Phi_{k-1,L_k} \end{bmatrix} = W_k^\top \varsigma_k\left(\Phi_{k-1,[1:L_k-1]}\right) + b_k.$$

3.1.2 Universal approximation capabilities of ANNs

One of the reasons for ANNs have been widely adopted in the years is their power of approximating a wide class of functions. Such a capability, usually referred to as *universal approximation property*, has been studied in works like [4, 55, 56, 6, 57] and hereafter recalled.

Theorem 3.1 (Universal Approximation [55]). *Let $f : \Omega \to \mathbb{R}^p$ be a continuous function defined over a compact set $\Omega \subset \mathbb{R}^n$ and $\Phi : \Omega \to \mathbb{R}^p$ an ANN with $k \in \mathbb{N}_{\geq 1}$ hidden layers whose output is computed as in* (3.6)*. Then, if the activation functions in the vectors ς_j, with $j \in \{0, 1, \ldots, k\}$ are not polynomial, it holds that*

$$f(x) = \Phi(x) + \varepsilon_\Phi(x), \tag{3.7}$$

where $\varepsilon_\Phi : \Omega \to \mathbb{R}^p$ being the approximation error. Moreover, there exists a constant $\bar{\varepsilon}_\Phi \in \mathbb{R}_{>0}$ such that

$$\sup_{x \in \Omega} \|\varepsilon_\Phi(x)\| \leq \bar{\varepsilon}_\Phi. \tag{3.8}$$

Note that, the value of $\Phi(x)$ in (3.7) corresponds to the output of the last layer of the DNN, i.e., $\Phi(x) = \Phi_k$, computed as in (3.6) with $j = k$.

The above result can be easily extended also to the case where the function that must be approximated is a matrix. Let $B : \Omega \to \mathbb{R}^{p \times q}$ be a continuous function, and $\Phi : \Omega \to \mathbb{R}^{pq}$. Then, it holds that

$$\operatorname{vec}(B(x)) = \Phi(x) + \varepsilon_\Phi(x), \tag{3.9}$$

with $\varepsilon_\Phi : \Omega \to \mathbb{R}^{pq}$.

Definition 3.1 (Vectorization). *Given a matrix $A \in \mathbb{R}^{p\times q}$ and having $A^{(i)} \in \mathbb{R}^p$, with $i \in \{1, 2, \dots, q\}$, denoting its i-th column, then $\mathrm{vec}\,(A) \in \mathbb{R}^{pq}$ is a column vector defined as*

$$\mathrm{vec}\,(A) = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(q)} \end{bmatrix}. \tag{3.10}$$

Moreover, if $a \in \mathbb{R}^n$ is a vector, then $a \equiv \mathrm{vec}\,(a)$.

For example, in the case $p = 2$ and $q = 3$, one has that

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \quad \mathrm{vec}\,(A) = \begin{bmatrix} a & d & b & e & c & f \end{bmatrix}^\top.$$

The expression in (3.7) can be equivalently expressed as

$$B(x) = \mathrm{vec}^{-1}\left(\Phi(x) + \varepsilon_\Phi(x)\right), \tag{3.11}$$

with $\mathrm{vec}^{-1}(\cdot)$ being the inverse of the vectorization operator. Before introducing this last one, the definition of the so-called *Kronecker product* must be recalled.

Definition 3.2 (Kronecker product). *Given two matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times q}$, the Kronecker product, denoted with $A \otimes B$, is a defined as*

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,m-1}B & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,m-1}B & a_{2,m}B \\ \dots & \dots & \dots & \dots & \dots \\ a_{n-1,1}B & a_{n-1,2}B & \dots & a_{n-1,m-1}B & a_{n-1,m}B \\ a_{n,1}B & a_{n,2}B & \dots & a_{n,m-1}B & a_{n,m}B \end{bmatrix} \in \mathbb{R}^{np\times mq} \tag{3.12}$$

Definition 3.3 (Vectorization inverse). *Given a vector $a \in \mathbb{R}^{pq}$, then, $\mathrm{vec}^{-1}(a) \in \mathbb{R}^{p\times q}$ is the matrix obtained performing the inverse of the vectorization operation, defined as*

$$\mathrm{vec}^{-1}(a) = (\mathrm{vec}\,(I_q)^\top \otimes I_p)(I_p \otimes a) \in \mathbb{R}^{p\times q}. \tag{3.13}$$

Moreover, it holds that $\mathrm{vec}\left(\mathrm{vec}^{-1}(a)\right) = a$.

For example, if $a = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^\top$, then result of $\mathrm{vec}^{-1}(a)$ choosing $p = 3$ and $q = 2$ is

$$\mathrm{vec}^{-1}(a) = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}.$$

Exploiting the formulation in (3.11) allows to express the estimate of a specific column of $B(x)$, denoted as $B^{(i)}(x) \in \mathbb{R}^p$, as

$$B^{(i)}(x) = \text{vec}^{-1}\left(\Phi(x)\right)^{(i)} + \text{vec}^{-1}\left(\varepsilon_\Phi(x)\right)^{(i)} \tag{3.14}$$

for $i \in \{1, 2, \dots, q\}$. Recalling the fact that $V_k \in \mathbb{R}^{(L_k-1)\times(L_{k+1}-1)}$, with $L_{k+1} - 1 = pq$ being the output size of Φ, one can conveniently express V_k as the concatenation of different sub-matrices $V_k^{[i]} \in \mathbb{R}^{(L_k-1)\times p}$, i.e.,

$$V_k = \begin{bmatrix} V_k^{[1]} & V_k^{[2]} & \cdots & V_k^{[q]} \end{bmatrix}. \tag{3.15}$$

Then, the expression of $\text{vec}^{-1}\left(\Phi(x)\right)^{(i)} \in \mathbb{R}^p$ is given by

$$\text{vec}^{-1}\left(\Phi(x)\right)^{(i)} = \text{vec}^{-1}\left(\Phi_k\right)^{(i)} = V_k^{[i]}\phi_k(\Phi_{k-1}). \tag{3.16}$$

To better understand the expression (3.16), a graphical representation of a network with $k = 2$ that estimates a 2×2 is presented in Figure 3.4. From such a depiction, it is possible to understand that one can isolate the estimate of the i-th column by performing the standard forward-pass up to layer $j = k - 1$ and then consider only the subset of weights $V_k^{[i]}$ for the last layer.

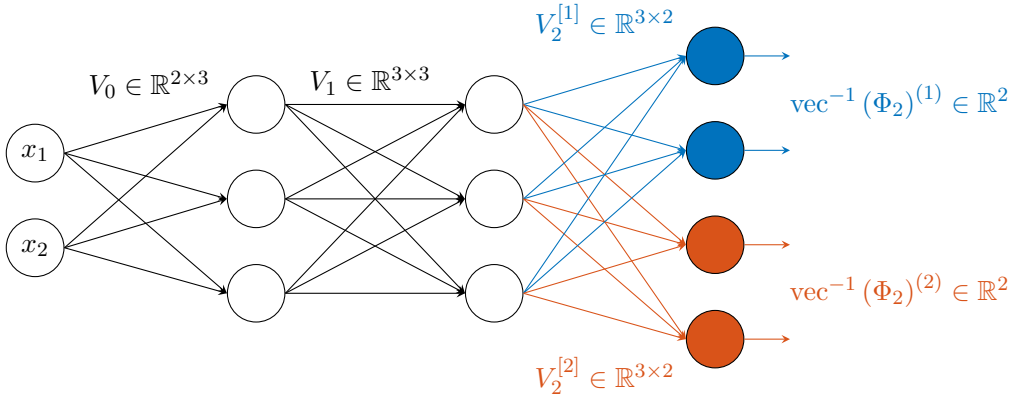


Figure 3.4: Example of a network with $k = 2$ hidden layers that approximates a 2×2 matrix.

3.1.3 Depth vs Width

One of the most critical aspects in the design of an ANN is the choice of the number of hidden layers k, namely the *depth*, and the number of neurons for each layer $j \in \{0, 1, \dots, k\}$, i.e., the *width*.

In the last decades, it has been studied how, from an approximation capability point of view, increasing the number of layers brings more benefits than increasing

the width of the ANN. Indeed, Håstad laid out the foundation for a proof in [5] considering logical circuits, which has been extended to ANNs by Montufar in [58]. From Theorem 3.1, the degree of approximation that a ANN can provide grows together with the number of neurons. Considering a *shallow* network, i.e., a ANN with $k = 1$, then the number of neurons tells how many times the approximator can change its slope. What Håstad did was to prove that, while in shallow ANNs such number grows polynomial with width, in DNNs, i.e., ANNs with $k \in \mathbb{N}_{\geq 2}$, it grows exponentially with depth.

More formally, consider a generic ANN $\Phi : \mathbb{R}^n \to \mathbb{R}$ with $k \in \mathbb{N}_{\geq 1}$ hidden layers, input vector $x \in \mathbb{R}^n$, $L_j = L$ for $i \in \{0, 1, \dots, k\}$, activation functions $\phi(\cdot) = \text{ReLU}(\cdot)$. Then, Φ has that the number of times N that the piece-wise affine approximation of the target function given by the ANN can change its slope, is given in terms of the following big-$\mathcal{O}$ and big-Ω notations based quantities

$$N = \mathcal{O}\left(2^k\right), \; N = \Omega\left(\left(\frac{L}{n}\right)^{(k-1)n} L^n\right). \tag{3.17}$$

This means that, in a shallow ANN, since $k = 1$, one has $N = \mathcal{O}(1)$ and $N = \Omega\left(L^n\right)$, i.e. polynomial. On the other hand, a deeper architecture ($k > 1$) ensures an exponential behavior of N, meaning that a DNN is more effective in terms of approximation capabilities.

In the rest of the dissertation, neural networks with MLP structure with $k \in \mathbb{N}_{\geq 2}$ hidden layers, i.e., DNNs, are considered.

3.1.4 Learning the weights

Once the structure of the neural network Φ has been chosen, a crucial operation is to find the optimal value of the weights V_j, for each layer $j \in \{0, 1, \dots, k\}$. This correspond to find the value of V_j that maximizes a given performance index. Such a procedure is often referred to as *learning* or *training*.
During the years, several learning algorithms have been proposed. Usually, they exploits *datasets*, which are collection of input-output pairs in the form

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N_\mathcal{D}},$$

where $N_\mathcal{D} \in \mathbb{N}_{>1}$ is the size of the dataset, $x^{(i)} \in \Omega \in \mathbb{R}^n$ is the input, and $y \in \mathbb{R}^p$ is the output. Then, an optimization over the dataset $\mathcal{D}$ with respect to a certain *loss function* $L : \mathcal{D} \to \mathbb{R}$ is performed. Depending on the task, different functions are selected. For example, in regression problems, the *Mean Squared Error* (MSE)

function

$$L(\mathcal{D}) = \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} \left(y^{(i)} - \hat{\Phi}\left(x^{(i)}\right) \right)^2, \tag{3.18}$$

is selected, where $\hat{\Phi} : \Omega \to \mathbb{R}^p$ is the network Φ characterized by non-optimal weights.

Due to the presence of nonlinearities in the neural network, introduced by the activation functions, the optimization problem is highly nonlinear, preventing the development of closed form solutions. For this reason, iterative methods, like *Gradient Descent* (GD) and its variants [59], must be adopted. Such techniques iteratively update the DNN parameters, i.e., weights and biases, relying on a low-order approximation of the behavior of the loss function, i.e.,

$$\begin{aligned} L(\mathcal{D}, \theta) \approx L\left(\mathcal{D}, \theta^{(t-1)}\right) + \nabla_\theta L\left(\mathcal{D}, \theta^{(t-1)}\right)\left(\theta - \theta^{(t-1)}\right) + \\ + \frac{1}{2}\left(\theta - \theta^{(t-1)}\right)^\top \mathcal{H}_L\left(\mathcal{D}, \theta^{(t-1)}\right)\left(\theta - \theta^{(t-1)}\right), \end{aligned}$$

where θ and $\theta^{(t-1)}$ represent the current value and the value at the previous time instant, respectively, while $\nabla_\theta L(\mathcal{D}, \theta^{(t-1)})$ and $\mathcal{H}_L\left(\mathcal{D}, \theta^{(t-1)}\right)$ are the gradient and the Hessian matrix of the loss, both computed in $\theta^{(t-1)}$.

Depending on the chosen algorithm, a different grade of approximation is employed. For example the *backpropagation* algorithm [60] requires the computation of the loss function gradient with respect to the parameters, while the *Adam* optimizer [61] relies on an estimation of the Hessian matrix. Let $\theta^{(k)}$ be the neural network parameters at a specific iteration, then an classical GD step with backpropagation is given by

$$\theta^{(k)} = \theta^{(k-1)} - \eta \nabla_\theta L\left(\mathcal{D}, \theta^{(k-1)}\right),$$

where $\eta \in \mathbb{R}_{>0}$ is the so-called *learning rate*, which is an hyperparameter whose tuning is fundamental for the convergence of the method. The size of η is dictated by the Lipschitz constant κ of the loss function, given by

$$\|L(\mathcal{D}, \theta_1) - L(\mathcal{D}, \theta_2)\| < \kappa \|\theta_1 - \theta_2\|.$$

However, the analytical expression of the loss function is unavailable because the underlying model is unknown. Hence, κ is not computable and, as a consequence, the learning rate η is chosen empirically small.

The optimization of the neural network weights is usually referred to as *training phase*. Once it is concluded, it is possible to *validate* the neural network model with respect to different figures of merit, e.g., MSE or objective criteria like the ones in

[62] and [63], calculated over a dataset which is different from the one used for the training. This allows to understand how well the network performs over unforeseen data. Once it is validated, the neural network is used for the intended scope during the so-called *inference phase*.

The learning methodology described above is usually referred to as *Supervised Learning* [64] and it is employed in many domains. However, there are cases in which data are not available a priori and hence it is not possible to build a structured training dataset. In such scenarios, and always depending on the task that must be performed, the learning procedure can be performed in several ways. For this reason, *learning-by-doing* approaches are employed. One of these approach is the so called *Reinforcement Learning* [65], whose preliminary concepts are recalled in the next section.

3.2 Reinforcement Learning

As stated at the end of the previous section, Reinforcement Learning (RL) [66] is a branch of machine learning that does not require dataset of pre-collected data. In the RL framework, represented graphically in Figure 3.5, there are two main entities, i.e., the *agent* and the *environment*. The former corresponds to the entity which has to take decisions, while the latter represents everything with which the agent interacts. At any given time $t \in \mathbb{R}_{\geq 0}$, the agent observes the environment, represented by a *state* $s_t \in \mathcal{S}$, with $\mathcal{S}$ being the so-called state space, and performs a certain action $a_t \in \mathcal{A}$, with $\mathcal{A}$ being the action space, according to a *policy* $\pi(a_t|s_t)$. The performed action changes the environment state. As soon as the state is changed, the agent receives an *instantaneous reward* $r_t \in \mathcal{R}$, i.e., a scalar which indicates how well the agent has performed at time t. Usually, the learning process is divided into *episodes*, during which the agent interacts with the environment until a complete attempt to perform the task is done or a fixed number of iterations is performed.

What made RL popular is its flexibility, since the action space $\mathcal{A}$ and the state space $\mathcal{S}$ are chosen entirely by the designer depending on the task. For example they can have continuous or discrete nature, with different degrees of complexity. As for the policy π, it can be continuous or discrete, deterministic or probabilistic.

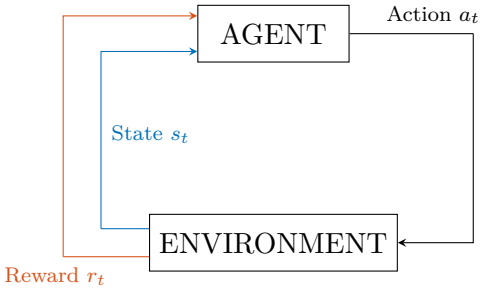


Figure 3.5: Graphical representation of the reinforcement learning framework.

3.2.1 Key concepts

In order to make the reader more confident with the key concepts related to RL, these last one are recalled and discussed, relying on the theory and notation introduced in [66].

Reward One of the main concepts in the RL framework is the so-called reward. At each time step t, the agent receives an instantaneous reward r_t which gives information on "how well" it performed in that specific time instant. The main goal of the agent is to maximize a quantity which is not the instantaneous reward, but rather the so-called (expected) *cumulative reward*, defined as

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}, \tag{3.19}$$

where $\gamma \in [0, 1]$ is the *discount rate*, which prioritizes earlier rewards. In particular, if γ is chosen close to zero, the agent will tend to perform actions that maximize immediate rewards, while having γ close to one will lead to the maximization of long term reward. Note that, (3.19) can be conveniently expressed in a recursive way as

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots + \gamma^T r_{t+T+1} \\ &= r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \cdots + \gamma^{T-1} r_{t+T+1} \right) \\ &= r_{t+1} + \gamma R_{t+1}. \end{aligned} \tag{3.20}$$

Markov Decision Process Each reinforcement learning problem is modeled as a Markov Decision Process (MDP), meaning that it satisfies the so-called *Markov property* given by

$$P(s_{t+1}, r_{t+1} | s_t, a_t) = P(s_{t+1}, r_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0), \tag{3.21}$$

where $P(\cdot|\cdot)$ denotes the conditional probability function. In other words, the environment state at time $t+1$ depends only on the state and action at time t. Such a property is also referred to c *memorylessness* property. A MDP is defined by the tuple

$$\langle \mathcal{S}, \mathcal{A}, r, p, \gamma \rangle, \tag{3.22}$$

where $\mathcal{S}$ and $\mathcal{A}$ are, respectively, the state and action spaces, γ is the discount rate in (3.19), and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, it is the expected reward given the current state s_t, action a_t, and the next state s_{t+1}. As for the function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, it is the so-called *dynamics* of the MDP, defined as the probability distribution of each possible successive state $s' \in \mathcal{S}$ and reward $r \in \mathcal{R}$, defined as

$$p(s', r|s, a) = P(s_t = s', r_t = r|s_{t-1} = s, a_{t-1} = a), \tag{3.23}$$

with

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \tag{3.24}$$

for all $s \in \mathcal{S}$, $a \in \mathcal{A}$. From the dynamics of the MDP, it is possible to retrieve several information about the MDP. For example, the *state-transition probabilities* $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ are given by

$$p(s'|s, a) = P(s_t = s'|s_{t-1} = s, a_{t-1} = a) = \sum_{r \in \mathcal{R}} p(s', r|s, a). \tag{3.25}$$

Moreover, it is possible to obtain the expected reward for a state action pair, defined by the function $r : \mathcal{S}, \mathcal{A} \to \mathbb{R}$

$$r(s, a) = \mathbb{E}[r_t|s_{t-1} = s, a_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a), \tag{3.26}$$

and the expected reward appearing in (3.22) as the three-argument function

$$r(s, a, s') = \mathbb{E}[r_t|s_{t-1} = s, a_{t-1} = a, s_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)}, \tag{3.27}$$

where the numerator and denominator of the fraction are computed as in (3.23) and (3.25), respectively.

Policy The policy π defines the behavior of the agent interacting with the environment. In particular $\pi(a, s)$ denotes the probability that the agent performs an action $a \in \mathcal{A}$ when it is in a state $s \in \mathcal{S}$. Depending on the nature of the task, the policy can be represented in different ways, e.g., a look-up table, a function, or even a neural network. Moreover, it can be either deterministic or probabilistic.

Value functions The expected cumulative reward that the agent can gain starting from a state s and following the policy π thereafter is the so-called *state-value function* $V^\pi : \mathcal{S} \to \mathbb{R}$, computed as

$$V^\pi(s) = \mathbb{E}_\pi[R_t|s_t = t] = \mathbb{E}_\pi\left[\sum_{k=0}^{T} \gamma^k r_{t+k+1}|s_t = s\right].$$

For any policy π and any state s, there is a relation between the value of s and the value of its possible successor states. Such a condition is given by

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi[R_t|s_t = s] \\ &= \mathbb{E}_\pi\left[r_{t+1} + \gamma R_{t+1}|s_t = s\right] \\ &= \sum_{a\in\mathcal{A}} \pi(s,a) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r|s,a)\Big[r + \gamma \mathbb{E}_\pi[R_{t+1}|s_{t+1} = s']\Big] \\ &= \sum_{a\in\mathcal{A}} \pi(s,a) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s',r|s,a)\Big[r + \gamma V^\pi(s')\Big], \end{aligned} \tag{3.28}$$

where the last equation is the so-called *Bellman equation* for V^π, which expresses a relationship between the value of a state and the values of its successor states.

Another quantity which is fundamental in the RL settings is the so-called *action-value function* $Q^\pi : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, which denotes the expected cumulative reward when the agent is in a state s, takes an action a and follows the policy π afterwards. Such a function is given by

$$Q^\pi(s,a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{T} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right]. \tag{3.29}$$

The idea behind reinforcement learning is to find a policy that maximizes the reward on the long run. In general, given two policies π_1 and π_2, it is possible to define that one is better than the other by comparing their value functions. In particular, π_1 is better than π_2 if and only if $V^{\pi_1}(s) \geq V^{\pi_2}(s)$, for every state $s \in \mathcal{S}$. In general, for every RL problem there exists a policy $\pi_\star$ which is better or equal to all other policies. Such a policy is referred to as *optimal policy* and it is characterized by the *optimal value function*

$$V^\star(s) = \max_\pi V^\pi(s), \tag{3.30}$$

for all $s \in \mathcal{S}$. Similarly, it is possible to define the *optimal action-value function* defined as

$$Q^\star(s,a) = \max_\pi Q^\pi(s,a), \tag{3.31}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Moreover, the above equation can be rewritten so that the optimal action-value function is expressed in terms of the optimal state-value function, i.e.,

$$Q^\star(s,a) = \mathbb{E}[r_{t+1} + \gamma V^\star(s_{t+1}) | s_t = s, a_t = a] \tag{3.32}$$

Similarly to what done for a general policy π, it is possible to express, for all $s \in \mathcal{S}$, the Bellman equation for V^*, also called *Bellman optimality equation*, defined as

$$\begin{aligned} V^\star(s) &= \max_{a \in \mathcal{A}} Q^{\pi_\star}(s,a) \\ &= \max_{a \in \mathcal{A}} \mathbb{E}_{\pi_\star}[R_t | s_t = s, a_t = a] \\ &= \max_{a \in \mathcal{A}} \mathbb{E}_{\pi_\star}[r_{t+1} + \gamma R_{t+1} | s_t = s, a_t = a] \\ &= \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a)[r + \gamma V^\star(s')], \end{aligned} \tag{3.33}$$

and representing the fact that the value of a state s under the optimal policy $\pi_\star$ is equal to the expected return of the best action from that state. The Bellman optimality condition can be computed also for $Q^\star$ as

$$\begin{aligned} Q^\star(s,a) &= \mathbb{E}\Big[r_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q^\star(s_{t+1}, a') | s_t = s, a_t = a\Big] \\ &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a)\Big[r + \gamma \max_{a' \in \mathcal{A}} Q^\star(s', a')\Big], \end{aligned} \tag{3.34}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

Having access to $Q^\star$ implies that finding optimal actions is very easy. In particular, if the agent is in a state s, it can simply find the action $a^\star$ such that

$$a^\star = \max_{a \in \mathcal{A}} Q^\star(s,a). \tag{3.35}$$

3.2.2 Q-learning

In the majority of real-world scenarios, the complete transition probability $p(s', r|s, a)$ is not available. As a result , it is not possible to have access to the optimal action-value function $Q^\star$. To overcome the issue, *Q-learning* has been introduced [67]. The aim behind the algorithm is to provide an estimate of $Q^\star$ starting from a random guess independently from the policy being applied and without requiring the knowledge transition probability of the environment. For this last reason, it belongs to the class of *model-free* algorithms. The Q-learning algorithm is presented in Algorithm 1. The required inputs are the learning rate α and the discount factor γ. The following theorem can be introduced.

Theorem 3.2 (Ideal convergence of the action-value function [66]). *Under the assumption that each state is visited infinitely often and action is performed infinitely often. Then, if* $\alpha \to 0$ *it holds that*

$$\hat{Q}(s,a) \to Q^\star(s,a),$$

with probability 1, for all $s \in \mathcal{S}$ *and* $a \in \mathcal{A}$.

In other words, it is guaranteed that the approximate of the action-value function $\hat{Q}$ converge to its optimal value $Q^\star$ if an infinite number of steps is performed.

The Q-learning algorithm presented in Algorithm 1 is suitable when the structure and size of the state and action spaces allow to express the optimal action value function as a table similar to the one presented in Table 3.1. However, in some applications, e.g., robotics, the dimension of the state and action spaces can be too large or these last ones may be even continuous. In such situations, an alternative way is necessary.

	A_1	A_2	$\dots$	A_m
S_1	$Q^\star(S_1, A_1)$	$Q^\star(S_1, A_2)$	$\dots$	$Q^\star(S_1, A_m)$
S_2	$Q^\star(S_2, A_1)$	$Q^\star(S_2, A_2)$	$\dots$	$Q^\star(S_2, A_m)$
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
S_n	$Q^\star(S_n, A_1)$	$Q^\star(S_n, A_2)$	$\dots$	$Q^\star(S_n, A_m)$

Table 3.1: Tabular representation of the optimal action value function in the case of discrete state and action spaces $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ and $\mathcal{A} = \{A_1, A_2, \dots, A_m\}$.

3.2.3 Deep Q-learning

As anticipated at the end of Section 3.2.2, in the cases in which it is not possible to express the action-value function in a tabular form, e.g., the size of the state and action spaces is very large, an alternative is required. Thanks to their universal approximation properties, DNNs like the ones presented in Section 3.1 can be used as parametric approximators for the optimal action-value function. In particular, given a network $\hat{\Phi}$ characterized by estimated weight and biases $\hat{V}_j$, with $j \in \{0, 1, \dots, k\}$, collected in a term θ, then one has

$$\hat{Q}(s, a, \theta) = \hat{\Phi}(x), \tag{3.36}$$

with $x = \begin{bmatrix} s^\top & a^\top \end{bmatrix}^\top$, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The goal would be to learn the matrices $\hat{V}_j$ in θ so that the map $\hat{Q}$ is close as close as possible its optimal counterpart.

Algorithm 1 Q-learning algorithm [67]

Require: Learning rate α, discount factor γ

Randomly initialize the estimate $\hat{Q}(s,a)$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}$

Set $\hat{Q}(\bar{s},a) = 0$, for all, $a \in \mathcal{A}$ ▷ $\bar{s}$ is the terminal state

for each episode **do**

Initialize state s_t

for each step t of the episode **do**

$$a_t = \begin{cases} \text{argmax}_{a\in\mathcal{A}}\, \hat{Q}(s_t,a) & \text{with probability } (1-\varepsilon) \\ \text{random from } \mathcal{A} & \text{with probability } \varepsilon \end{cases}$$ ▷ ε-greedy policy

Take action a_t, observe r_t and s'

$\hat{Q}(s_t,a_t) = \hat{Q}(s_t,a_t) + \alpha\left[r_t + \gamma \max_{a\in\mathcal{A}} \hat{Q}(s',a) - \hat{Q}(s_t,a_t)\right]$ ▷ Q-update

$s_t = s'$ ▷ state update

end for

end for

When DNNs are used while solving a Q-learning problem, this last one is referred to as a *Deep Q-learning* problem and the general framework is called Deep Reinforcement Learning (DRL). One of the main algorithms is the one proposed in [68], which aims to update the DNN parameters θ such that the loss function

$$L(\theta) = \mathbb{E}\left[\left(\hat{Q}(s_t,a_t|\theta) - y_t\right)^2\right] \tag{3.37}$$

is minimized, where y_t is the so-called target function given by

$$y_t = r(s_t,a_t) + \gamma \hat{Q}(s_{t+1}, \hat{\mu}(s_{t+1})|\theta), \tag{3.38}$$

with $\hat{\mu}$ being defined so that

$$\hat{\mu}(s_t) = \underset{a\in\mathcal{A}}{\text{argmax}}\, \hat{Q}(s_t,a|\theta). \tag{3.39}$$

The update of the parameters can be done, for example, exploiting the techniques described in Section 3.1.4. For the complete explanation of the algorithm, the reader is invited to refer to [68].

When dealing with RL problems, one of the key aspect is the *exploration-exploitation* dilemma, which refers to the two possible strategies that the agent should balance during the learning. In fact, the agent could *exploit* past (incomplete and possibly misleading) experience in order to chose the best option or *explore* the environment choosing new options to improve knowledge about it. To balance exploration and exploitation in Algorithm 1, *ε-greedy* strategy is implemented: with

a probability $1-\varepsilon$ the best action (according to the current knowledge) is selected, while with probability ε the agent chooses a random action.

3.2.4 Actor-Critic

The Q-learning algorithm and the other methodologies that aim to estimate a value function are also referred to as *critic-only* methods. Once a sufficiently accurate estimate for the value function $\hat{Q}$ is provided, actions are chosen according to a deterministic policy π, defined as

$$\pi(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \hat{Q}(s, a). \tag{3.40}$$

Since the convergence results in Theorem 3.2 are valid under the strict assumption that each state is visited infinitely often, they are not applicable in general (consider, for example, the case in which the state space $\mathcal{S}$ is continuous [69]).

Another class of algorithms for solving RL problems is the one represented by the so-called *actor-only* methods (also known as *policy gradient* methods), whose aim is to provide a direct estimate of the policy, without relying on any form of stored value function. Examples of algorithms belonging to such a class are the one proposed in [70] and REINFORCE, introduced in [71]. In policy gradient methods, the aim is to find a parameterized policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$, where θ is the set of parameters. In the case in which the policy is represented by a DNN like the one in Section 3.1, then the parameters are the weights and the biases of the network. Starting from an initial guess, the value of these parameters is adapted in the direction of the gradient of a cost function which can be, for example the *average return* [72]. An advantage of actor-only methods is that convergence is guaranteed under some condition of the estimated gradients and on the learning rates [73]. The main drawback is that, as studied in [73], the estimated gradients may have large variance. Moreover, gradients are computed without using any knowledge of the past estimates.

In order to combine the benefits of the two classes, *Actor-Critic* methods [74] have been developed. Such methods can be schematized as in Figure 3.6 and their behavior summarized as follows. The actor, characterized by a set of parameters θ_π, prescribes a policy, which is evaluated by the critic, characterized by parameters θ_Q, using *policy evaluation* methods (e.g., TD(λ) [75] or LSTD [76]). Collecting samples from the environment, the critic parameters θ_Q, and hence the value function, are updated as in critic-only methods. Note that, in contrast to critic-only method, actions are not selected solving a problem like (3.40). Instead, the actor parameters θ_π, and hence the policy, are updated along the policy gradient direction using a

small step size. The main benefit of actor-critic methods is that they can approximate policies that deal with continuous action spaces, while having low variance in the policy gradient thanks to the introduction of the critic. In the following, two actor-critics methods are briefly described.

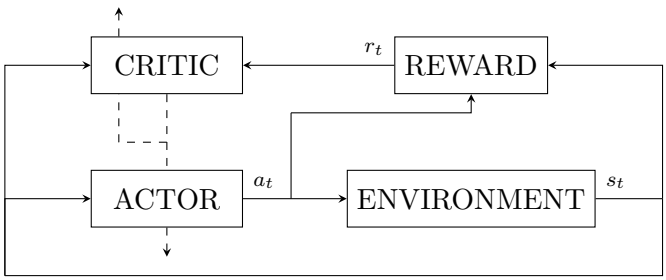


Figure 3.6: Block diagram of a generic actor-critic algorithm.

Deep Deterministic Policy Gradient

The Deep Deterministic Policy Gradient (DDPG) is an actor-critic algorithm capable of handling continuous state and action spaces, being applicable to domains such as dynamical systems control. The algorithm, described in Algorithm 2, composed by the following elements:

- an **actor neural network** μ_θ, characterized parameters θ;
- a **critic neural network** Q_ϕ, characterized by parameters ϕ;
- a couple of **target networks** $\mu_{\theta'}$ and $Q_{\phi'}$, characterized by parameters θ' and ϕ', respectively;
- a **replay buffer** $\mathcal{D}_{\text{rb}}$ which stores past experience.

The target networks are copies of their original counterpart, but their parameters are updated with a *soft update*. In particular, given a DNN characterized by parameters θ and its target counterpart with parameters θ', then the soft update of the target network is given by

$$\theta' = \tau\theta' + (1-\tau)\theta, \tag{3.41}$$

with $\tau \in [0,1]$ being an hyper-parameter. The use of target networks with the above updating has been shown to provide a more stable learning (the reader is invited to refer to [77] for mote details).

As for the replay buffer $\mathcal{D}_{\text{rb}}$, it is introduced to exploit the concept of *Experience Replay* [78]. In particular, each tuple $\langle s_t, a_t, r_t, s_{t+1}, d\rangle$, with $d \in \{0,1\}$ indicating

if the state s_{t+1} is a terminal state, is stored in $\mathcal{D}_{\text{rb}}$. Then, learning is applied on a batch of experience tuples randomly sampled from $\mathcal{D}_{\text{rb}}$. This allows to improve data efficiency and the quality of the learning process [77].

As visible from Algorithm 2, in order to enforce an exploratory behavior, a noise ϵ sampled from the normal distribution $\mathcal{N}$ is added to the action chosen by the policy. Then, to make sure that the overall action is between the bounds of the action space $\mathcal{A}$, a saturation operation is performed. Specifically,

$$a_t = \text{clip}(\mu_\theta + \epsilon, \bar{a}, \underline{a}),$$

where $\bar{a}$ and $\underline{a}$ represent the upper and lower bound of the action space $\mathcal{A}$, respectively.

Algorithm 2 Deep Deterministic Policy Gradient [77]

Require: Soft-update rate τ, discount factor γ

Randomly initialize the parameters θ and ϕ

Initialize the target networks $\theta' \leftarrow \theta$ and $\phi' \leftarrow \phi$

for each training episode **do**

Initialize state s_t

for each step t of the episode **do**

Take action $a_t = \text{clip}(\mu_\theta + \epsilon, \bar{a}, \underline{a})$ ▷ Add noise $\epsilon \sim \mathcal{N}$ for exploration

Observe s', r_t and d ▷ $d \in \{0, 1\}$, $d = 1$ if s' is terminal

Store (s_t, a_t, r_t, s', d) in $\mathcal{D}_{\text{rb}}$ ▷ Populate replay buffer

Randomly sample batch of N transitions $\mathcal{B}$ from $\mathcal{D}_{\text{rb}}$

$y_t(r_t, s', d) = r + \gamma(1 - d)Q_{\phi'}(s', \mu_{\theta'}(s'))$ ▷ Compute the targets

Update critic with gradient descent

$$\nabla_\phi \frac{1}{N} \sum_{(s,a,r,s',d)\in\mathcal{B}} (Q_\phi(s, a) - y(r, s', d))^2$$

Update actor with gradient ascent

$$\nabla_\theta \frac{1}{N} \sum_{(s,a,r,s',d)\in\mathcal{B}} Q_\phi(s, \mu_\theta(s))$$

Target actor soft update $\theta' \leftarrow \tau\theta' + (1 - \tau)\theta$

Target critic soft update $\phi' \leftarrow \tau\phi' + (1 - \tau)\phi$

$s_t = s'$ ▷ State update

end for

end for

Twin-Delayed DDPG

Despite the good results provided by DDPG algorithm, this last one is very likely to fail due to its sensitiveness to hyperparameters tuning. A common failure mode is that the learned Q function overestimates the optimal value function, leading to a a sub-optimal policy. For this reason, Twin-Delayed DDPG (TD3) has been developed [79]. differently from the original DDPG proposed in [77], three different modifications are introduced:

1. two critic networks, namely, Q_{ϕ_1} and Q_{ϕ_2}, approximate two different Q functions (this is the reason behind the "twin");

2. the policy and the target networks are updated less frequently than the critic networks (this is the reason behind the "delayed");

3. noise is added to the target action, making it harder for the policy to exploit Q function errors (this is referred to as *target policy smoothing*).

The TD3 algorithm is presented in Algorithm 3 and its most important components are described hereafter, starting from the target policy smoothing. Differently from DDPG, the actions used to build the Q-learning target are not based on the policy μ_θ, but on its target counterpart $\mu_{\theta'}$. Moreover, a clipped noise is added one each dimension of the action, resulting in

$$a'(s') = \text{clip}\left(\mu_{\theta'}(s') + \text{clip}(\epsilon, -c, c), \bar{a}, \underline{a}\right), \tag{3.42}$$

with ϵ being sampled from the normal distribution $\mathcal{N}$ and c being an hyperparameter. Smoothing the estimate of the Q function over similar actions, target policy smoothing serves as a regularizer for the learning algorithm, avoiding that the policy exploits incorrect value peaks estimated by the Q function approximator.

Then, the two critics use a single target, computed using the Q function which gives the smaller target value, i.e.,

$$y(r, s', d) = r + \gamma(1 - d) \min_{i \in \{1,2\}} Q_{\phi_i'}(s', a'(s')),$$

with $a'(s')$ as in (3.42). Such a target is then used to compute the loss functions that will be used for the optimization of the critic networks parameters. Such functions are defined as

$$L(\phi_i, \mathcal{D}_{\text{rb}}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}_{\text{rb}}}\left[\left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2\right],$$

with $i \in \{1, 2\}$. Employing the smaller value for the target (3.42) and regressing toward that counteracts the overestimation of the value function.

Finally, the actor is updated by maximizing Q_{ϕ_1}, similarly to what is done in the DDPG. However, in TD3 the policy is updated less frequently. In general the policy and the target networks are updated once every two updates of the critic networks.

Algorithm 3 Twin Delayed DDPG [79]

Require: Soft-update rate τ, discount factor γ

Randomly initialize the parameters θ, ϕ_1, and ϕ_2

Initialize the target networks $\theta' \leftarrow \theta$, $\phi'_1 \leftarrow \phi_1$, and $\phi'_2 \leftarrow \phi_2$

for each training episode **do**

Initialize state s_t

for each step t of the episode **do**

Take action $a_t = \text{clip}(\mu_\theta + \epsilon, \bar{a}, \underline{a})$ ▷ Add noise $\epsilon \sim \mathcal{N}$ for exploration

Observe s', r_t and d ▷ $d \in \{0, 1\}$, $d = 1$ if s' is terminal

Store (s_t, a_t, r_t, s', d) in $\mathcal{D}_{\text{rb}}$ ▷ Populate replay buffer

Randomly sample batch of N transitions $\mathcal{B}$ from $\mathcal{D}_{\text{rb}}$

$a'(s') = \text{clip}\left(\mu_{\theta'}(s') + \text{clip}(\epsilon, -c, c), \bar{a}, \underline{a}\right)$ with $\epsilon \sim \mathcal{N}$ ▷ Target actions

$y(r, s', d) = r + \gamma(1 - d) \min_{i \in \{1,2\}} Q_{\phi'_i}(s', a'(s'))$ ▷ Compute the targets

Update the critics with gradient descent

$$\nabla_\phi \frac{1}{N} \sum_{(s,a,r,s',d) \in \mathcal{B}} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \text{ for } i \in \{1, 2\}$$

if time for delayed update **then**

Update the actor with gradient ascent

$$\nabla_\theta \frac{1}{N} \sum_{(s,a,r,s',d) \in \mathcal{B}} Q_{\phi_1}(s, \mu_\theta(s))$$

Target actor soft update $\theta' \leftarrow \tau\theta' + (1 - \tau)\theta$

Target critics soft update $\phi'_i \leftarrow \tau\phi'_i + (1 - \tau)\phi_i$ with $i \in \{1, 2\}$

end if

$s_t = s'$ ▷ State update

end for

end for

Chapter 4

Preliminaries on Robotics

The controllers and observers presented in this dissertation are employed in applications that involves open chain robotic manipulators. In this chapter, the basic concepts in the field of robotics, instrumental to the development of the techniques in this dissertation are introduced.

4.1 Basic Definitions

The term *robot* is used to refer to servo-mechanical system which is able to perform repetitive operations with different degree of complexity. A first classification distinguishes between *fixed* robots and *mobile* ones [80]. In particular, the former class contains all the robots that are somehow anchored to a surface, without the possibility of moving, while to the latter belongs all the robot that can freely move in the space (e.g., underwater vehicles, legged robots, humanoid robots, quadrotors).

The robots considered in this dissertation are the so-called manipulators, which belong to the class of fixed robot. In general, a manipulator is built as the combination of *joints*, *links*, and the *end-effector*. In particular:

- Joints are actuated components that gives mobility to the robot. Depending on the application, a joint can be *revolute*, *prismatic*, or *spherical* [24]. The first type allows a rotation around an axis, the second a translational movement along a single direction, while the third allows rotation around three non-coaxial axes intersecting a common point. Depending on the joint used, the robot is characterized by different Degrees of Freedom (DoF). For example, a robot with three revolute joints is characterized by three DoF.

- Links are the rigid elements that compose the robot structure, interconnected

through joints.

- The end-effector is the utensil used to interact with the surrounding environment. Depending on the task the robot must perform, several types of end-effectors can be mounted (e.g., gripper, torch, wielder, etc.), and these last ones can be made of soft or hard materials.

In this dissertation, the class of *open-chain* manipulators like the one in Figure 4.1, is considered.

Figure 4.1: Example of open-chain manipulator with seven revolute joints, eight links, and a gripper as end-effector.

4.2 Pose of a rigid body

Before presenting fundamental notions about robot modeling, it is important to introduce the concept of *pose* of a rigid body. Let $O_1 - x_1y_1z_1$ be an orthonormal reference frame attached to a fixed point in space. Then, consider a generic rigid body to which is attached a reference frame $O_2 - x_2y_2z_2$. Then, it is possible to define the pose of the rigid body as the combination of position and orientation of $O_2 - x_2y_2z_2$ with respect to $O_1 - x_1y_1z_1$.

The position can be simply described by the three dimensional vector $p_2^1 \in \mathbb{R}^3$

defined as

$$p_2^1 = \begin{bmatrix} p_{2,x}^1 & p_{2,y}^1 & p_{2,z}^1 \end{bmatrix}^\top,$$

where $p_{2,x}^1 \in \mathbb{R}$, $p_{2,y}^1 \in \mathbb{R}$, and $p_{2,z}^1 \in \mathbb{R}$ denote the position of O_2 with respect to x_1, y_1, and z_1, respectively.

Differently form the position, the orientation of a reference frame with respect to another can be expressed in different ways, such as *Euler angles*, *quaternions*, *rotation matrices*, and *axis-angle*. In the following, the last two of the above list are briefly introduced.

Rotation matrix A matrix R is said to be a rotation matrix if $R \in SO(3) \subset \mathbb{R}^{3\times3}$, where $SO(3)$, known as the *special orthogonal group*, contains all the matrices $R \in \mathbb{R}^{3\times3}$ such that $R^\top R = I_3$ and $\det(R) = 1$ [81].

Using a rotation matrix, the orientation of frame $O_2 - x_2y_2z_2$ with respect to frame $O_1 - x_1y_1z_1$ is given by

$$R_2^1 = \begin{bmatrix} x_2^1 & y_2^1 & z_2^1 \end{bmatrix},$$

where $x_2^1 \in \mathbb{R}^3$, $y_2^1 \in \mathbb{R}^3$, and $z_2^1 \in \mathbb{R}^3$ are the projection of the axes of O_2 over the axes of the base reference frame O_1. Moreover, the following properties hold:

1. Given a rotation matrix R_2^1, it is possible to express the orientation of $O_1 - x_1y_1z_1$ with respect to $O_2 - x_2y_2z_2$ as $R_1^2 = (R_2^1)^{-1} = (R_2^1)^\top$

2. Assume that there exists a rotation matrix R_1^0 that expresses the orientation of $O_1 - x_1y_1z_1$ with respect to frame $O_0 - x_0y_0z_0$. Then, it holds that $R_2^0 = R_1^0 R_2^1$. More in general, it holds that

$$R_n^0 = R_1^0 R_2^1 \cdots R_{n-1}^{n-2} R_n^{n-1}.$$

3. Given a vector $v_2 \in \mathbb{R}^3$, expressed with respect to reference frame O_2. Then, it is possible to find its representation with respect to O_1 as $v_1 = R_2^1 v_2$.

Axis-angle It is possible to represent a rotation between O_2 and O_1 by means of four parameters. In particular, it is possible to find a rotation axis $\hat{\omega} \in \mathbb{R}^3$, with $\|\hat{\omega}\| = 1$, and an angle $\alpha \in [0, \pi]$. Then, rotating O_1 of a quantity α around $\hat{\omega}$ would result in an overlapping of the two frames.

4.2.1 Change of orientation representation

It is possible, and often convenient, to change the way of representing a rotation, e.g., from rotation matrix to axis-angle and vice versa. For example, one can obtain a rotation matrix $R \in SO(3)$ associated with a rotation α around axis $\hat{\omega}$ according to the Rodrigues' formula [81]

$$R = \hat{\omega}\hat{\omega}^\top(1 - \cos(\alpha)) + I_3 \,\mathrm{skew}(\hat{\omega})\sin(\alpha), \tag{4.1}$$

where the operator $\mathrm{skew}(\cdot)$ is defined as follows.

Definition 4.1 (Skew operator). *Given a vector* $v \in \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^\top \in \mathbb{R}^3$*, then* $\mathrm{skew}(v) \in \mathbb{R}^{3\times 3}$ *is a matrix defined as*

$$\mathrm{skew}(v) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

which satisfies the skew-symmetric property $\mathrm{skew}(v)^\top = -\mathrm{skew}(v)$.

Vice versa, it is possible to translate a rotation matrix

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in SO(3)$$

into an axis $\hat{w}$ and an angle α, exploiting the algorithm in [81, Chapter 3], recalled for convenience in Algorithm 4.

4.3 Kinematics modeling

Consider an open-chain manipulator characterized by $n \in \mathbb{N}_{>0}$ DoF. Then, it is possible to define the vector of the joint variables $q \in \mathbb{R}^n$ as

$$q(t) = \begin{bmatrix} q_1(t) & q_2(t) & \cdots & q_n(t) \end{bmatrix}^\top,$$

which contains the value of each joint composing the the robot. In particular, if the i-th joint is of a revolute type, then $q_i \in \mathbb{R}$ is an angle expressed in radians, while if it is prismatic, it is a distance expressed in meters. The space in which the vector q is defined is the typically referred to as *joint space*.

Then, defining a base reference frame $O_b - x_b y_b z_b$ fixed in space and a reference frame $O_e - x_e y_e z_e$ attached to the end-effector of the robot, it is possible to express

Algorithm 4 Rotation matrix to Axis-Angle [81]

Require: A rotation matrix $R \in SO(3)$

if $R = I_3$ **then**

Set $\alpha = 0$ and $\hat{\omega} = 0_3$, since it is undefined

end if

if $\mathrm{tr}(R) = -1$ **then**

Set $\alpha = \pi$ and $\hat{\omega}$ equal to any of

$\hat{\omega} = \frac{1}{2\sqrt{2(1+r_{33})}} \begin{bmatrix} r_{13} \\ r_{23} \\ 1+r_{33} \end{bmatrix}$, $\hat{\omega} = \frac{1}{2\sqrt{2(1+r_{22})}} \begin{bmatrix} r_{12} \\ 1+r_{22} \\ r_{32} \end{bmatrix}$, $\hat{\omega} = \frac{1}{2\sqrt{2(1+r_{11})}} \begin{bmatrix} 1+r_{11} \\ r_{21} \\ r_{31} \end{bmatrix}$

that is a feasible solution.

else

Set $\alpha = \cos^{-1}\left(\frac{1}{2}(\mathrm{tr}(R) - 1)\right) \in [0, \pi)$

Compute $\mathrm{skew}(\hat{w}) = \frac{1}{2\sin(\alpha)}(R - R^\top)$ and, from that, extract $\hat{w}$

end if

the pose of this last one by means of the *homogeneous transformation matrix* $T_e^b \in \mathbb{R}^{4\times4}$. This last one, which depends on the vector q, is defined as

$$T_e^b(q) = \begin{bmatrix} R_e^b(q) & p_e^b(q) \\ 0_3^\top & 1 \end{bmatrix}, \tag{4.2}$$

where $p_e^b \in \mathbb{R}^3$ and $R_e^b \in SO(3)$ represent the position and orientation of O_e with respect to O_b, respectively. The space in which the pose of the end-effector is define is the so-called *operational space*.

When performing the kinematics modeling of a manipulator, it is possible to distinguish between *forward kinematics*, *differential kinematics*, and *inverse kinematics*, described in the following.

4.3.1 Forward Kinematics

The aim of *forward kinematics* is to compute the pose of the end-effector of the robot, given the vector q. In particular, if the pose is expressed as an homogeneous transformation matrix, the objective is to compute the mapping $T_e^b : \mathbb{R}^n \to \mathbb{R}^{4\times4}$.

Considering an open chain manipulator with n joints and $n+1$ links, if one attaches a reference frame to each link, then the pose of the end-effector with respect to the base frame can be computed as

$$T_e^b(q) = T_0^b A_1^0(q_1) A_2^1(q_2) \cdots A_n^{n-1}(q_n) T_e^n. \tag{4.3}$$

Since, in general, the homogeneous transformation matrices T_0^b and T_e^n are constant, forward kinematics aims to provide an expression of the transformation matrices $A_i^{i-1}(q_i)$, with $i \in \{1, 2, \dots n\}$, each of which is a function of a single joint variable.

To accomplish such an objective, it is possible to adopt the so-called DH convention [24], which gives a systematic method to assign the reference frames to the robot components, and then, after having identified some parameters, computes the value of the transformation matrices $A_i^{i-1}(q_i)$. For convenience, the DH methodology is briefly recalled in the following.

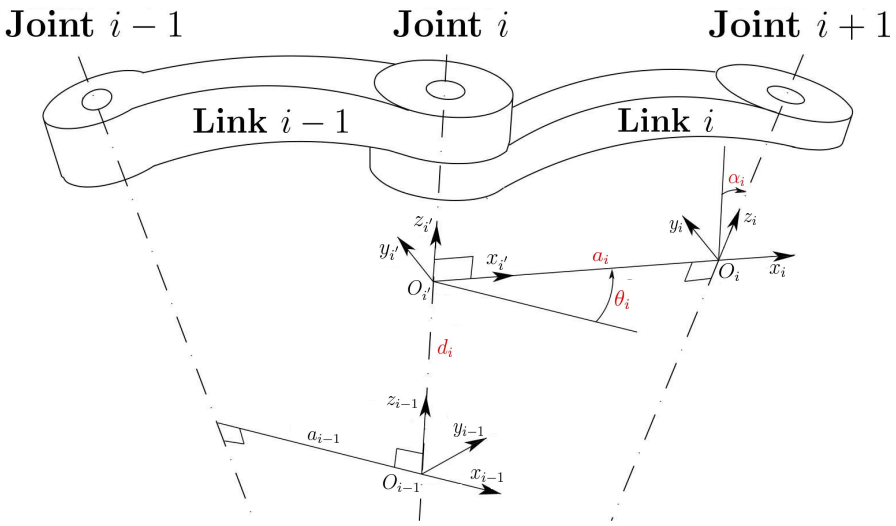


Figure 4.2: Graphical representation of the Denavit-Hartenberg convention, with the parameters highlighted in red.

Consider Figure 4.2, with joint i connecting link $i-1$ to link i. Then, according to the DH convention, the right-handed frame $O_i - x_i y_i z_i$ is defined following the following steps:

1. Chose axis z_i along the axis of joint $i+1$.

2. Place the origin O_i at the intersection of z_i with the common normal between z_{i-1} and z_i. Moreover, locate $O_{i'}$ at the intersection of the common normal with z_{i-1}.

3. Chose axis x_i along the common normal between z_{i-1} and z_i with positive direction from joint i to joint $i+1$.

4. Select axis y_i so that the right-handed frame is completed.

There are some cases in which the definition of $O_i - x_i y_i z_i$ is not unique. For a complete analysis of such conditions, one should refer to [24, Chapter 2].

Once the frames are defined, the pose of $O_i - x_i y_i z_i$ with respect to $O_{i-1} - x_{i-1} y_{i-1} z_{i-1}$ is specified by the four parameters $a_i, d_i, \alpha_i, \theta_i \in \mathbb{R}$. In particular, a_i is the distance between O_i and $O_{i'}$, d_i is the coordinate of $O_{i'}$ along axis z_{i-1}, α_i is the angle between z_{i-1} and z_i about axis x_i, negative when the rotation is clockwise, while θ_i is the angle between x_{i-1} and x_i about axis z_{i-1}, negative when the rotation is clockwise. The value of a_i and α_i is always constant, while the remaining two depends on the type of joint i. In particular, if it is a revolute joint, then d_i is constant and $\theta_i = q_i(t)$. On the other hand, if it is prismatic, then θ_i is constant and $d_i = q_i(t)$.

Finally, the transformation matrix $A_i^{i-1}(q_i)$ is computed as

$$A_i^{i-1}(q_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.4}$$

Before going further with the kinematics modeling, it is worth recalling that there exists also a *modified* DH convention, introduced in [82]. Differently from standard DH convention, the coordinates of the frame $O_i - x_i y_i z_i$ is put on axis of joint i and not $i+1$. Moreover, the transformation matrix $A_i^{i-1}(q_i)$ is computed as

$$A_i^{i-1}(q_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -d_i\sin(\alpha_{i-1}) \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & d_i\cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.5}$$

4.3.2 Differential Kinematics

Similarly to forward kinematics, the aim of *differential kinematics* is to provide a mapping from the joint space toward the operational space. However, the differential kinematics studies the relationship between *velocities*. In particular, given the vector of joint velocities $\dot{q} \in \mathbb{R}^n$, the objective is to discover how this last one influences the velocity of the end-effector frame $O_e - x_e y_e z_e$. Such a velocity is denoted as $v_e \in \mathbb{R}^6$ and given by

$$v_e = \begin{bmatrix} \dot{p}_e \\ w_e \end{bmatrix}, \tag{4.6}$$

where $\dot{p}_e \in \mathbb{R}^3$ and $w \in \mathbb{R}^3$ represent the linear and angular velocities, respectively. It is possible to put in relation v_e with $\dot{q}$ by means of

$$v_e = J(q)\dot{q}, \tag{4.7}$$

where $J(q) \in \mathbb{R}^{6\times n}$ is the so-called *geometric Jacobian* of the manipulator. This last one can be seen as the composition of two sub-matrices, i.e.,

$$J(q) = \begin{bmatrix} J_p(q) \\ J_o(q) \end{bmatrix}, \tag{4.8}$$

where $J_p(q) \in \mathbb{R}^{3\times n}$ represents the contribution of the joints velocity to the linear velocity of the end-effector, while $J_o(q) \in \mathbb{R}^{3\times n}$ the contribution to the angular one.

It is clear that, to compute the differential kinematics, it is fundamental to provide an expression of the matrix $J(q)$. Noticing that each column of the Jacobian is associated with a joint, it is possible to compute each column of $J(q)$ independently. In particular, if the i-th joint is *prismatic*, one has that

$$J^{(i)}(q) = \begin{bmatrix} J_p^{(i)}(q) \\ J_o^{(i)}(q) \end{bmatrix} = \begin{bmatrix} z_{i-1} \\ 0_3 \end{bmatrix}, \tag{4.9}$$

while, if the i-th joint is a *revolute* joint, the associated column is given by

$$J^{(i)}(q) = \begin{bmatrix} J_p^{(i)}(q) \\ J_o^{(i)}(q) \end{bmatrix} = \begin{bmatrix} z_{i-1} \times (p_e^b - p_{i-1}) \\ z_{i-1} \end{bmatrix}. \tag{4.10}$$

The dependency on the joint variable vector q comes from the fact that z_{i-1}, p_{i-1}, and p_e are depends on the joint configuration. In particular, z_{i-1} is composed by the first three elements of the third column of the transformation matrix $T_{i-1}^b(q) = T_0^b A_i^0(q_i) \cdots A_{i-1}^{i-2}(q_{i-1})$, while p_{i-1} corresponds to the first three elements of the fourth column of the same matrix. As for p_e^b, it is the position of the end-effector frame with respect to the base frame and it is equal to the first three elements of the fourth column of $T_e^b(q)$.

In the case in which the modified DH convention is used, (4.9) and (4.10) are substituted, respectively, by the equations

$$J^{(i)}(q) = \begin{bmatrix} J_p^{(i)}(q) \\ J_o^{(i)}(q) \end{bmatrix} = \begin{bmatrix} z_i \\ 0_3 \end{bmatrix}$$

and

$$J^{(i)}(q) = \begin{bmatrix} J_p^{(i)}(q) \\ J_o^{(i)}(q) \end{bmatrix} = \begin{bmatrix} z_i \times (p_e^b - p_i) \\ z_i \end{bmatrix}.$$

Note that, in some cases, it is required to solve the so-called *inverse differential kinematics* problem, which finds the set of joints velocities $\dot{q}$, starting from a velocity of the end-effector v_e. Such a relation is given by

$$\dot{q} = J^{-1}(q)v_e. \tag{4.11}$$

Obviously, there are some cases in which it is not possible to compute the inverse of $J(q)$, e.g., $n \neq 6$, or the matrix $J(q)$ is singular. In such a condition, one could rely on sub-optimal approaches such as the *Moore-Penrose pseudoinverse* [83], defined as follows.

Definition 4.2 (Moore-Penrose pseudoinverse). *Let $A \in \mathbb{R}^{n\times m}$ be a matrix with real entries. Then, if A has full column rank, i.e., it has linearly independent columns, it has a left pseudoinverse computed as*

$$A^+ = (A^\top A)^{-1}A^\top$$

and it holds that $A^+A = I_m$. Otherwise, if A has full row rank, it has a right pseudoinverse given by

$$A^+ = A^\top(AA^\top)^{-1},$$

with condition $AA^+ = I_m$ being verified.

4.3.3 Inverse Kinematics

Computing the joint variables vector that corresponds to a certain pose of the end-effector is the objective of the so so-called *inverse kinematics* problem.

Let $T_e^\star$ be the transformation matrix that describes the pose of the end-effector for which is required to find the joint variable vector. Differently from forward kinematics, for open chain manipulators the computation of inverse kinematics may not be trivial, for three main reasons:

1. If the manipulator is redundant, i.e., $n > 6$, then it is possible that the pose $T_e^\star$ can be reached with multiple (potentially infinite) joint configurations.
2. In the case in which $T_e^\star$ does not belong to the operational space, the solution of the inverse kinematics problem does not exist.
3. In general, it is not guaranteed that a closed form solution is available.

For this reason, it is often convenient to treat the inverse kinematics problem as an optimization problem of the form

$$q^\star = \underset{q}{\operatorname{argmin}} \left\| T_e^b(q) - T_e^\star \right\|, \tag{4.12}$$

to which, depending on the requirements, are added constraints.

In the literature, several solutions to solve the the inverse kinematic problem have been proposed through the years (see, e.g., [83, 84, 85]).

4.4 Dynamic Modeling

A common way to formulate the dynamical model of a manipulator is the one which relies on Euler-Lagrange (EL) equations. One of the main advantages of EL formulation is that it provides a closed form of the dynamical model which explicitly contains the input, hence being suitable for control design. Other modeling techniques which do not have this advantage, but are widely used for implementation and simulation are the ones that rely on Newton-Euler formulation, which is a recursive numerical model, like [86] and [87].

Consider an open chain manipulator composed of rigid links, like the one described in the previous sections, and let $q \in \mathbb{R}^n$ be joint variables. These last ones, which describe the motion of the links of the manipulator, are also referred to as the *generalized coordinates*.

Another fundamental component in the EL formulation is the so-called *Lagrangian* of the system. In the case of a robotic manipulator with n DoF, the Lagrangian is a function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined as

$$\mathcal{L}(q, \dot{q}) = \mathcal{T}(q, \dot{q}) - \mathcal{U}(q), \tag{4.13}$$

where $\mathcal{T} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the total kinematic energy, while $\mathcal{U} : \mathbb{R}^n \to \mathbb{R}$ is the total potential energy.

The former is expressed as the quadratic form

$$\mathcal{T}(q, \dot{q}) = \frac{1}{2}\dot{q}^\top M(q)\dot{q}, \tag{4.14}$$

where $M : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ is the so-called *inertia matrix* of the manipulator, which is a Symmetric Positive Definite (SPD) matrix that collects the fixed masses and the pose-dependent rotational inertia terms required for the computation of the kinetic energy [24].

As for the latter, it is expressed as

$$\mathcal{U}(q) = -\sum_{i=1}^{n}(m_{l_i} g_0^\top p_{l_i} + m_{m_i} g_0^\top p_{m_i}), \tag{4.15}$$

where $g_0 \in \mathbb{R}^3$ is the gravity acceleration vector expressed in the base frame, m_{l_u} and m_{m_i} are the link and rotor masses, respectively, while $p_{l_i} \in \mathbb{R}^3$ and $p_{m_i} \in \mathbb{R}^3$

are the the positions of the link and rotor center of mass, respectively. Note that, these last two are dependent only on the generalized coordinate vector, making the potential energy dependent only on q.

The overall dynamical model of the robotic manipulator can be retrieved using the *Lagrange equations*, which relate the Lagrangian functions $\mathcal{L}$ and the vector of the generalized forces $\xi \in \mathbb{R}^n$ by applying *D'Alembert's principle*

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \xi_i, \tag{4.16}$$

for all $i \in \{1, 2, \dots, m\}$.

Consider the Lagrangian (4.13) with kinetic and potential energy (4.14) and (4.15). Then, substituting in (4.16), the robotic manipulator dynamics are given by

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F_v\dot{q} + F_s\text{sign}\,(\dot{q}) + g(q) = \tau, \tag{4.17}$$

where $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C(q, \dot{q})\dot{q} \in \mathbb{R}^n$ models the Coriolis and centrifugal forces, $F_v\dot{q} \in \mathbb{R}^n$ represents viscous friction, $F_s\text{sign}\,(\dot{q}) \in \mathbb{R}^n$ is the Coulomb friction, $g(q) \in \mathbb{R}^n$ is the vector of gravitational torques, while $\tau \in \mathbb{R}^n$ is the input torque given by the actuators. For the full computation of (4.17), the reader should refer to [24, Chapter 7].

For an open-chain manipulator, it is possible to bounds the terms of (4.17) as

$$k_0 < \|M(q)\| < k_1 + k_2\|q\| + k_3\|q\|^2 \tag{4.18a}$$

$$\|C(q, \dot{q})\| < (k_4 + k_5\|q\|)\|\dot{q}\| \tag{4.18b}$$

$$\|g(q)\| < k_6 + k_7\|q\|, \tag{4.18c}$$

with $k_0, k_1, \dots k_7 \in \mathbb{R}_{>0}$. In the case in which all the n joints are of a revolute type, it holds that $q_i \in [0, 2\pi)$. Hence, the bounds in (4.18) can be simplified as

$$k_0 < \|M(q)\| < k_1 \tag{4.19a}$$

$$\|C(q, \dot{q})\| < k_2\|\dot{q}\| \tag{4.19b}$$

$$\|g(q)\| < k_3, \tag{4.19c}$$

with $k_0, k_1, k_2, k_3 \in \mathbb{R}_{>0}$.

Part II

Deep Neural Network based Integral Sliding Mode Control Framework

Chapter 5

The DNN-ISM Framework

As detailed in Chapter 2, one of the main advantages of ISM control is that it rejects the matched perturbation acting on the system from the initial time instant thanks to an integral sliding variable. In particular, the sliding manifold is extended to the whole state space thanks to the introduction of the so-called transient function. However, the computation of such a component requires the complete knowledge of the system dynamics which, in many applications, may not be available.

The problem of designing a control law able to deal with uncertain dynamics has been addressed in different ways. In the literature, the two main frameworks that aim to solve the aforementioned problem are robust control [88, 1] and adaptive control [89, 48, 90]. Thanks to the universal approximation property of ANNs, in the last twenty years the literature has been enriched with methodologies that exploit them to compensate the lack of knowledge of plant dynamics (see, e.g., [91, 92] and the references therein).

In general, ANNs require a large amount of data to adjust their weights, i.e., their internal parameters. If this training phase is done offline, then when the trained networks are actually used in the control strategy they behave like static maps. Moreover, since ANNs are intrinsically affected by approximation errors, hence it is required to adopt robust control strategies. Moreover, as in general measurement are affected by noise, it often is difficult possible to have access to a large quantity of good-quality data, affecting the quality of the approximation.

To overcome such an issue, adaptive control principles may be exploited. Indeed, the training of the network weights can be performed via suitable adaptation laws, adjusting their value online while the plant is operating. Since such adaptation laws are obtained performing Lyapunov's analysis, it is possible to provide some theoretical guarantees on the controlled system. Example of controllers that employ

such a methodology can be found in [93, 94, 95, 96, 97].

This chapter presents the DNN-ISM control strategy for perturbed nonlinear system which, differently from the works present in the literature, is characterized by fully unknown dynamics. The proposed methodology exploits two DNNs with an arbitrary number of hidden layers to estimate the unknown drift dynamics and control effectiveness matrix, instrumental for the design of the ISM controller. The weights of the DNNs are adjusted online according to adaptation laws derived from stability analysis. The validity of the control scheme is assessed in simulation and experimentally on a real Franka Emika Panda robot, present at the University of Pavia.

5.1 Problem Formulation

Consider a control-affine system expressed in the canonical reduced form (see, Section 2.2.1) and affected by matched disturbance

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} f_1(x(t), t) \\ f_2(x(t), t) + \bar{B}(x(t), t)u(t) + h(x(t), t) \end{bmatrix}, \tag{5.1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$, with $\mathcal{X}$ being a compact set containing the origin, is the full state vector, $x_1 \in \mathcal{X}_1 \subset \mathbb{R}^{n-m}$ and $x_2 \in \mathcal{X}_2 \subset \mathbb{R}^m$ are its components, $u \in \mathbb{R}^m$ is the control vector, $f_1 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n-m}$ and $f_2 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ are the components of the drift dynamics, $\bar{B} : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{m \times m}$ is the control effectiveness matrix, while $h : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ is the matched perturbation vector.

The terms f_1, f_2, and $\bar{B}$ in (5.1) satisfy the following assumptions.

Assumption 5.1. *The terms f_1, f_2 are unknown functions of class $\mathrm{C}^0(\mathcal{X})$. Moreover, there exist some known constants $\bar{f}_1, \bar{f}_2 \in \mathbb{R}_{>0}$ such that, for all $x(t) \in \mathcal{X}$, it holds that*

$$\sup_{x(t) \in \mathcal{X}} \|f_1(x(t))\| \leq \bar{f}_1, \qquad \sup_{x(t) \in \mathcal{X}} \|f_2(x(t))\| \leq \bar{f}_2.$$

Assumption 5.2. *The term $\bar{B}$ is an unknown function of class $\mathrm{C}^0(\mathcal{X})$. Moreover, for all $x \in \mathcal{X}$ and $t \in \mathbb{R}_{\geq 0}$, the matrix $\bar{B}(x(t), t)$ is symmetric and positive definite, and there exist some known constants $\underline{\gamma}, \bar{\gamma} \in \mathbb{R}_{>0}$ so that*

$$\underline{\gamma} < \underline{\lambda}(\bar{B}(x(t), t)) \leq \bar{\lambda}(\bar{B}(x(t), t)) < \bar{\gamma},$$

with $\underline{\lambda}(\bar{B}(x(t), t))$ and $\bar{\lambda}(\bar{B}(x(t), t))$ being the smallest and largest eigenvalue of $\bar{B}(x(t), t)$, respectively.

As for the perturbation h, the following assumption holds.

Assumption 5.3. *There exists a known constant $\bar{h} \in \mathbb{R}_{>0}$ such that the matched perturbation is bounded in norm as*

$$h(x(t),t)) \in \mathcal{H}, \qquad \mathcal{H} \coloneqq \left\{ v(x(t),t) \in \mathbb{R}^m : \|v(x(t),t)\| \leq \bar{h} \right\},$$

for all $x \in \mathcal{X}$ and $t \in \mathbb{R}_{\geq 0}$.

Then, if one defines some desired trajectory for the system state

$$x^\star(t) = \begin{bmatrix} x_1^\star(t) \\ x_2^\star(t) \end{bmatrix} \in \mathcal{X}, \tag{5.2}$$

for all $t \in \mathbb{R}_{\geq 0}$, where $x_1^\star(t) \in \mathcal{X}_1$ and $x_2^\star(t) \in \mathcal{X}_2$ are functions of class C^1 with bounded derivatives $\dot{x}_1^\star(t) \in \mathbb{R}^{n-m}$ and $\dot{x}_2^\star(t) \in \mathbb{R}^m$, it is possible to express the dynamics of the tracking error $e(t) = x(t) - x^\star(t) \in \mathbb{R}^n$ as

$$\dot{e}(t) = \begin{bmatrix} \dot{x}_1(t) - \dot{x}_1^\star(t) \\ \dot{x}_2(t) - \dot{x}_2^\star(t) \end{bmatrix} = \begin{bmatrix} f_1(x(t),t) - \dot{x}_1^\star(t) \\ f_2(x(t),t) + \bar{B}(x(t),t)u(t) + h(x(t),t) - \dot{x}_2^\star(t) \end{bmatrix}, \tag{5.3}$$

characterized by initial condition $e(t_0) = x(t_0) - x^\star(t_0)$, with $t_0 \in \mathbb{R}_{\geq 0}$.

With the objective of stabilizing the error system in (5.3) around the origin while rejecting the matched disturbance from the initial time instant, one could design an ISM controller, detailed in Section 2.4 and recalled hereafter for convenience.

In particular, the integral sliding variable $\sigma : \mathcal{X} \to \mathbb{R}^m$ is defined as

$$\sigma(x(t)) = \sigma_0(x(t)) - z(x(t)), \tag{5.4}$$

where, thanks to the fact that system (5.3) is of reduced form, the conventional sliding variable $\sigma_0 : \mathcal{X} \to \mathbb{R}^m$ can be conveniently defined as

$$\sigma_0(x(t)) = C_1(x_1(t) - x_1^\star(t)) + C_2(x_2(t) - x_2^\star(t)), \tag{5.5}$$

with $C_1 \in \mathbb{R}^{m \times (n-m)}$ and $C_2 \in \mathbb{R}^{m \times m}$ being design matrices, with the latter satisfying the following assumption.

Assumption 5.4. *The design matrix $C_2 \in \mathbb{R}^{m \times m}$ is chosen so that $C_2\bar{B}(x(t),t)$ is symmetric and positive definite, for all $x \in \mathcal{X}$ and $t \geq t_0$.*

Note that, since $\bar{B}(x(t),t)$ is assumed to be symmetric and positive definite (see Assumption 5.2), it is sufficient to design, for example, $C_2 = kI_m$, with $k \in \mathbb{R}_{>0}$, for having Assumption 5.4 satisfied.

As for the transient variable $z : \mathcal{X} \to \mathbb{R}^m$ it is defined as

$$z(x(t)) = \sigma_0(x(t_0)) + \int_{t_0}^{t} \Big\{ C_1 \Big[f_1(x(\tau), \tau) - \dot{x}_1^\star(\tau) \Big] + \\ + C_2 \Big[f_2(x(\tau), \tau) + \bar{B}(x(\tau), \tau) u_n(\tau) - \dot{x}_2^\star(\tau) \Big] \Big\} d\tau, \tag{5.6}$$

implying that

$$\dot{z}(x(t)) = C_1 \Big[f_1(x(t), t) - \dot{x}_1^\star(t) \Big] + C_2 \Big[f_2(x(t), t) + \bar{B}(x(t), t) u_n(t) - \dot{x}_2^\star(t) \Big], \tag{5.7}$$

with initial condition $z(x(t_0)) = \sigma_0(x(t_0))$. The term $u_n(t) \in \mathbb{R}^m$, as detailed in Section 2.4, is the nominal control law that dictates the dynamics of the system while in sliding mode. The complete control law is defined as

$$u(t) = u_n(t) + u_r(t), \tag{5.8}$$

where $u_r(t) \in \mathbb{R}^m$ is the discontinuous component, defined according to the unit vector approach

$$u_r(t) = -\rho \frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \tag{5.9}$$

with $\rho \in \mathbb{R}_{>0}$ being the discontinuous control gain. If this last one is chosen according to Theorem 2.3, then a sliding mode on the manifold $\sigma(x(t)) = 0_m$ is enforced for $t \geq t_0$. Then, if $u_n(t)$ is properly designed, the tracking error $e(t)$ is steered to zero.

However, as specified in Assumptions 5.1 and 5.2, the dynamics of the system is fully unknown, implying that it is not possible to compute the transient variable in (5.6), and, as a consequence, enforce a sliding mode condition.

5.2 Approximating the Dynamics using DNNs

As detailed in Section 3.1.2, DNNs can be employed as powerful function approximators. In this case, it is possible to approximate the dynamics of the system (5.1) using two ideal DNNs $\Phi : \mathcal{X} \to \mathbb{R}^n$ and $\Psi : \mathcal{X} \to \mathbb{R}^{m^2}$ as

$$f(x(t), t) = \begin{bmatrix} f_1(x(t), t) \\ f_2(x(t), t) \end{bmatrix} = \Phi(x(t)) + \varepsilon_\Phi(x(t)), \tag{5.10a}$$

$$\bar{B}(x(t), t) = \mathrm{vec}^{-1}\left(\Psi(x(t))\right) + \varepsilon_\Psi(x(t)), \tag{5.10b}$$

where $\varepsilon_\Phi : \mathcal{X} \to \mathbb{R}^n$ and $\varepsilon_\Psi : \mathcal{X} \to \mathbb{R}^{m \times m}$ are the approximation errors. Note that, differently from what done in Chapter 3, in this case ε_Ψ is defined as a matrix for convenience. The DNN Φ is characterized by $k_\Phi \in \mathbb{N}_{\geq 2}$ hidden layers and each layer

j, with $j \in \{0, 1, \dots, k_\Phi\}$, contains L_{Φ_j} neurons and activated using a nonlinear function $\phi_j : \mathbb{R}^{L_{\Phi_j}} \to \mathbb{R}^{L_{\Phi_j}}$. Similarly, Ψ is characterized by $k_\Psi \in \mathbb{N}_{\geq 2}$ hidden layers and each layer j, with $j \in \{0, 1, \dots, k_\Psi\}$, contains L_{Ψ_j} neurons and activated using a nonlinear function $\psi_j : \mathbb{R}^{L_{\Psi_j}} \to \mathbb{R}^{L_{\Psi_j}}$. The following assumption about the activation functions hold.

Assumption 5.5. *The activation functions $\phi_j : \mathbb{R}^{L_{\Phi_j}} \to \mathbb{R}^{L_{\Phi_j}}$ and $\psi_p : \mathbb{R}^{L_{\Psi_p}} \to \mathbb{R}^{L_{\Psi_p}}$, with $j \in \{0, 1, \dots, k_\Phi\}$ and $p \in \{0, 1, \dots, k_\Psi\}$, are of class C^1 and Lipschitz continuous.*

Example of functions satisfying such an assumption are the sigmoid and the hyperbolic tangent.

It is possible to express the output of each layer of Φ and Ψ in the form proposed in (3.6). In particular, for the former DNN, one has

$$\Phi_j = \begin{cases} V_j^\top \phi_j(\Phi_{j-1}) & \text{for } j \in \{1, 2, \dots, k_\Phi\} \\ V_0^\top x_h & \text{for } j = 0, \end{cases} \tag{5.11}$$

where $V_j \in \mathbb{R}^{L_{\Phi_j} \times L_{\Phi_{j+1}}}$ is the matrix containing ideal weights and biases associated with the j-th layer and $x_h = \begin{bmatrix} x^\top & 1 \end{bmatrix}^\top \in \mathbb{R}^{n+1}$. As for the DNN Ψ, a similar expression can be derived, i.e.,

$$\Psi_j = \begin{cases} U_j^\top \psi_j(\Psi_{j-1}) & \text{for } j \in \{1, 2, \dots, k_\Psi\} \\ U_0^\top x_h & \text{for } j = 0, \end{cases} \tag{5.12}$$

where, similarly to the previous case, $U_j \in \mathbb{R}^{L_{\Psi_j} \times L_{\Psi_{j+1}}}$ is the matrix containing ideal weights and bias associated with the j-th layer of the DNN. Moreover, since the overall output of a DNN coincides with the output of the last layer, it holds that $\Phi(x(t)) \equiv \Phi_{k_\Phi}$ and $\Psi(x(t)) \equiv \Psi_{k_\Psi}$, and, as a consequence, the expression (5.10) can be rewritten as

$$f(x(t), t) = \begin{bmatrix} f_1(x(t), t) \\ f_2(x(t), t) \end{bmatrix} = \Phi_{k_\Phi} + \varepsilon_\Phi(x(t)), \tag{5.13a}$$

$$\bar{B}(x(t), t) = \mathrm{vec}^{-1}\left(\Psi_{k_\Psi}\right) + \varepsilon_\Psi(x(t)). \tag{5.13b}$$

Then, it is convenient to provide an expression for f_1, f_2 and of each i-th column of $\bar{B}$, with $i \in \{1, 2, \dots, m\}$. In particular, following the reasoning in Section 3.1.2, the matrices $V_{k_\Phi} \in \mathbb{R}^{L_{k_\Phi} \times n}$ and $U_{k_\Psi} \in \mathbb{R}^{L_{k_\Psi} \times m^2}$, with $L_{k_\Phi} = L_{\Phi_{k_\Phi}}$ and $L_{k_\Psi} = L_{\Psi_{k_\Psi}}$ for sake of readability, can be seen as the composition of different sub-matrices, i.e.,

$$V_{k_\Phi} = \begin{bmatrix} V_{k_\Phi}^{[1]} & V_{k_\Phi}^{[2]} \end{bmatrix},$$

$$U_{k_\Psi} = \begin{bmatrix} U_{k_\Psi}^{[1]} & U_{k_\Psi}^{[2]} & \cdots & U_{k_\Psi}^{[m]} \end{bmatrix},$$

where $V_{k_\Phi}^{[1]} \in \mathbb{R}^{L_{k_\Phi} \times (n-m)}$, $V_{k_\Phi}^{[2]} \in \mathbb{R}^{L_{k_\Phi} \times m}$, and $U_{k_\Psi}^{[i]} \in \mathbb{R}^{L_{k_\Psi} \times m}$. Then, one has that

$$f_1(x(t),t) = (V_{k_\Phi}^{[1]})^\top \phi_{k_\Phi}(\Phi_{k_\Phi - 1}) + \varepsilon_\Phi^{[1]}(x(t)) = \Phi_{k_\Phi}^{[1]} + \varepsilon_\Phi^{[1]}(x(t)), \tag{5.14a}$$

$$f_2(x(t),t) = (V_{k_\Phi}^{[2]})^\top \phi_{k_\Phi}(\Phi_{k_\Phi - 1}) + \varepsilon_\Phi^{[2]}(x(t)) = \Phi_{k_\Phi}^{[2]} + \varepsilon_\Phi^{[2]}(x(t)), \tag{5.14b}$$

$$\bar{B}^{(i)}(x(t),t) = (U_{k_\Psi}^{[i]})^\top \psi_{k_\Psi}(\Psi_{k_\Psi - 1}) + \varepsilon_\Psi^{(i)}(x(t)) = \Psi_{k_\Psi}^{[i]} + \varepsilon_\Psi^{(i)}(x(t)). \tag{5.14c}$$

The terms $\varepsilon_\Phi^{[1]} : \mathcal{X} \to \mathbb{R}^{n-m}$ and $\varepsilon_\Phi^{[2]} : \mathcal{X} \to \mathbb{R}^m$ are the components of ε_Φ, while $\varepsilon_\Psi^{(i)} : \mathcal{X} \to \mathbb{R}^m$ is the i-th column of ε_Ψ.

By virtue of the universal approximation property of Φ and Ψ and the boundedness of f_1, f_2, and $\bar{B}$ (see Assumptions 5.1 and 5.2), the following assumption about the ideal DNNs can be introduced.

Assumption 5.6. *There exist some known constants $\bar{V}, \bar{U}, \bar{\varepsilon}_{\Phi_1}, \bar{\varepsilon}_{\Phi_2}, \bar{\varepsilon}_\Psi \in \mathbb{R}_{>0}$ such that the ideal weights and the approximation errors are bounded as*

$$\sup_{j \in \{0,1,\dots,k_\Phi\}} \|V_j\| \leq \bar{V}, \qquad \sup_{j \in \{0,1,\dots,k_\Psi\}} \|U_j\| \leq \bar{U},$$

$$\sup_{x(t) \in \mathcal{X}} \left\| \varepsilon_\Phi^{[1]}(x(t)) \right\| \leq \bar{\varepsilon}_{\Phi_1}, \quad \sup_{x(t) \in \mathcal{X}} \left\| \varepsilon_\Phi^{[2]}(x(t)) \right\| \leq \bar{\varepsilon}_{\Phi_2}, \quad \sup_{x(t) \in \mathcal{X}} \|\varepsilon_\Psi(x(t))\| \leq \bar{\varepsilon}_\Psi.$$

Note that the knowledge of the matrices V_j and U_j, is not available. Hence, the ideal estimates in (5.10) cannot be computed. For this reason, it is possible to define another couple of DNNs, namely $\hat{\Phi} : \mathcal{X} \to \mathbb{R}^n$ and $\hat{\Psi} : \mathcal{X} \to \mathbb{R}^{m^2}$, that have the same structure and activation functions of the ideal ones, but they are characterized by an approximation of the ideal weights and biases.

Similarly to the ideal case, an expression the output of each layer of the DNNs can be provided. In particular, for $\hat{\Phi}$ on has

$$\hat{\Phi}_j = \begin{cases} \hat{V}_j^\top \phi_j(\hat{\Phi}_{j-1}) & \text{for } j \in \{1, 2, \dots, k_\Phi\} \\ \hat{V}_0^\top x_h & \text{for } j = 0, \end{cases} \tag{5.15}$$

where $\hat{V}_j \in \mathbb{R}^{L_{\Phi_j} \times L_{\Phi_{j+1}}}$ is the estimate of V_j. As for $\hat{\Psi}$, one has that

$$\hat{\Psi}_j = \begin{cases} \hat{U}_j^\top \psi_j(\hat{\Psi}_{j-1}) & \text{for } j \in \{1, 2, \dots, k_\Psi\} \\ \hat{U}_0^\top x_h & \text{for } j = 0, \end{cases} \tag{5.16}$$

with $\hat{U}_j \in \mathbb{R}^{L_{\Psi_j} \times L_{\Psi_{j+1}}}$ being the estimate of U_j. Then, similarly for what done for the ideal case, the matrices associated with the last layers can be written as the composition of sub-matrices, i.e.

$$\hat{V}_{k_\Phi} = \begin{bmatrix} \hat{V}_{k_\Phi}^{[1]} & \hat{V}_{k_\Phi}^{[2]} \end{bmatrix},$$
$$\hat{U}_{k_\Psi} = \begin{bmatrix} \hat{U}_{k_\Psi}^{[1]} & \hat{U}_{k_\Psi}^{[2]} & \cdots & \hat{U}_{k_\Psi}^{[m]} \end{bmatrix},$$

where $\hat{V}_{k_\Phi}^{[1]} \in \mathbb{R}^{L_{k_\Phi} \times (n-m)}$, $\hat{V}_{k_\Phi}^{[2]} \in \mathbb{R}^{L_{k_\Phi} \times m}$, and $\hat{U}_{k_\Psi}^{[i]} \in \mathbb{R}^{L_{k_\Psi} \times m}$. Hence, it is possible to compute $\hat{f}_1$, $\hat{f}_2$, and the columns of $\hat{\bar{B}}$ as

$$\hat{f}_1(x(t), t) = (\hat{V}_{k_\Phi}^{[1]})^\top \phi_{k_\Phi}(\hat{\Phi}_{k_\Phi - 1}) = \hat{\Phi}_{k_\Phi}^{[1]}, \tag{5.17a}$$
$$\hat{f}_2(x(t), t) = (\hat{V}_{k_\Phi}^{[2]})^\top \phi_{k_\Phi}(\hat{\Phi}_{k_\Phi - 1}) = \hat{\Phi}_{k_\Phi}^{[2]}, \tag{5.17b}$$
$$\hat{\bar{B}}^{(i)}(x(t), t) = (\hat{U}_{k_\Psi}^{[i]})^\top \psi_{k_\Psi}(\hat{\Psi}_{k_\Psi - 1}) = \hat{\Psi}_{k_\Psi}^{[i]}, \tag{5.17c}$$

with $i \in \{1, 2, \ldots, m\}$. Before introducing the DNN-ISM control strategy, it is required to introduce the expression of the approximation error of the DNNs, which is instrumental for the design of the adaptation laws and the stability analysis. In the following, the formulation of the error between the optimal DNNs and the estimated ones will be provided, first considering Φ, and then Ψ.

5.2.1 Approximation error of the Drift Dynamics DNN

Consider the error of $\hat{\Phi}$ with respect to Φ, given by

$$\tilde{\Phi}(x(t)) = \Phi(x(t)) - \hat{\Phi}(x(t)). \tag{5.18}$$

Such an error can be expressed for each layer $j \in \{1, 2, \ldots, k_\Phi\}$ as

$$\begin{aligned} \tilde{\Phi}_j &= \Phi_j - \hat{\Phi}_j \in \mathbb{R}^{L_{\Phi_{j+1}}} \\ &= V_j^\top \phi_j(\Phi_{j-1}) - \hat{V}_j^\top \phi_j(\hat{\Phi}_{j-1}) \\ &= V_j^\top \phi_j(\Phi_{j-1}) - \hat{V}_j^\top \phi_j(\hat{\Phi}_{j-1}) + V_j^\top \phi_j(\hat{\Phi}_{j-1}) - V_j^\top \phi_j(\hat{\Phi}_{j-1}) \\ &= \tilde{V}_j^\top \phi_j(\hat{\Phi}_{j-1}) + V_j^\top \left(\phi_j(\Phi_{j-1}) - \phi_j(\hat{\Phi}_{j-1}) \right), \end{aligned} \tag{5.19}$$

where $\tilde{V}_j = V_j - \hat{V}_j$. For $j = 0$, one has that

$$\begin{aligned} \tilde{\Phi}_0 &= \Phi_0 - \hat{\Phi}_0 \\ &= V_0^\top x_h - \hat{V}_0^\top x_h \\ &= \tilde{V}_0^\top x_h. \end{aligned} \tag{5.20}$$

Expression (5.19) cannot be computed directly since the ideal activations $\phi_j(\Phi_{j-1})$ are unknown. However, these last ones can be approximated using Taylor expansion centered around the estimated vector $\hat{\Phi}_{j-1}$, obtaining

$$\phi_j(\Phi_{j-1}) = \phi_j(\hat{\Phi}_{j-1}) + \left.\frac{\partial \phi_j}{\partial \Phi}\right|_{\Phi=\hat{\Phi}_{j-1}} (\Phi_{j-1} - \hat{\Phi}_{j-1}) + \mathrm{O}^2(\tilde{\Phi}_{j-1}), \tag{5.21}$$

where $\left.\frac{\partial \phi_j}{\partial \Phi}\right|_{\Phi=\hat{\Phi}_{j-1}} \in \mathbb{R}^{L_{\Phi_j} \times L_{\Phi_j}}$ is the Jacobian matrix of the activation function vector $\phi_j(\cdot)$ with respect to its argument, computed in the estimated output of the previous layer $\hat{\Phi}_{j-1}$. As for $\mathrm{O}^2(\tilde{\Phi}_{j-1}) \in \mathbb{R}^{L_{\Phi_j}}$, it denotes the lumped terms of order higher than one.

Substituting (5.21) in (5.19), and having $\phi_j = \phi_j(\Phi_{j-1})$, $\hat{\phi}_j = \phi_j(\hat{\Phi}_{j-1})$, and $\hat{\phi}'_j = \left.\frac{\partial \phi_j}{\partial \Phi}\right|_{\Phi=\hat{\Phi}_{j-1}}$ for sake of readability, one has

$$\begin{aligned}
\tilde{\Phi}_j &= \tilde{V}_j^\top \hat{\phi}_j + V_j^\top (\phi_j - \hat{\phi}_j) \\
&= \tilde{V}_j^\top \hat{\phi}_j + V_j^\top \left(\hat{\phi}_j + \hat{\phi}'_j \tilde{\Phi}_{j-1} + \mathrm{O}^2(\tilde{\Phi}_{j-1}) - \hat{\phi}_j \right) \\
&= \tilde{V}_j^\top \hat{\phi}_j + V_j^\top \left(\hat{\phi}'_j \tilde{\Phi}_{j-1} + \mathrm{O}^2(\tilde{\Phi}_{j-1}) \right) \\
&= \tilde{V}_j^\top \hat{\phi}_j + V_j^\top \hat{\phi}'_j \tilde{\Phi}_{j-1} + V_j^\top \mathrm{O}^2(\tilde{\Phi}_{j-1}).
\end{aligned} \tag{5.22}$$

Since, $V_j = \tilde{V}_j + \hat{V}_j$, it is possible to write

$$\begin{aligned}
\tilde{\Phi}_j &= \tilde{V}_j^\top \hat{\phi}_j + V_j^\top \hat{\phi}'_j \tilde{\Phi}_{j-1} + V_j^\top \mathrm{O}^2(\tilde{\Phi}_{j-1}) \\
&= \tilde{V}_j^\top \hat{\phi}_j + (\tilde{V}_j^\top + \hat{V}_j^\top) \hat{\phi}'_j \tilde{\Phi}_{j-1} + V_j^\top \mathrm{O}^2(\tilde{\Phi}_{j-1}) \\
&= \tilde{V}_j^\top \hat{\phi}_j + \tilde{V}_j^\top \hat{\phi}'_j \tilde{\Phi}_{j-1} + \hat{V}_j^\top \hat{\phi}'_j \tilde{\Phi}_{j-1} + V_j^\top \mathrm{O}^2(\tilde{\Phi}_{j-1}) \\
&= \tilde{V}_j^\top \hat{\phi}_j + \hat{V}_j^\top \hat{\phi}'_j \tilde{\Phi}_{j-1} + \Delta_{\Phi_j},
\end{aligned} \tag{5.23}$$

where $\Delta_{\Phi_j} \coloneqq \tilde{V}_j^\top \hat{\phi}'_j \tilde{\Phi}_{j-1} + V_j^\top \mathrm{O}^2(\tilde{\Phi}_{j-1}) \in \mathbb{R}^{L_{\Phi_{j+1}}}$.

Before going further with the analysis, the following property of the Kronecker product, defined as in Definition 3.2, is introduced.

Lemma 5.1 (Kronecker product and vectors [98, Proposition 7.1.9]). *Given three matrices $A \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^{m \times q}$, and $C \in \mathbb{R}^{q \times p}$, and letting* $\mathrm{vec}\,(\cdot)$ *be the vectorization operation in Definition 3.1, it holds that*

$$\mathrm{vec}\,(ABC) = \left(C^\top \otimes A\right) \mathrm{vec}\,(B) \in \mathbb{R}^{np}$$

Since $\tilde{V}_j^\top \hat{\phi}_j \in \mathbb{R}^{L_{\Phi_{j+1}}}$, it holds that

$$\tilde{V}_j^\top \hat{\phi}_j = \mathrm{vec}\left(\tilde{V}^\top \hat{\phi}_j\right) = \mathrm{vec}\left(\hat{\phi}_j^\top \tilde{V}_j\right) = \mathrm{vec}\left(\hat{\phi}_j^\top \tilde{V}_j I_{L_{\Phi_{j+1}}}\right),$$

and, applying Lemma 5.1 with $A \equiv \hat{\phi}_j^\top$, $B \equiv \tilde{V}_j$, and $C \equiv I_{L_{\Phi_{j+1}}}$ one can write

$$\tilde{V}_j^\top \hat{\phi}_j = \left(I_{L_{\Phi_{j+1}}}^\top \otimes \hat{\phi}_j^\top\right) \mathrm{vec}\left(\tilde{V}_j\right) = \left(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top\right) \mathrm{vec}\left(\tilde{V}_j\right).$$

Then, substituting $\tilde{V}_j^\top \hat{\phi}_j$ one has

$$\begin{aligned}\tilde{\Phi}_j &= \tilde{V}_j^\top \hat{\phi}_j + \hat{V}_j^\top \hat{\phi}_j' \tilde{\Phi}_{j-1} + \Delta_{\Phi_j} \\ &= \left(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top\right) \mathrm{vec}\left(\tilde{V}_j\right) + \hat{V}_j^\top \hat{\phi}_j' \tilde{\Phi}_{j-1} + \Delta_{\Phi_j}, \end{aligned} \tag{5.24}$$

for $j \in \{1, 2, \ldots, k_\Phi\}$. As for the input layer $j = 0$, one has $\phi_0 = \hat{\phi}_0 = x_h$, meaning that the Taylor expansion is not needed and hence

$$\begin{aligned}\tilde{\Phi}_0 &= \tilde{V}_0^\top x_h \\ &= \mathrm{vec}\left(\tilde{V}_0^\top x_h\right) \\ &= \left(I_{L_{\Phi_1}} \otimes x_h^\top\right) \mathrm{vec}\left(\tilde{V}_0\right), \end{aligned} \tag{5.25}$$

Before going any further, it is convenient to introduce the oriented product operator, instrumental for the following analysis.

Definition 5.1. *Given some matrices A_i, with $i \in \{1, 2, \ldots, N\}$, characterized by compatible dimensions, then*

$$\prod_{1=1}^{\overset{\curvearrowleft}{N}} A_i = A_N A_{N-1} \cdots A_1,$$

represents the oriented product. Moreover, it holds that

$$\prod_{1=p}^{\overset{\curvearrowleft}{p-1}} A_i \coloneqq 1.$$

The overall estimation error of the DNN coincides with (5.24) with $j = k_\Phi$, having

$$\tilde{\Phi}_{k_\Phi} = \left(I_n \otimes \hat{\phi}_{k_\Phi}^\top\right) \mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \hat{V}_{k_\Phi}^\top \hat{\phi}_{k_\Phi}' \tilde{\Phi}_{k_\Phi - 1} + \Delta_{\Phi_{k_\Phi}}.$$

In general, the expression of $\tilde{\Phi}_j$ depends on $\tilde{\Phi}_{j-1}$. Hence, also the above equation has a recursive nature. In fact, substituting $\tilde{\Phi}_{k_\Phi - 1}$, one has

$$\begin{aligned}\tilde{\Phi}_{k_\Phi} &= \left(I_n \otimes \hat{\phi}_{k_\Phi}^\top\right) \mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \hat{V}_{k_\Phi}^\top \hat{\phi}_{k_\Phi}' \tilde{\Phi}_{k_\Phi - 1} + \Delta_{\Phi_{k_\Phi}} \\ &= \left(I_n \otimes \hat{\phi}_{k_\Phi}^\top\right) \mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \hat{V}_{k_\Phi}^\top \hat{\phi}_{k_\Phi}' \left(I_{L_{k_\Phi}} \otimes \hat{\phi}_{k_\Phi - 1}^\top\right) \mathrm{vec}\left(\tilde{V}_{k_\Phi - 1}\right) + \\ &\quad + \Delta_{\Phi_{k_\Phi}} + \hat{V}_{k_\Phi}^\top \hat{\phi}_{k_\Phi}' \Delta_{\Phi_{k_\Phi - 1}} + \hat{V}_{k_\Phi}^\top \hat{\phi}_{k_\Phi}' \hat{V}_{k_\Phi - 1}^\top \hat{\phi}_{k_\Phi - 1}' \tilde{\Phi}_{k_\Phi - 2}\end{aligned}$$

Then, substituting $\tilde{\Phi}_{k_\Phi-2}$ leads to

$$\begin{aligned}\tilde{\Phi}_{k_\Phi} = &\left(I_n \otimes \hat{\phi}_{k_\Phi}^\top\right) \mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \hat{V}_{k_\Phi}^\top \hat{\phi}'_{k_\Phi} \left(I_{L_{k_\Phi}} \otimes \hat{\phi}_{k_\Phi-1}^\top\right) \mathrm{vec}\left(\tilde{V}_{k_\Phi-1}\right) + \\ &+ \Delta_{\Phi_{k_\Phi}} + \hat{V}_{k_\Phi}^\top \hat{\phi}'_{k_\Phi} \Delta_{\Phi_{k_\Phi-1}} + \hat{V}_{k_\Phi}^\top \hat{\phi}'_{k_\Phi} \hat{V}_{k_\Phi-1}^\top \hat{\phi}'_{k_\Phi-1} \Delta_{\Phi_{k_\Phi-2}} + \\ &+ \hat{V}_{k_\Phi}^\top \hat{\phi}'_{k_\Phi} \hat{V}_{k_\Phi-1}^\top \hat{\phi}'_{k_\Phi-1} \left(I_{L_{k_\Phi-1}} \otimes \hat{\phi}_{k_\Phi-2}^\top\right) \mathrm{vec}\left(\tilde{V}_{k_\Phi-2}\right) + \\ &+ \hat{V}_{k_\Phi}^\top \hat{\phi}'_{k_\Phi} \hat{V}_{k_\Phi-1}^\top \hat{\phi}'_{k_\Phi-1} \hat{V}_{k_\Phi-2}^\top \hat{\phi}'_{k_\Phi-2} \tilde{\Phi}_{k_\Phi-3}\end{aligned}$$

If one keeps substituting until $\tilde{\Phi}_0$ grouping the similar terms, the overall approximation error can be conveniently expressed as

$$\tilde{\Phi}_{k_\Phi} = \sum_{j=0}^{k_\Phi} \left(\prod_{l=j+1}^{\overset{\curvearrowleft}{k_\Phi}} \hat{V}_l^\top \hat{\phi}'_l \right) \left(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top\right) \mathrm{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi} \left(\prod_{l=j+1}^{\overset{\curvearrowleft}{k_\Phi}} \hat{V}_l^\top \hat{\phi}'_l \right) \Delta_{\Phi_j},$$

recalling that, applying Definition 5.1,

$$\prod_{l=j+1}^{\overset{\curvearrowleft}{k_\Phi}} \hat{V}_l^\top \hat{\phi}'_l \Bigg|_{j=k_\Phi} = \prod_{l=k_\Phi+1}^{\overset{\curvearrowleft}{k_\Phi}} \hat{V}_l^\top \hat{\phi}'_l \coloneqq 1.$$

Note that, since the Taylor's expansion is not required for $j = 0$, it holds that $\Delta_{\Phi_0} = 0_{L_{\Phi_1}}$ and, as a consequence, the second summation starts at $j = 1$.

The above expression of $\tilde{\Phi}_{k_\Phi}$ can be simplified introducing the matrices $\Xi_{\Phi_j} \in \mathbb{R}^{n \times L_{\Phi_{j+1}}}$ and $\Lambda_{\Phi_j} \in \mathbb{R}^{n \times L_{\Phi_j} L_{\Phi_{j+1}}}$, defined as

$$\Xi_{\Phi_j} \coloneqq \prod_{l=j+1}^{\overset{\curvearrowleft}{k_\Phi}} \hat{V}_l^\top \hat{\phi}'_l, \tag{5.26a}$$

$$\Lambda_{\Phi_j} \coloneqq \Xi_{\Phi_j} \left(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top\right), \tag{5.26b}$$

with $\Lambda_{\Phi_0} = \Xi_{\Phi_0}(I_{L_{\Phi_1}} \otimes x_h)$ and $\Xi_{\Phi_{k_\Phi}} \coloneqq 1$. In particular

$$\tilde{\Phi}_{k_\Phi} = \sum_{j=0}^{k_\Phi} \Lambda_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi} \Xi_{\Phi_j} \Delta_{\Phi_j}. \tag{5.27}$$

Since Φ is the DNN that approximates the drift term components $f_1 \in \mathbb{R}^{n-m}$ and $f_2 \in \mathbb{R}^m$, one can conveniently define the approximation error of the complete network associated with specific outputs, i.e.

$$\begin{aligned}\tilde{\Phi}_{k_\Phi}^{[p]} &= \Phi_{k_\Phi}^{[p]} - \hat{\Phi}_{k_\Phi}^{[p]} \\ &= (V_{k_\Phi}^{[p]})^\top \phi_{k_\Phi} - (\hat{V}_{k_\Phi}^{[p]})^\top \hat{\phi}_{k_\Phi} \\ &= (V_{k_\Phi}^{[p]})^\top \phi_{k_\Phi} - (\hat{V}_{k_\Phi}^{[p]})^\top \hat{\phi}_{k_\Phi} + (V_{k_\Phi}^{[p]})^\top \hat{\phi}_{k_\Phi} - (V_{k_\Phi}^{[p]})^\top \hat{\phi}_j\end{aligned}$$

$$
\begin{aligned}
&= \left((V_{k_\Phi}^{[p]})^\top - (\hat{V}_{k_\Phi}^{[p]})^\top\right)\hat{\phi}_{k_\Phi} + (V_{k_\Phi}^{[p]})^\top\left(\phi_{k_\Phi} - \hat{\phi}_{k_\Phi}\right)\\
&= (\tilde{V}_{k_\Phi}^{[p]})^\top\hat{\phi}_{k_\Phi} + (V_{k_\Phi}^{[p]})^\top\left(\phi_{k_\Phi} - \hat{\phi}_{k_\Phi}\right)\\
&= (\tilde{V}_{k_\Phi}^{[p]})^\top\hat{\phi}_{k_\Phi} + (\hat{V}_{k_\Phi}^{[p]})^\top\hat{\phi}'_{k_\Phi}\tilde{\Phi}_{k_\Phi-1} + \Delta_{\Phi_{k_\Phi}}^{[p]},
\end{aligned}
$$

with $p \in \{1, 2\}$. Then, using the properties of the Kronecker product, one has

$$
\begin{aligned}
\tilde{\Phi}_{k_\Phi}^{[1]} &= \left(I_{(n-m)} \otimes \hat{\phi}_{k_\Phi}{}^\top\right)\mathrm{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right) + (\hat{V}_{k_\Phi}^{[1]})^\top\hat{\phi}'_{k_\Phi}\tilde{\Phi}_{k_\Phi-1} + \Delta_{\Phi_{k_\Phi}}^{[1]} \in \mathbb{R}^{n-m},\\
\tilde{\Phi}_{k_\Phi}^{[2]} &= \left(I_{m} \otimes \hat{\phi}_{k_\Phi}{}^\top\right)\mathrm{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right) + (\hat{V}_{k_\Phi}^{[2]})^\top\hat{\phi}'_{k_\Phi}\tilde{\Phi}_{k_\Phi-1} + \Delta_{\Phi_{k_\Phi}}^{[2]} \in \mathbb{R}^{m},
\end{aligned}
$$

where the last elements of the above equations are $\Delta_{\Phi_{k_\Phi}}^{[1]} \coloneqq (\tilde{V}_{k_\Phi}^{[1]})^\top\hat{\phi}'_{k_\Phi}\tilde{\Phi}_{k_\Phi-1} + (V_{k_\Phi}^{[1]})^\top\mathrm{O}^2(\tilde{\Phi}_{k_\Phi-1}) \in \mathbb{R}^{n-m}$ and $\Delta_{\Phi_{k_\Phi}}^{[2]} \coloneqq (\tilde{V}_{k_\Phi}^{[2]})^\top\hat{\phi}'_{k_\Phi}\tilde{\Phi}_{k_\Phi-1} + (V_{k_\Phi}^{[2]})^\top\mathrm{O}^2(\tilde{\Phi}_{k_\Phi-1}) \in \mathbb{R}^{m}$.

Recursively substituting $\tilde{\Phi}_{j-1}$ as done previously, the explicit expression of $\tilde{\Phi}_{k_\Phi}^{[1]}$ and $\tilde{\Phi}_{k_\Phi}^{[2]}$ is given by

$$
\begin{aligned}
\tilde{\Phi}^{[1]}(x) = \sum_{j=0}^{k_\Phi-1}\left((\hat{V}_{k_\Phi}^{[1]})^\top\hat{\phi}'_{k_\Phi}\right)\left(\overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Phi-1}}\hat{V}_l^\top\hat{\phi}'_l\right)\left(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top\right)\mathrm{vec}\left(\tilde{V}_j\right) + \Delta_{\Phi_{k_\Phi}}^{[1]} +\\
+ \left(I_{(n-m)} \otimes \hat{\phi}_{k_\Phi}^\top\right)\mathrm{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right) + \sum_{j=1}^{k_\Phi-1}\left((\hat{V}_{k_\Phi}^{[1]})^\top\hat{\phi}'_{k_\Phi}\right)\left(\overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Phi-1}}\hat{V}_l^\top\hat{\phi}'_l\right)\Delta_{\Phi_j},
\end{aligned}
$$

and

$$
\begin{aligned}
\tilde{\Phi}^{[2]}(x) = \sum_{j=0}^{k_\Phi-1}\left((\hat{V}_{k_\Phi}^{[2]})^\top\hat{\phi}'_{k_\Phi}\right)\left(\overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Phi-1}}\hat{V}_l^\top\hat{\phi}'_l\right)\left(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top\right)\mathrm{vec}\left(\tilde{V}_j\right) + \Delta_{\Phi_{k_\Phi}}^{[2]} +\\
+ \left(I_{m} \otimes \hat{\phi}_{k_\Phi}^\top\right)\mathrm{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right) + \sum_{j=1}^{k_\Phi-1}\left((\hat{V}_{k_\Phi}^{[2]})^\top\hat{\phi}'_{k_\Phi}\right)\left(\overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Phi-1}}\hat{V}_l{}^\top\hat{\phi}'_l\right)\Delta_{\Phi_j}.
\end{aligned}
$$

Then, if one defines the matrices $\Xi_{\Phi_j}^{[1]} \in \mathbb{R}^{(n-m)\times L_{\Phi_{j+1}}}$, $\Xi_{\Phi_j}^{[2]} \in \mathbb{R}^{m\times L_{\Phi_{j+1}}}$, $\Lambda_{\Phi_j}^{[1]} \in \mathbb{R}^{(n-m)\times L_{\Phi_j}L_{\Phi_{j+1}}}$, and $\Lambda_{\Phi_j}^{[2]} \in \mathbb{R}^{m\times L_{\Phi_j}L_{\Phi_{j+1}}}$ as

$$
\Xi_{\Phi_j}^{[p]} \coloneqq (\hat{V}_{k_\Phi}^{[p]})^\top\hat{\phi}'_{k_\Phi}\overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Phi-1}}\hat{V}_l^\top\hat{\phi}'_l, \tag{5.28a}
$$

$$
\Lambda_{\Phi_j}^{[p]} \coloneqq \Xi_{\Phi_j}^{[p]}\left(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top\right), \tag{5.28b}
$$

for $p \in \{1,2\}$, with $\Xi^{[p]}_{\Phi_{k_\Phi}} \coloneqq 1$, $\Lambda^{[1]}_{\Phi_0} \coloneqq \Xi^{[1]}_{\Phi_0}\left(I_{L_{\Phi_1}} \otimes x_h^\top\right)$, $\Lambda^{[2]}_{\Phi_0} \coloneqq \Xi^{[2]}_{\Phi_0}\left(I_{L_{\Phi_1}} \otimes x_h^\top\right)$, $\Lambda^{[1]}_{\Phi_{k_\Phi}} \coloneqq \left(I_{(n-m)} \otimes \hat{\phi}^\top_{k_\Phi}\right) \in \mathbb{R}^{(n-m)\times(n-m)L_{k_\Phi}}$, and $\Lambda^{[2]}_{\Phi_{k_\Phi}} \coloneqq \left(I_m \otimes \hat{\phi}^\top_{k_\Phi}\right) \in \mathbb{R}^{m\times mL_{k_\Phi}}$, the expression of the approximation errors can be simplified as

$$\tilde{\Phi}^{[1]}_{k_\Phi} = \Lambda^{[1]}_{\Phi_{k_\Phi}} \operatorname{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right) + \Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=0}^{k_\Phi-1} \Lambda^{[1]}_{\Phi_j} \operatorname{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi-1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} \tag{5.29a}$$

$$\tilde{\Phi}^{[2]}_{k_\Phi} = \Lambda^{[2]}_{\Phi_{k_\Phi}} \operatorname{vec}\left(\tilde{V}^{[2]}_{k_\Phi}\right) + \Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=0}^{k_\Phi-1} \Lambda^{[2]}_{\Phi_j} \operatorname{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi-1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j}. \tag{5.29b}$$

5.2.2 Approximation error of the Control Effectiveness DNN

As detailed in Section 3.1.2, when a network is used for approximating a matrix, it can be treated as a multi-head DNN, with each head sharing the input, all the hidden layers, and being the approximation of a column (see Figure 3.4).

Similarly to what has been done in the previous section, the approximation error between Ψ and $\hat{\Psi}$, can be computed for each layer $j \in \{0, 1, \ldots, k_\Psi\}$. To take into account the fact that the last layer ($j = k_\Psi$) approximates multiple columns, the error associated to this last one is expressed for each column $i \in \{1, 2, \ldots, m\}$.

For $j = \{1, 2, \ldots, k_\Psi - 1\}$, one can express the approximation error as

$$\begin{aligned}
\tilde{\Psi}_j &= \Psi_j - \hat{\Psi}_j \\
&= U_j^\top \psi_j(\Psi_{j-1}) - \hat{U}_j^\top \psi_j(\hat{\Psi}_{j-1}) \\
&= U_j^\top \psi_j(\Psi_{j-1}) - \hat{U}_j^\top \psi_j(\hat{\Psi}_{j-1}) + U_j^\top \psi_j(\hat{\Psi}_{j-1}) - U_j^\top \psi_j(\hat{\Psi}_{j-1}) \\
&= \left(U_j^\top - \hat{U}_j^\top\right) \psi_j(\hat{\Psi}_{j-1}) + U_j^\top \left(\psi_j(\Psi_{j-1}) - \psi_j(\hat{\Psi}_{j-1})\right) \\
&= \tilde{U}_j^\top \psi_j(\hat{\Psi}_{j-1}) + U_j^\top \left(\psi_j(\Psi_{j-1}) - \psi_j(\hat{\Psi}_{j-1})\right).
\end{aligned} \tag{5.30}$$

where $\tilde{U}_j = U_j - \hat{U}_j$. For $j = 0$, it holds that

$$\begin{aligned}
\tilde{\Psi}_0 &= \Psi_0 - \hat{\Psi}_0 \\
&= U_0^\top x_h - \hat{U}_0^\top x_h \\
&= \tilde{U}_0^\top x_h.
\end{aligned} \tag{5.31}$$

Since $\psi_j(\Psi_{j-1})$ is unknown, it is approximated using Taylor expansion centered around the estimated vector $\hat{\Psi}_{j-1}$, obtaining

$$\psi_j(\Psi_{j-1}) = \psi_j(\hat{\Psi}_{j-1}) + \left.\frac{\partial \psi_j}{\partial \Psi}\right|_{\Psi=\hat{\Psi}_{j-1}} (\Psi_{j-1} - \hat{\Psi}_{j-1}) + \mathrm{O}^2(\tilde{\Psi}_{j-1}), \tag{5.32}$$

with $\frac{\partial \psi_j}{\partial \Psi}\Big|_{\Psi=\hat{\Psi}_{j-1}} \in \mathbb{R}^{L_{\Psi_j} \times L_{\Psi_j}}$ being the Jacobian matrix of the activation function vector $\psi_j(\cdot)$ with respect to its argument, computed in the estimated output of the precious layer $\hat{\Psi}_{j-1}$. As for $\mathrm{O}^2(\tilde{\Psi}_{j-1}) \in \mathbb{R}^{L_{\Psi_j}}$, it contains the lumped terms of order higher than one.

If one substitutes (5.32) in (5.30), and defines $\psi_j = \psi_j(\Psi_{j-1})$, $\hat{\psi}_j = \psi_j(\hat{\Psi}_{j-1})$, and $\hat{\psi}'_j = \frac{\partial \psi_j}{\partial \Psi}\Big|_{\Psi=\hat{\Psi}_{j-1}}$ for sake of readability, it holds that

$$\begin{aligned}
\tilde{\Psi}_j &= \tilde{U}_j^\top \hat{\psi}_j + U_j^\top (\psi_j - \hat{\psi}_j) \\
&= \tilde{U}_j^\top \hat{\psi}_j + U_j^\top \left(\hat{\psi}_j + \hat{\psi}'_j \tilde{\Psi}_{j-1} + \mathrm{O}^2(\tilde{\Psi}_{j-1}) - \hat{\psi}_j \right) \\
&= \tilde{U}_j^\top \hat{\psi}_j + U_j^\top \left(\hat{\psi}'_j \tilde{\Psi}_{j-1} + \mathrm{O}^2(\tilde{\Psi}_{j-1}) \right) \\
&= \tilde{U}_j^\top \hat{\psi}_j + U_j^\top \hat{\psi}'_j \tilde{\Psi}_{j-1} + U_j^\top \mathrm{O}^2(\tilde{\Psi}_{j-1}) \\
&= \tilde{U}_j^\top \hat{\psi}_j + (\tilde{U}_j^\top + \hat{U}_j^\top) \hat{\psi}'_j \tilde{\Psi}_{j-1} + U_j^\top \mathrm{O}^2(\tilde{\Psi}_{j-1}) \\
&= \tilde{U}_j^\top \hat{\psi}_j + \tilde{U}_j^\top \hat{\psi}'_j \tilde{\Psi}_{j-1} + \hat{U}_j^\top \hat{\psi}'_j \tilde{\Psi}_{j-1} + U_j^\top \mathrm{O}^2(\tilde{\Psi}_{j-1}) \\
&= \tilde{U}_j^\top \hat{\psi}_j + \hat{U}_j^\top \hat{\psi}'_j \tilde{\Psi}_{j-1} + \Delta_{\Psi_j},
\end{aligned} \tag{5.33}$$

with $\Delta_{\Psi_j} \coloneqq \tilde{U}_j^\top \hat{\psi}'_j \tilde{\Psi}_{j-1} + U_j^\top \mathrm{O}^2(\tilde{\Psi}_{j-1}) \in \mathbb{R}^{L_{\Psi_{j+1}}}$.

Since $\tilde{U}_j^\top \hat{\psi}_j \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}}$, one can write

$$\tilde{U}_j^\top \hat{\psi}_j = \mathrm{vec}\left(\tilde{U}^\top \hat{\psi}_j \right) = \mathrm{vec}\left(\hat{\psi}_j^\top \tilde{U}_j \right) = \mathrm{vec}\left(\hat{\psi}_j^\top \tilde{U}_j I_{L_{\Psi_{j+1}}} \right),$$

and, applying Lemma 5.1 with $A \equiv \hat{\psi}_j^\top$, $B \equiv \tilde{U}_j$, and $C \equiv I_{L_{\Psi_{j+1}}}$ it holds that

$$\tilde{U}_j^\top \hat{\psi}_j = \left(I_{L_{\Psi_{j+1}}}^\top \otimes \hat{\psi}_j^\top \right) \mathrm{vec}\left(\tilde{U}_j \right) = \left(I_{L_{\Psi_{j+1}}} \otimes \hat{\psi}_j^\top \right) \mathrm{vec}\left(\tilde{U}_j \right).$$

Hence, expression (5.33) can be rewritten as

$$\begin{aligned}
\tilde{\Psi}_j &= \tilde{U}_j^\top \hat{\psi}_j + \hat{U}_j^\top \hat{\psi}'_j \tilde{\Psi}_{j-1} + \Delta_{\Psi_j} \\
&= \left(I_{L_{\Psi_{j+1}}} \otimes \hat{\psi}_j^\top \right) \mathrm{vec}\left(\tilde{U}_j \right) + \hat{U}_j^\top \hat{\psi}'_j \tilde{\Psi}_{j-1} + \Delta_{\Psi_j}.
\end{aligned} \tag{5.34}$$

In the case $j = 0$, one has $\psi_0 = \hat{\psi}_0 = x_h$. Hence, the Taylor expansion is not employed and it holds that

$$\begin{aligned}
\tilde{\Psi}_0 &= \tilde{U}_0^\top x_h \\
&= \mathrm{vec}\left(\tilde{U}_0^\top x_h \right) \\
&= \left(I_{L_{\Psi_1}} \otimes x_h^\top \right) \mathrm{vec}\left(\tilde{U}_0 \right).
\end{aligned} \tag{5.35}$$

As anticipated, for the last layer ($j = k_\Psi$), the error is expressed for each column $i \in \{1, 2, \ldots, m\}$. In particular, having $\Psi_{k_\Psi}^{[i]}$ and $\hat{\Psi}_{k_\Psi}^{[i]}$ defined as in (5.14c) and (5.17c), respectively, the error associated with the i-th column is given by

$$\tilde{\Psi}_{k_\Psi}^{[i]} = \Psi_{k_\Psi}^{[i]} - \hat{\Psi}_{k_\Psi}^{[i]}$$

$$
\begin{aligned}
&= (U_{k_\Psi}^{[i]})^\top \psi_{k_\Psi} - (\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}_{k_\Psi} \\
&= (U_{k_\Psi}^{[i]})^\top \psi_j - (\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}_j + (U_{k_\Psi}^{[i]})^\top \hat{\psi}_j - (U_{k_\Psi}^{[i]})^\top \hat{\psi}_j \\
&= \left((U_{k_\Psi}^{[i]})^\top - (\hat{U}_{k_\Psi}^{[i]})^\top\right) \hat{\psi}_{k_\Psi} + (U_{k_\Psi}^{[i]})^\top \left(\psi_{k_\Psi} - \hat{\psi}_{k_\Psi}\right) \\
&= \tilde{U}_{k_\Psi}^{[i]^\top} \hat{\psi}_{k_\Psi} + (U_{k_\Psi}^{[i]})^\top \left(\psi_{k_\Psi} - \hat{\psi}_{k_\Psi}\right) \\
&= \tilde{U}_{k_\Psi}^{[i]^\top} \hat{\psi}_{k_\Psi} + (\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}'_{k_\Psi} \tilde{\Psi}_{k_\Psi - 1} + \Delta_{\Psi_{k_\Psi}}^{[i]} \\
&= \left(I_m \otimes \hat{\psi}_{k_\Psi}{}^\top\right) \mathrm{vec}\left(\tilde{U}_{k_\Phi}^{[i]}\right) + (\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}'_{k_\Psi} \tilde{\Psi}_{k_\Psi - 1} + \Delta_{\Psi_{k_\Psi}}^{[i]},
\end{aligned} \tag{5.36}
$$

with $\Delta_{\Psi_{k_\Psi}}^{[i]} \coloneqq (\tilde{U}_{k_\Psi}^{[i]})^\top \hat{\psi}'_{k_\Psi} \tilde{\Psi}_{k_\Psi - 1} + (U_{k_\Psi}^{[i]})^\top \mathrm{O}^2 \left(\tilde{\Psi}_{k_\Psi - 1}\right) \in \mathbb{R}^m$.

Then, analogously as done for $\tilde{\Phi}_j$, recursively substituting (5.34) in (5.36) until $\tilde{\Psi}_0$ and collecting the common terms, one obtains

$$
\begin{aligned}
\tilde{\Psi}_{k_\Psi}^{[i]} = \sum_{j=0}^{k_\Psi - 1} \left((\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}'_{k_\Psi}\right) \left(\overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Psi - 1}} \hat{U}_l^\top \hat{\psi}'_l\right) \left(I_{L_{\Psi_{j+1}}} \otimes \hat{\psi}_j^\top\right) \mathrm{vec}\left(\tilde{U}_j\right) + \Delta_{k_\Psi}^{[i]} + \\
+ \left(I_m \otimes \hat{\psi}_{k_\Psi}^\top\right) \mathrm{vec}\left(\tilde{U}_{k_\Phi}^{[i]}\right) + \sum_{j=1}^{k_\Psi - 1} \left((\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}'_{k_\Psi}\right) \left(\overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Psi - 1}} \hat{U}_l{}^\top \hat{\psi}'_l\right) \Delta_{\Psi_j},
\end{aligned}
$$

for $i \in \{1, 2, \ldots, m\}$. Moreover, defining the matrices $\Xi_{\Psi_j}^{[i]} \in \mathbb{R}^{m \times L_{\Psi_{j+1}}}$ and $\Lambda_{\Psi_j}^{[i]} \in \mathbb{R}^{m \times L_{\Psi_j} L_{\Psi_{j+1}}}$ as

$$
\Xi_{\Psi_j}^{[i]} \coloneqq (\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}'_{k_\Psi} \overset{\curvearrowleft}{\prod_{l=j+1}^{k_\Psi - 1}} \hat{U}_l^\top \hat{\psi}'_l, \tag{5.37a}
$$

$$
\Lambda_{\Psi_j}^{[i]} \coloneqq \Xi_{\Psi_j}^{[i]} \left(I_{L_{\Psi_{j+1}}} \otimes \hat{\psi}_j^\top\right), \tag{5.37b}
$$

with $\Xi_{\Psi_{k_\Psi}}^{[i]} \coloneqq 1$, $\Xi_{\Psi_{k_\Psi - 1}}^{[i]} \coloneqq (\hat{U}_{k_\Psi}^{[i]})^\top \hat{\psi}'_{k_\Psi}$ (see Definition 5.1), $\Lambda_{\Psi_0}^{[i]} \coloneqq \Xi_{\Psi_0}^{[i]} \left(I_{L_{\Psi_1}} \otimes x_h^\top\right)$, and $\Lambda_{\Psi_{k_\Psi}}^{[i]} \coloneqq \left(I_m \otimes \hat{\psi}_{k_\Psi}^\top\right) \in \mathbb{R}^{m \times m L_{k_\Psi}}$, equation (5.37b) can be rewritten as

$$
\tilde{\Psi}_{k_\Psi}^{[i]} = \Lambda_{\Psi_{k_\Psi}}^{[i]} \mathrm{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right) + \Delta_{\Psi_{k_\Psi}}^{[i]} + \sum_{j=0}^{k_\Psi - 1} \Lambda_{\Psi_j}^{[i]} \mathrm{vec}\left(\tilde{U}_j\right) + \sum_{j=1}^{k_\Psi - 1} \Xi_{\Psi_j}^{[i]} \Delta_{\Psi_j} \tag{5.38}
$$

5.3 The DNN-ISM Control Strategy

In the following, the DNN-ISM control scheme, developed in [99, 100, 101] and depicted in Figure 5.1, is presented and analyzed.

Considering the system in (5.3), and inspired by the ISM control framework, the integral sliding variable is defined as in

$$
\sigma(x(t)) = \sigma_0(x(t)) - \hat{z}(x(t)), \tag{5.39}
$$

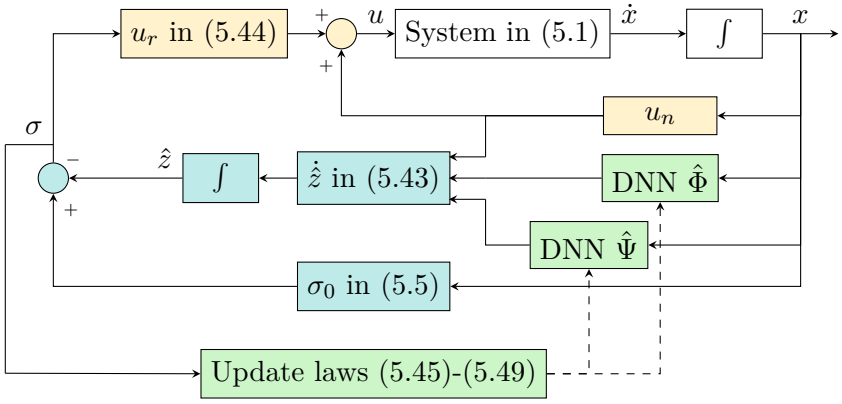


Figure 5.1: Block diagram of the DNN-ISM control scheme. The blocks associated with the DNNs, the sliding variable, and the control law are highlighted in green, blue, and yellow, respectively.

where $\sigma_0 : \mathcal{X} \to \mathbb{R}^m$ is the conventional sliding mode defined as in (5.5), while $\hat{z} : \mathcal{X} \to \mathbb{R}^m$ is the estimate of the transient variable in (5.6). In particular, this last one is computed relying on the estimate of the dynamics of f_1, f_2, and $\bar{B}$, having

$$\hat{z}(x(t)) = \sigma_0(x(t_0)) + \int_{t_0}^{t} \Big\{ C_1 \Big[\hat{f}_1(x(\tau),\tau) - \dot{x}_1^\star(\tau) \Big] + \\ + C_2 \Big[\hat{f}_2(x(\tau),\tau) + \hat{\bar{B}}(x(\tau),\tau) u_n(\tau) - \dot{x}_2^\star(\tau) \Big] \Big\} d\tau. \tag{5.40}$$

Proposition 5.1. *Given a matrix $A \in \mathbb{R}^{n\times m}$ and a vector $b \in \mathbb{R}^m$, then the product $Ab \in \mathbb{R}^n$ can be computed as*

$$Ab = \sum_{i=1}^{m} A^{(i)} b_i,$$

where $A^{(i)} \in \mathbb{R}^n$ denotes the i-th column of A and $b_i \in \mathbb{R}$ the i-th element of b.

Exploiting Proposition 5.1, the above expression of $\hat{z}(x(t))$ can be rewritten as

$$\hat{z}(x(t)) = \sigma_0(x(t_0)) + \int_{t_0}^{t} \Big\{ C_1 \Big[\hat{f}_1(x(\tau),\tau) - \dot{x}_1^\star(\tau) \Big] + \\ + C_2 \Big[\hat{f}_2(x(\tau),\tau) + \sum_{i=1}^{m} \hat{\bar{B}}^{(i)}(x(\tau),\tau) u_{n,i}(\tau) - \dot{x}_2^\star(\tau) \Big] \Big\} d\tau. \tag{5.41}$$

Then, substituting (5.17), one has that

$$\hat{z}(x(t)) = \sigma_0(x(t_0)) + \int_{t_0}^{t} \Big\{ C_1 \Big[\hat{\Phi}_{k_\Phi}^{[1]} - \dot{x}_1^\star(\tau) \Big] + \\ + C_2 \Big[\hat{\Phi}_{k_\Phi}^{[2]} + \sum_{i=1}^{m} \hat{\Psi}_{k_\Psi}^{[i]} u_{n,i}(\tau) - \dot{x}_2^\star(\tau) \Big] \Big\} d\tau \tag{5.42}$$

where $\hat{\Phi}^{[1]}_{k_\Phi} \equiv \hat{\Phi}^{[1]}(x(\tau)) \in \mathbb{R}^{n-m}$, $\hat{\Phi}^{[2]}_{k_\Phi} \equiv \hat{\Phi}^{[2]}(x(\tau)) \in \mathbb{R}^{m}$, and $\hat{\Psi}^{[i]}_{k_\Psi} \equiv \hat{\Psi}^{[i]}(x(\tau)) \in \mathbb{R}^{m}$. The dynamics of the estimate of the transient variable can be then expressed as implying that

$$\dot{\hat{z}}(x(t)) = C_1\Big[\hat{\Phi}^{[1]}_{k_\Phi} - \dot{x}^\star_1(t)\Big] + C_2\Big[\hat{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_{n,i}(t) - \dot{x}^\star_2(t)\Big], \tag{5.43}$$

with initial conditions $z(x(t_0)) = \sigma_0(x(t_0))$. The matrices $C_1 \in \mathbb{R}^{m\times(n-m)}$ and $C_2 \in \mathbb{R}^{m\times m}$ are chosen as described in Section 5.1. In particular, the latter satisfies Assumption 5.4.

Similarly to the ideal case, the overall control law $u(t) \in \mathbb{R}^m$ is the one in (5.8), where the switching law $u_r(t)$ is chosen according to the unit vector approach

$$u_r(t) = -\rho\frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \tag{5.44}$$

with $\rho \in \mathbb{R}_{>0}$ being the discontinuous control gain. The choice of the nominal control law $u_n(t) \in \mathbb{R}^m$ depends on the performance that one wants to obtain. Some hints on the selection of this last one will be provided later in this chapter.

One fundamental aspect in the design of the DNN-ISM control scheme is the choice of the adaptation laws for the layers of the DNNs $\hat{\Phi}$ and $\hat{\Psi}$. Such laws, proposed in the following, depend on the integral sliding variable σ and are derived according to the analysis in Section 5.3.2.

For what concerns the DNN $\hat{\Phi}$, the layers up to the penultimate one, i.e., for $j \in \{0, 1, \dots, k_\Phi - 1\}$, the adaptation is done through

$$\mathrm{vec}\left(\dot{\hat{V}}_j\right) = \underset{\mathcal{B}_{\Phi_j}}{\mathrm{proj}}\left(\Gamma_{\Phi_j}\Lambda^\top_{\Phi_j}C^\top\sigma\right) \in \mathbb{R}^{L_{\Phi_j}L_{\Phi_{j+1}}}, \tag{5.45}$$

where $C = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \in \mathbb{R}^{m\times n}$, $\Lambda_{\Phi_j} \in \mathbb{R}^{n\times L_{\Phi_j}L_{\Phi_{j+1}}}$ is the one defined in (5.26b) and it corresponds to

$$\Lambda_{\Phi_j} = \begin{bmatrix} \Lambda^{[1]}_{\Phi_j} \\ \Lambda^{[2]}_{\Phi_j} \end{bmatrix}, \tag{5.46}$$

with $\Lambda^{[1]}_{\Phi_j} \in \mathbb{R}^{(n-m)\times L_{\Phi_j}L_{\Phi_{j+1}}}$ and $\Lambda^{[2]}_{\Phi_j} \in \mathbb{R}^{m\times L_{\Phi_j}L_{\Phi_{j+1}}}$ being the ones defined in (5.28). As for $\Gamma_{\Phi_j} \in \mathbb{R}^{L_{\Phi_j}L_{\Phi_{j+1}}\times L_{\Phi_j}L_{\Phi_{j+1}}}$, it is the adaptation rate matrix, defined as diagonal with positive entries, while $\mathrm{proj}(\cdot)$ is the projection operator defined as in Appendix A, with set $\mathcal{B}_{\Phi_j}$ defined later in Section 5.3.1. The weights sub-matrices associated with the last layer ($j = k_\Phi$) are adjusted according to

$$\mathrm{vec}\left(\dot{\hat{V}}^{[1]}_{k_\Phi}\right) = \underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\mathrm{proj}}\left(\Gamma^{[1]}_{\Phi_{k_\Phi}}(\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top C_1^\top\sigma\right) \in \mathbb{R}^{L_{k_\Phi}(n-m)}, \tag{5.47a}$$

$$\operatorname{vec}\left(\dot{\hat{V}}_{k_\Phi}^{[2]}\right) = \underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\operatorname{proj}}\left(\Gamma_{\Phi_{k_\Phi}}^{[2]}(\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_2^\top \sigma\right) \in \mathbb{R}^{L_{k_\Phi} m}, \tag{5.47b}$$

where $\Gamma_{\Phi_{k_\Phi}}^{[1]} \in \mathbb{R}^{(n-m)L_{k_\Phi}\times(n-m)L_{k_\Phi}}$ and $\Gamma_{\Phi_{k_\Phi}}^{[2]} \in \mathbb{R}^{mL_{k_\Phi}\times mL_{k_\Phi}}$ are diagonal matrices with positive entries, while $\Lambda_{\Phi_{k_\Phi}}^{[1]} \in \mathbb{R}^{(n-m)\times(n-m)L_{k_\Phi}}$ and $\Lambda_{\Phi_{k_\Phi}}^{[2]} \in \mathbb{R}^{m\times mL_{k_\Phi}}$ are the ones defined below equation (5.28).

As for the DNN $\hat{\Psi}$, for $j \in \{0, 1, \dots, k_\Psi - 1\}$, one has that

$$\operatorname{vec}\left(\dot{\hat{U}}_j\right) = \underset{\mathcal{B}_{\Psi_j}}{\operatorname{proj}}\left(\Gamma_{\Psi_j}\left(\sum_{i=1}^{m} u_{n,i}(\Lambda_{\Psi_j}^{[i]})^\top\right) C_2^\top \sigma\right) \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}}, \tag{5.48}$$

where $\Lambda_{\Psi_j}^{[i]} \in \mathbb{R}^{m\times L_{\Psi_j} L_{\Psi_{j+1}}}$ is defined as in (5.37b), $u_{n,i} \in \mathbb{R}$ is the i-th component of the nominal control law, while $\Gamma_{\Psi_j} \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}\times L_{\Psi_j} L_{\Psi_{j+1}}}$ is the adaptation rate matrix, chosen as diagonal with positive entries. The weight sub-matrices of last layer of $\hat{\Psi}$ are adapted according to

$$\operatorname{vec}\left(\dot{\hat{U}}_{k_\Psi}^{[i]}\right) = \underset{\mathcal{B}_{\Psi_{k_\Psi}}}{\operatorname{proj}}\left(\Gamma_{\Psi_{k_\Psi}} u_{n,i}(\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top C_2^\top \sigma\right) \in \mathbb{R}^{L_{k_\Psi} m}, \tag{5.49}$$

with $i \in \{1, 2, \dots, m\}$, where $\Gamma_{\Psi_{k_\Psi}} \in \mathbb{R}^{mL_{k_\Psi}\times mL_{k_\Psi}}$ is the diagonal matrix of the adaptation rates, while $\Lambda_{\Psi_{k_\Psi}}^{[i]} \in \mathbb{R}^{m\times mL_{k_\Psi}}$ is the one defined below equation (5.37).

For sake of completeness, it is possible to express the time evolution of the weight matrices of the DNNs employed for the approximation of the dynamics. In particular, for $\hat{\Phi}$ one has

$$\begin{aligned}
V_j(t) &= V_j(t_0) + \int_{t_0}^{t} \operatorname{vec}^{-1}\left(\underset{\mathcal{B}_{\Phi_j}}{\operatorname{proj}}\left(\Gamma_{\Phi_j}\left(\Lambda_{\Phi_j}(x(\tau))\right)^\top C^\top \sigma(x(\tau))\right)\right) d\tau,\\
V_{k_\Phi}^{[1]}(t) &= V_{k_\Phi}^{[1]}(t_0) + \int_{t_0}^{t} \operatorname{vec}^{-1}\left(\underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\operatorname{proj}}\left(\Gamma_{\Phi_{k_\Phi}}^{[1]}\left(\Lambda_{\Phi_{k_\Phi}}^{[1]}(x(\tau))\right)^\top C_1^\top \sigma(x(\tau))\right)\right) d\tau,\\
V_{k_\Phi}^{[2]}(t) &= V_{k_\Phi}^{[2]}(t_0) + \int_{t_0}^{t} \operatorname{vec}^{-1}\left(\underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\operatorname{proj}}\left(\Gamma_{\Phi_{k_\Phi}}^{[2]}\left(\Lambda_{\Phi_{k_\Phi}}^{[2]}(x(\tau))\right)^\top C_2^\top \sigma(x(\tau))\right)\right) d\tau.
\end{aligned}$$

As for $\hat{\Psi}$, the evolution of the weights is given by

$$\begin{aligned}
U_j(t) &= U_j(t_0) + \int_{t_0}^{t} \operatorname{vec}^{-1}\left(\underset{\mathcal{B}_{\Psi_j}}{\operatorname{proj}}\left(\Gamma_{\Psi_j}\left(\sum_{i=1}^{m} u_{n,i}\left(\Lambda_{\Psi_j}^{[i]}(x(\tau))\right)^\top\right) C_2^\top \sigma(x(\tau))\right)\right) d\tau,\\
U_{k_\Psi}^{[i]}(t) &= U_{k_\Psi}^{[i]}(t_0) + \int_{t_0}^{t} \operatorname{vec}^{-1}\left(\underset{\mathcal{B}_{\Psi_{k_\Psi}}}{\operatorname{proj}}\left(\Gamma_{\Psi_{k_\Psi}} u_{n,i}\left(\Lambda_{\Psi_{k_\Psi}}^{[i]}(x(\tau))\right)^\top C_2^\top \sigma(x(\tau))\right)\right) d\tau.
\end{aligned}$$

The aim of the projection operator, defined as in Appendix A, in the weight adaptation laws is to maintain the weights in the admissible sets $\mathcal{B}_{\Phi_j} \subset \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}}$, with $j \in \{0, 1, \dots, k_\Phi\}$, and $\mathcal{B}_{\Psi_j} \subset \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}}$, with $j \in \{0, 1, \dots, k_\Psi\}$, which are defined in the following.

5.3.1 Use of Parameter Projection and its effects

The ideal weights of the DNNs are assumed to be bounded in norm by some known constants $\bar{V}, \bar{U} \in \mathbb{R}_{>0}$ that depends on some superficial knowledge of the system, such as the bounds of f_1, f_2, and $\bar{B}$. In particular, recalling Assumption 5.6, it holds that

$$\sup_{j\in\{0,1,\dots,k_\Phi\}} \|V_j\| \leq \bar{V}, \qquad \sup_{j\in\{0,1,\dots,k_\Psi\}} \|U_j\| \leq \bar{U}.$$

Such a knowledge about the DNNs can be exploited to limit the evolution of the estimated weights during their adaptation in two sets $\mathcal{B}_{\Phi_j} \subset \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}}$ and $\mathcal{B}_{\Psi_j} \subset \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}}$. Such sets can be conveniently defines as

$$\mathcal{B}_{\Phi_j} \coloneqq \left\{ \operatorname{vec}\left(\hat{V}_j\right) \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}} : \left\| \operatorname{vec}\left(\hat{V}_j\right) \right\| - \bar{V} \leq 0 \right\}, \tag{5.50a}$$

$$\mathcal{B}_{\Psi_j} \coloneqq \left\{ \operatorname{vec}\left(\hat{U}_j\right) \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}} : \left\| \operatorname{vec}\left(\hat{U}_j\right) \right\| - \bar{U} \leq 0 \right\}. \tag{5.50b}$$

Following the notation in Appendix A, and specifically equation (A.1), one can define the underlying functions $\mathcal{P}_{\Phi_j} : \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}} \to \mathbb{R}$ and $\mathcal{P}_{\Psi_j} : \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}} \to \mathbb{R}$ as

$$\mathcal{P}_{\Phi_j}\left(\operatorname{vec}\left(\hat{V}_j\right)\right) \coloneqq \left\| \operatorname{vec}\left(\hat{V}_j\right) \right\| - \bar{V}, \tag{5.51a}$$

$$\mathcal{P}_{\Psi_j}\left(\operatorname{vec}\left(\hat{U}_j\right)\right) \coloneqq \left\| \operatorname{vec}\left(\hat{U}_j\right) \right\| - \bar{U}. \tag{5.51b}$$

Such functions are characterized by gradients $\frac{\partial \mathcal{P}_{\Phi_j}}{\partial \hat{V}_j} \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}}$ and $\frac{\partial \mathcal{P}_{\Psi_j}}{\partial \hat{U}_j} \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}}$, which can be computed as follows.

Denoting with $\hat{V}_{j,(l,h)}$ the element of $\hat{V}_j$ in the l-th row and h-th column, one has that

$$\mathcal{P}_{\Phi_j}\left(\hat{V}_j\right) \coloneqq \left\| \operatorname{vec}\left(\hat{V}_j\right) \right\| - \bar{V} = \sqrt{\sum_{l=1}^{L_{\Phi_j}} \sum_{h=1}^{L_{\Phi_{j+1}}} \hat{V}_{j,(l,h)}^2} - \bar{V},$$

which, differentiating element by element yields the partial derivatives

$$\begin{aligned}\frac{\partial \mathcal{P}_{\Phi_j}\left(\hat{V}_j\right)}{\partial \hat{V}_{j,(c,r)}} &= \frac{1}{2\sqrt{\sum_{l=1}^{L_{\Phi_j}} \sum_{h=1}^{L_{\Phi_{j+1}}} \hat{V}_{j,(l,h)}^2}} \left(2\hat{V}_{j,(c,r)}\right) \\ &= \frac{\hat{V}_{j,(c,r)}}{\sqrt{\sum_{l=1}^{L_{\Phi_j}} \sum_{h=1}^{L_{\Phi_{j+1}}} \hat{V}_{j,(l,h)}^2}} = \frac{\hat{V}_{j,(c,r)}}{\left\| \operatorname{vec}\left(\hat{V}_j\right) \right\|}.\end{aligned}$$

Finally, gathering all the partial derivatives, the gradient vector results in

$$\frac{\partial \mathcal{P}_{\Phi_j}}{\partial \hat{V}_j} = \nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j} = \frac{\operatorname{vec}\left(\hat{V}_j\right)}{\left\| \operatorname{vec}\left(\hat{V}_j\right) \right\|}. \tag{5.52}$$

Performing the same steps, the gradient vector $\frac{\partial \mathcal{P}_{\Psi_j}}{\partial \hat{U}_j}$ can be computed as

$$\frac{\partial \mathcal{P}_{\Psi_j}}{\partial \hat{U}_j} = \nabla_{\hat{U}_j} \mathcal{P}_{\Psi_j} = \frac{\operatorname{vec}\left(\hat{U}_j\right)}{\left\|\operatorname{vec}\left(\hat{U}_j\right)\right\|}. \tag{5.53}$$

According to the definition (A.4) of the smooth projection operator, one has

$$\underset{\mathcal{B}_{\Phi_j}}{\operatorname{proj}}(\tau) = \begin{cases} \tau, & \text{if } \hat{V}_j \in \mathcal{B}^{\mathrm{o}}_{\Phi_j} \text{ or } \nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j} \cdot \tau \leq 0 \\ \left(I - c(\hat{V}_j) \Theta \frac{\nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j} (\nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j})^\top}{(\nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j})^\top \Theta \nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j}} \right) \tau & \text{if } \hat{V}_j \in \mathcal{B}^{\alpha}_{\Phi_j} \setminus \mathcal{B}^{\mathrm{o}}_{\Phi_j} \text{ and } \nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j} \cdot \tau > 0, \end{cases}$$

where, according to the definitions in Appendix A, $\mathcal{B}^{\mathrm{o}}_{\Phi_j}$ and $\mathcal{B}^{\alpha}_{\Phi_j}$ denote, respectively, the interior of $\mathcal{B}_{\Phi_j}$ and the α-boundary layer around $\mathcal{B}_{\Phi_j}$. As for the matrix $\Theta \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}} \times L_{\Phi_j} L_{\Phi_{j+1}}}$ here is the counterpart of Γ used in equation (A.4) to avoid confusion with the learning rate matrices. By setting $\Theta = \gamma I_{L_{\Phi_j} L_{\Phi_{j+1}}}$, with $\gamma \in \mathbb{R}_{>0}$, and substituting (5.52) it holds that

$$\begin{aligned} \underset{\mathcal{B}_{\Phi_j}}{\operatorname{proj}}(\tau) &= \begin{cases} \tau, \\ \left(I - c(\hat{V}_j) \frac{\gamma \nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j} (\nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j})^\top}{\gamma (\nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j})^\top \nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j}} \right) \tau, \end{cases} \\ &= \begin{cases} \tau, \\ \left(I - c(\hat{V}_j) \frac{\nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j} (\nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j})^\top}{\left\| \nabla_{\hat{V}_j} \mathcal{P}_{\Phi_j} \right\|^2} \right) \tau, \end{cases} \end{aligned}$$

Then, substituting (5.52), the above equation becomes

$$\underset{\mathcal{B}_{\Phi_j}}{\operatorname{proj}}(\tau) = \begin{cases} \tau, & \text{if } \hat{V}_j \in \mathcal{B}^{\mathrm{o}}_{\Phi_j} \text{ or } \operatorname{vec}\left(\hat{V}_j\right) \cdot \tau \leq 0 \\ \left(I - c(\hat{V}_j) \frac{\operatorname{vec}(\hat{V}_j) \operatorname{vec}(\hat{V}_j)^\top}{\left\| \operatorname{vec}(\hat{V}_j) \right\|^2} \right) \tau & \text{if } \hat{V}_j \in \mathcal{B}^{\alpha}_{\Phi_j} \setminus \mathcal{B}^{\mathrm{o}}_{\Phi_j} \text{ and } \operatorname{vec}\left(\hat{V}_j\right) \cdot \tau > 0. \end{cases} \tag{5.54}$$

Following the same steps and substituting (5.53), the smooth projection operator for the weight matrices $\hat{U}_j$ analogously as

$$\underset{\mathcal{B}_{\Psi_j}}{\operatorname{proj}}(\tau) = \begin{cases} \tau & \text{if } \hat{U}_j \in \mathcal{B}^{\mathrm{o}}_{\Psi_j} \text{ or } \operatorname{vec}\left(\hat{U}_j\right) \cdot \tau \leq 0 \\ \left(I - c(\hat{U}_j) \frac{\operatorname{vec}(\hat{U}_j) \operatorname{vec}(\hat{U}_j)^\top}{\left\| \operatorname{vec}(\hat{U}_j) \right\|^2} \right) \tau & \text{if } \hat{U}_j \in \mathcal{B}^{\alpha}_{\Psi_j} \setminus \mathcal{B}^{\mathrm{o}}_{\Psi_j} \text{ and } \operatorname{vec}\left(\hat{U}_j\right) \cdot \tau > 0. \end{cases} \tag{5.55}$$

Lemma 5.2 (Boundedness of the DNN weights and biases). *Let $\mathcal{B}_{\Phi_j}$ and $\mathcal{B}_{\Psi_j}$ be the admissible sets defined as in* (5.50)*, while $proj_{\mathcal{B}_{\Phi_j}}$ and $proj_{\mathcal{B}_{\Psi_j}}$ are the smooth projection operators defined as in* (5.54) *and* (5.55)*, respectively. Moreover,*

consider the weight adaptation laws in (5.45), (5.47), (5.48), *and* (5.49). *Then, one has that*

1. $V_j(t) \in \mathcal{B}^{\alpha}_{\Phi_j}$, *for all* $t \geq t_0$ *and* $j \in \{0, 1, \ldots, k_\Phi\}$;
2. $U_j(t) \in \mathcal{B}^{\alpha}_{\Psi_j}$, *for all* $t \geq t_0$ *and* $j \in \{0, 1, \ldots, k_\Psi\}$;
3. $-\text{vec}\left(V_j - \hat{V}_j\right)^\top \Gamma^{-1}_{\Phi_j} \underset{\mathcal{B}_{\Phi_j}}{\text{proj}}\left(\tau\right) \leq -\text{vec}\left(V_j - \hat{V}_j\right)^\top \Gamma^{-1}_{\Phi_j}\tau$, *for all the layers* $j \in \{0, 1, \ldots, k_\Phi\}$;
4. $-\text{vec}\left(U_j - \hat{U}_j\right)^\top \Gamma^{-1}_{\Psi_j} \underset{\mathcal{B}_{\Psi_j}}{\text{proj}}\left(\tau\right) \leq -\text{vec}\left(U_j - \hat{U}_j\right)^\top \Gamma^{-1}_{\Psi_j}\tau$, *for all the layers* $j \in \{0, 1, \ldots, k_\Psi\}$;

Proof. The proof can be easily developed by exploiting the proof of points 3 and 4 of Lemma A.1. □

Before introducing the stability analysis associated with the DNN-ISM control scheme, the following considerations must be made.

The fact that the activation functions of the networks Φ, $\hat{\Phi}$, Ψ, and $\hat{\Psi}$, i.e., $\phi_j(\cdot)$ and $\psi_j(\cdot)$, are chosen differentiable and Lipschitz continuous ensures that their Jacobian matrices are bounded in norm. In particular, there exists some constants $\bar{d}_{\Phi_j}, \bar{d}_{\Psi_p} \in \mathbb{R}_{>0}$ such that

$$\left\|\phi_j'\right\| \leq \bar{d}_{\phi_j}, \quad \left\|\hat{\phi}_j'\right\| \leq \bar{d}_{\phi_j}, \quad \left\|\psi_p'\right\| \leq \bar{d}_{\psi_p}, \quad \left\|\hat{\psi}_p'\right\| \leq \bar{d}_{\psi_p}, \tag{5.56}$$

for all $j \in \{0, 1, \ldots, k_\Phi\}$ and $p \in \{0, 1, \ldots, k_\Psi\}$. Such a fact, along with Lemma 5.2 and with the fact that for classical activation functions (e.g., sigmoid, hyperbolic tangent, etc.) the higher order of the Taylor's expansion are bounded [93], implies that the terms Δ_{Φ_j} and Δ_{Ψ_j}, defined in Section 5.2.1 and Section 5.2.2, respectively, are bounded.

Proposition 5.2. *There exist some constants* $\bar{c}_{\Phi_1}, \bar{c}_{\Phi_2}, \bar{c}_{\Psi} \in \mathbb{R}_{>0}$ *such that the residual terms appearing in* (5.29) *and* (5.38) *as*

$$\left\|\Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j}\right\| \leq \bar{c}_{\Phi_1}, \qquad \left\|\Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j}\right\| \leq \bar{c}_{\Phi_2},$$

$$\left\|\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi - 1} \Xi^{[i]}_{\Psi_j} \Delta_{\Psi_j}\right\| \leq \bar{c}_{\Psi}.$$

5.3.2 Sliding mode Existence

The main theoretical results related to the DNN-ISM control strategy are presented in the following.

Theorem 5.1. *Consider the nonlinear system in* (5.3)*, controlled via ISM control law* u *in* (5.8)*, with switching law* u_r *in* (5.44) *with integral sliding variable* σ *defined as in* (5.39)*. Moreover, the DNNs weights are adapted according to the adaptation laws* (5.45) - (5.49)*. If Assumptions 5.1 - 5.6 and Proposition 5.2 hold, and*

$$\rho > \frac{\|C_1\| \{\bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1}\} + \|C_2\| \{\bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + m(\bar{c}_{\Psi} + \bar{\varepsilon}_{\Psi}) \|u_n\| + \bar{h}\} + \bar{\eta}}{\underline{\lambda}(C_2)\underline{\gamma}},$$

with $\bar{\eta} \in \mathbb{R}_{>0}$*, then,* $\sigma(x(t)) \to 0_m$ *for* $t \to \infty$*.*

Proof. See Appendix B.1. □

The use of the DNNs $\hat{\Phi}$ and $\hat{\Psi}$, which are characterized by estimated weights, introduces unavoidable approximation errors with respect to the real functions. In particular, the error associated with the drift dynamics components is given by

$$\begin{aligned}
f_1 - \hat{\Phi}_{k_\Phi}^{[1]} &= \tilde{\Phi}_{k_\Phi}^{[1]} + \varepsilon_\Phi^{[1]} \\
&= \Lambda_{\Phi_{k_\Phi}}^{[1]} \operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right) + \Delta_{\Phi_{k_\Phi}}^{[1]} + \sum_{j=0}^{k_\Phi - 1} \Lambda_{\Phi_j}^{[1]} \operatorname{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi - 1} \Xi_{\Phi_j}^{[1]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[1]}, \\
f_2 - \hat{\Phi}_{k_\Phi}^{[2]} &= \tilde{\Phi}_{k_\Phi}^{[2]} + \varepsilon_\Phi^{[2]} \\
&= \Lambda_{\Phi_{k_\Phi}}^{[2]} \operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right) + \Delta_{\Phi_{k_\Phi}}^{[2]} + \sum_{j=0}^{k_\Phi - 1} \Lambda_{\Phi_j}^{[2]} \operatorname{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi - 1} \Xi_{\Phi_j}^{[2]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[2]}.
\end{aligned}$$

As for the approximation error related to the control effectiveness term, it can be computed as

$$\begin{aligned}
\bar{B}^{(i)} - \hat{\Psi}_{k_\Psi}^{[i]} &= \tilde{\Psi}_{k_\Psi}^{[i]} + \varepsilon_\Psi^{(i)} \\
&= \Lambda_{\Psi_{k_\Psi}}^{[i]} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right) + \sum_{j=0}^{k_\Psi - 1} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right) + \Delta_{\Psi_{k_\Psi}}^{[i]} + \sum_{j=1}^{k_\Psi - 1} \Xi_{\Psi_j}^{[i]} \Delta_{\Psi_j} + \varepsilon_\Psi^{(i)},
\end{aligned}$$

with $i \in \{0, 1, \ldots, m\}$. Such errors are not quantifiable, since they depend on the quantities $\varepsilon_\Phi^{[1]}$, $\varepsilon_\Phi^{[2]}$, Δ_{Φ_j}, with $j \in \{0, 1, \ldots, k_\Phi\}$, ε_Ψ, Δ_{Ψ_j}, with $j \in \{0, 1, \ldots, k_\Phi\}$, which are, by nature, unknown. Nevertheless, exploiting the analysis performed earlier in this chapter, it is possible to see how all the above approximation errors are bounded. For sake of simplicity, and without loss of generality, only the boundedness of the errors associated with the drift dynamics components is shown.

As it is highlighted in the proof of Theorem 5.1, the effect of terms $\Lambda^{[p]}_{\Phi_{k_\Phi}} \operatorname{vec}\left(\tilde{V}^{[p]}_{k_\Phi}\right)$ and $\sum_{j=0}^{k_\Phi-1} \Lambda^{[p]}_{\Phi_j} \operatorname{vec}\left(\tilde{V}_j\right)$, with $p \in \{1,2\}$, is compensated by the weight adaptation laws (5.45) and (5.47). Hence, it is sufficient to assess that the other terms of the equation are bounded to confirm that the whole error is limited. In particular, the term $\varepsilon^{[p]}_\Phi$ is bounded by virtue of the universal approximation property [55]. As for $\sum_{j=1}^{k_\Phi-1} \Xi^{[p]}_{\Phi_j} \Delta_{\Phi_j}$, its boundedness can be deduced from the following reasoning. First, the use of the proj operator, described in Appendix A, ensures that the estimated weights $\hat{V}_j$ are always bounded in the set $\mathcal{B}^\alpha_{\Phi_j}$. This means that, since the ideal weights V_j are bounded by definition, then also the weight estimation errors $\tilde{V}_j$ are bounded. Since the activation functions $\hat{\phi}_j$ are of class C^1 and Lipschitz continuous, then, it follows that $\Xi^{[p]}_{\Phi_j}$ is bounded as well. Considering the expression of $\Delta^{[p]}_{\Phi_{k_\Phi}}$ and Δ_{Φ_j} with the aforementioned considerations, then one can conclude that $\tilde{V}_j^\top \hat{\phi}'_j \tilde{\Phi}_{j-1}$ is bounded. Finally, for a wide range of activation functions, including the hyperbolic tangent and the sigmoid function, the term of order two of the Taylor approximation, i.e., $O^2(\tilde{\Phi}_{j-1})$, is bounded, as detailed in [93].

Following the same reasoning for the approximation error associated with the control effectiveness, the following result can be introduced

Proposition 5.3. *At each time instant $t_k \geq t_0$, the norm of the approximation errors introduced by the DNNs $\hat{\Phi}$ and $\hat{\Psi}$ is bounded. In particular, the quantities*

$$\sup_{x\in\mathcal{X}} \left\| f_1 - \hat{\Phi}^{[1]}_{k_\Phi,t_k} \right\|, \quad \sup_{x\in\mathcal{X}} \left\| f_2 - \hat{\Phi}^{[2]}_{k_\Phi,t_k} \right\|, \quad \sup_{x\in\mathcal{X}} \left\| \sum_{i=1}^{m} \bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi,t_k} \right\|,$$

where $\hat{\Phi}^{[p]}_{k_\Phi,t_k}$ and $\hat{\Psi}^{[i]}_{k_\Psi,t_k}$ denote the output of the $\hat{\Phi}$ and $\hat{\Psi}$ when they are parameterized by weight matrices $V_j(t_k)$ and $U_j(t_k)$, are bounded.

From Theorem 5.1 follows that, there exists a time instant in which a practical sliding mode is enforced. Hence, the following holds.

Proposition 5.4. *There exists a time instant $t_1 < \infty$ such that the sliding mode is achieved in a ς-neighborhood of the sliding manifold. i.e.,*

$$\{x(t) \in \mathcal{X} : \|\sigma(x(t))\| \leq \varsigma\}$$

for $t \geq t_1$, with $\varsigma \in \mathbb{R}_{>0}$.

By virtue of Proposition 5.4, it is possible to design a control strategy that, combining DNN-ISM with ASMC, ensures that sliding mode $\sigma(x(t)) = 0_m$ in finite time. Such a strategy can be summarized as follows:

1. for $t \in [t_0, t_1]$, the discontinuous control gain ρ is chosen according to Theorem 5.1 and kept constant, while the DNNs are updated according to (5.45), (5.47), (5.48), and (5.49);

2. for $t \in (t_1, \infty]$, the weights of the DNNs are no longer adapted, i.e., $\text{vec}\left(\dot{V}_j\right) = 0$ for $j \in \{0, 1, \ldots, k_\Phi\}$ and $\text{vec}\left(\dot{U}_j\right) = 0$ for $j \in \{0, 1, \ldots, k_\Psi\}$, while the discontinuous control gain is adapted.

Before introducing the theoretical results of the aforementioned strategy, an observation, related to the existence needs to be made.

Proposition 5.5. *Following the same steps reported in the proof of Theorem 5.1, and relying on the bounds of the system and the ideal weights for bounding the approximation error, it holds that if*

$$\bar{\rho} = \frac{\|C_1\|\left(\bar{V}^{k_\Phi} + \bar{f}_1\right) + \|C_2\|\left(\bar{V}^{k_\Phi} + \bar{f}_2 + \bar{h}\right) + \bar{\eta}}{\underline{\lambda}(C_2)\underline{\gamma}} + \frac{\|C_2\|\left(\bar{U}^{k_\Psi} + \bar{\gamma}\right)\|u_n\|}{\underline{\lambda}(C_2)\underline{\gamma}}, \quad (5.57)$$

with $\bar{\eta} \in \mathbb{R}_{>0}$, then $\sigma(x(t)) = 0_m$ for $t \geq t_0$.

In other words, Proposition 5.5 says that, if one selects a discontinuous control gain that compensates all the disturbances of the system and the worst possible approximation error of the DNNs, then a sliding mode is enforced from the initial time instant, without the need of adapting the networks.

For convenience, defining

$$\bar{\rho}_1 = \frac{\|C_1\|\left(\bar{V}^{k_\Phi} + \bar{f}_1\right) + \|C_2\|\left(\bar{V}^{k_\Phi} + \bar{f}_2 + \bar{h}\right)}{\underline{\lambda}(C_2)\underline{\gamma}}, \quad (5.58)$$

$$\bar{\rho}_2 = \frac{\|C_2\|\left(\bar{U}^{k_\Psi} + \bar{\gamma}\right)}{\underline{\lambda}(C_2)\underline{\gamma}}, \quad (5.59)$$

one can rewrite (5.57) in a more compact way as

$$\bar{\rho} = \bar{\rho}_1 + \bar{\rho}_2\|u_n\| + \frac{\bar{\eta}}{\underline{\lambda}(C_2)\underline{\gamma}}.$$

Theorem 5.2. *Consider the nonlinear system in* (5.3) *and integral sliding variable σ in* (5.39)*. For $t > t_1$, with t_1 being the one defined in Proposition 5.4 , let* (5.3) *be controlled via ISM control law u in* (5.8)*, where the switching law is designed as*

$$u_r(t) = -(\hat{\rho}_1(t) + \hat{\rho}_2(t)\|u_n\|)\frac{\sigma(x(t))}{\|\sigma(x(t))\|},$$

with $\hat{\rho}_1(t_1)$ and $\hat{\rho}_2(t_1)$ chosen so that $\hat{\rho}_1(t_1) + \hat{\rho}_2(t_1)\|u_n\|$ satisfies Theorem 5.1. Moreover, let $\text{vec}\left(\dot{V}_j\right) = 0$ for $j \in \{0, 1, \ldots, k_\Phi\}$ and $\text{vec}\left(\dot{U}_j\right) = 0$ for $j \in$

$\{0, 1, \ldots, k_\Psi\}$. *If Assumptions 5.1 - 5.6 hold and the discontinuous control gain components are adapted according to*

$$\dot{\hat{\rho}}_1(t) = \underset{\varrho_1}{\text{proj}} \left(\left(\|C_2\| \, \bar{\gamma} + \alpha_1 \right) \|\sigma\| \right), \tag{5.60a}$$

$$\dot{\hat{\rho}}_2(t) = \underset{\varrho_2}{\text{proj}} \left(\left(\|C_2\| \, \bar{\gamma} \, \|u_n\| + \alpha_2 \right) \|\sigma\| \right), \tag{5.60b}$$

with $\alpha_1, \alpha_2 \in \mathbb{R}_{>0}$ *being constants acting as learning rates,* $\bar{\gamma}$ *being the one appearing in Assumption 5.2,* $\varrho_1 \coloneqq \{r \in \mathbb{R}_{>0} : r \leq \bar{\rho}_1\}$, *and* $\varrho_2 \coloneqq \{r \in \mathbb{R}_{>0} : r \leq \bar{\rho}_2\}$, *then a sliding mode* $\sigma(x(t)) = 0_m$ *is enforced for* $t \geq t_2$, *with* $t_1 < t_2 < \infty$.

Proof. See Appendix B.2. □

5.4 Practical Aspects

Before presenting the simulations and experiments used for assessing the efficacy of the DNN-ISM control strategy, it is worth to analyze a couple of aspects associated with practical implementation.

5.4.1 Computational Complexity Analysis

In the following, the computational complexity analysis associated with the computation of the adaptation laws of the DNNs weights is carried out. Since the analysis is analogous for both networks, it is carried out only for the DNN $\hat{\Phi}$.

The weights of the j-th are updated according to

$$\begin{aligned} \text{vec}\left(\dot{\hat{V}}_j\right) &= \underset{\mathcal{B}_{\Phi_j}}{\text{proj}} \left(\Gamma_{\Phi_j} \Lambda_{\Phi_j}^\top C^\top \sigma \right) \\ &= \underset{\mathcal{B}_{\Phi_j}}{\text{proj}} \left(\Gamma_{\Phi_j} \left[\Xi_{\Phi_j} (I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top) \right]^\top C^\top \sigma \right). \end{aligned}$$

In the practice, in most iterations, the projection operator behaves like the identity function, i.e., $\text{proj}(\tau) \equiv \tau$. For this reason, it is reasonable to perform the following analysis considering

$$\text{vec}\left(\dot{\hat{V}}_j\right) = \Gamma_{\Phi_j} \left[\Xi_{\Phi_j} (I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top) \right]^\top C^\top \sigma.$$

From that, it is clear that the most recurrent operation in the computation of the weight update is the matrix multiplication.

In general, given two compatible matrices $A \in \mathbb{R}^{h \times l}$, and $B \in \mathbb{R}^{l \times m}$, the computational complexity of a standard matrix multiplication algorithm that finds the

matrix $C = AB$ has computational complexity $\mathcal{O}(hlm)$. This because each element C_{ij} of the resulting matrix $C \in \mathbb{R}^{h\times m}$ is the result of a scalar product between the i-th row of A and the j-th column of B. Since the scalar product of two l-dimensional vectors has computational complexity $\mathcal{O}(l)$, and the number of elements in C is hm, it is clear how the complexity of $C = AB$ is $\mathcal{O}(hlm)$. Then, if A and B are square matrices in $\mathbb{R}^{h\times h}$, the complexity becomes $\mathcal{O}(h^3)$. Over the years some algorithms to lower the complexity have been proposed, but even the more efficient *Strassen's Algorithm* introduced in [102] reached a lower bound of $\mathcal{O}(h^{\log 7}) \approx \mathcal{O}(h^{2.807})$, so in the following analysis the standard algorithm with cubic complexity will be considered.

Since modern libraries (e.g., Eigen, NumPy, TensorFlow) handle the transposition operation very efficiently, such an operation can be neglected in the following analysis, without compromising the validity of this last one.

The first term which is analyzed is Ξ_{Φ_j}, computed as in (5.26a). As it is clear from its definition, it is just a series of matrix multiplication alternating between estimated weight matrices and Jacobian matrices, i.e.,

$$\Xi_{\Phi_j} = \prod_{l=j+1}^{\overset{\curvearrowleft}{k_\Phi}} \hat{V}_l^\top \hat{\phi}'_l = \left(\hat{V}_{k_\Phi}^\top \hat{\phi}'_{k_\Phi}\right)\left(\hat{V}_{k_\Phi-1}^\top \hat{\phi}'_{k_\Phi-1}\right)\cdots\left(\hat{V}_{j+1}^\top \hat{\phi}'_{j+1}\right), \tag{5.61}$$

with each term inside the parenthesis being a matrix multiplication between $\hat{V}_l^\top \in \mathbb{R}^{L_{\Phi_{l+1}}\times L_{\Phi_l}}$ and $\hat{\phi}'_l \in \mathbb{R}^{L_{\Phi_l}\times L_{\Phi_l}}$, meaning that the total complexity for the computation of each term in Ξ_{Φ_j} is $\mathcal{O}\left(\sum_{l=j+1}^{k_\Phi} L_{\Phi_j}^2 L_{\Phi_{j+1}}\right)$.

Then, since each term $\left(\hat{V}_l^\top \hat{\phi}'_l\right) \in \mathbb{R}^{L_{\Phi_{l+1}}\times L_{\Phi_l}}$ must be multiplied together, the complexity for Ξ_{Φ_j} is

$$\mathcal{O}\left(\sum_{l=j+1}^{k_\Phi} L_{\Phi_l}^2 L_{\Phi_{l+1}} + L_{\Phi_{j+1}} \sum_{l=j+2}^{k_\Phi} L_{\Phi_l} L_{\Phi_{l+1}}\right), \tag{5.62}$$

where the first term is associated with the computation of each single product, and the latter one is the one accounting for the final product.

Then, the complexity analysis of the Kronecker product $(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top)$ is carried out. In general, since the Kronecker product of $A \in \mathbb{R}^{h\times l}$ with $B \in \mathbb{R}^{d\times m}$ is defined as in Definition 3.2, its computational cost is $\mathcal{O}(hldm)$. However, since in this case of, $A = I_{L_{\Phi_{j+1}}}$, the elements of $B = \hat{\phi}_j^\top$ just get multiply by 1, and thus one can consider the Kronecker product as a copy of $\hat{\phi}_j^\top$ on a block diagonal matrix with

L_{j+1} blocks, i.e.,

$$I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top = \begin{bmatrix} \hat{\phi}_j^\top & & & & \\ & \hat{\phi}_j^\top & & & \\ & & \ddots & & \\ & & & \hat{\phi}_j^\top & \\ & & & & \hat{\phi}_j^\top \end{bmatrix} \in \mathbb{R}^{L_{\Phi_{j+1}} \times L_{\Phi_j} L_{\Phi_{j+1}}}, \tag{5.63}$$

with a complexity lowered to $\mathcal{O}(L_{\Phi_{j+1}})$.

Then, since $\Xi_{\Phi_j} \in \mathbb{R}^{n \times L_{\Phi_{j+1}}}$, the complexity of the matrix multiplication term $\Lambda_{\Phi_j} = \Xi_{\Phi_j}(I_{L_{\Phi_{j+1}}} \otimes \hat{\phi}_j^\top) \in \mathbb{R}^{n \times L_{\Phi_j} L_{\Phi_{j+1}}}$ is $\mathcal{O}(nL_{\Phi_{j+1}}^2 L_{\Phi_j})$, which, summed up to the complexity necessary to obtain the two components of the product, results in

$$\mathcal{O}\left(\sum_{l=j+1}^{k_\Phi} L_{\Phi_l}^2 L_{\Phi_{l+1}} + L_{\Phi_{j+1}} \sum_{l=j+2}^{k_\Phi} L_{\Phi_l} L_{\Phi_{l+1}} + L_{\Phi_{j+1}} + nL_{\Phi_{j+1}}^2 L_{\Phi_j} \right).$$

Recalling that $C \in \mathbb{R}^{m \times n}$ and $\sigma \in \mathbb{R}^m$, then the multiplication $\Lambda_{\Phi_j}^\top C^\top \sigma \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}}$ has complexity is $\mathcal{O}(nL_{\Phi_j} L_{\Phi_{j+1}} + nm)$, which, added to the complexity of Λ_{Φ_j} leads to

$$\mathcal{O}\Bigg(\sum_{l=j+1}^{k_\Phi} L_{\Phi_l}^2 L_{\Phi_{l+1}} + L_{\Phi_{j+1}} \sum_{l=j+2}^{k_\Phi} L_{\Phi_l} L_{\Phi_{l+1}} + L_{\Phi_{j+1}} + nL_{\Phi_{j+1}}^2 L_{\Phi_j} + nL_{\Phi_j} L_{\Phi_{j+1}} + nm \Bigg).$$

Finally, premultiplying by the gain matrix $\Gamma_{\Phi_j} \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}} \times L_{\Phi_j} L_{\Phi_{j+1}}}$ results in a total complexity for the computation of vec $\left(\dot{V}_j\right)$ which is

$$\mathcal{O}\Bigg(\sum_{l=j+1}^{k_\Phi} L_{\Phi_l}^2 L_{\Phi_{l+1}} + L_{\Phi_{j+1}} \sum_{l=j+2}^{k_\Phi} L_{\Phi_l} L_{\Phi_{l+1}} + L_{\Phi_{j+1}} + nL_{\Phi_{j+1}}^2 L_{\Phi_j} + nL_{\Phi_j} L_{\Phi_{j+1}} + nm + L_{\Phi_j}^2 L_{\Phi_{j+1}}^2 \Bigg). \tag{5.64}$$

Finally, since there are k_Φ layers in the DNN, the overall complexity for computing the update laws of all the layers is given by

$$\mathcal{O}\Bigg(\sum_{j=0}^{k_\Phi} \sum_{l=j+1}^{k_\Phi} L_{\Phi_l}^2 L_{\Phi_{l+1}} + \sum_{j=0}^{k_\Phi} L_{\Phi_{j+1}} \sum_{l=j+2}^{k_\Phi} L_{\Phi_l} L_{\Phi_{l+1}} + \sum_{j=0}^{k_\Phi} L_{\Phi_{j+1}} + n \sum_{j=0}^{k_\Phi} L_{\Phi_{j+1}}^2 L_{\Phi_j} + n \sum_{j=0}^{k_\Phi} L_{\Phi_j} L_{\Phi_{j+1}} + k_\Phi nm + \sum_{j=0}^{k_\Phi} L_{\Phi_j}^2 L_{\Phi_{j+1}}^2 \Bigg). \tag{5.65}$$

Assuming that $L_{\Phi_j} = L$ for $j \in \{0, 1, \dots, k_\Phi\}$, the total complexity can be reformulated as

$$\mathcal{O}\Bigg(\sum_{j=0}^{k_\Phi - 1} (k_\Phi - j)L^3 + \sum_{j=0}^{k_\Phi - 2} (k_\Phi - j - 1)L^3 + k_\Phi L + nk_\Phi L^3 + \\ + nk_\Phi L^2 + k_\Phi nm + k_\Phi L^4 \Bigg).$$

Since in general it holds that $k_\Phi \ll L$, the dominant term in the above complexity is $\mathcal{O}(kL^4)$, which is polynomial of order four with respect to the number of neurons and linear in the number of layers, meaning that, from the computational cost point of view, it is more convenient to exploit the depth of the DNN, than its width.

The above result, accompanied with the ones provided in Section 3.1.3, confirm that, it is more convenient to adopt ANNs with deep architecture ($k \geq 2$), than shallow ones ($k = 1$) when the adaptation of the weights is done according to (5.45), (5.47), (5.48), and (5.49), both from the computational and to the approximation capability point of views.

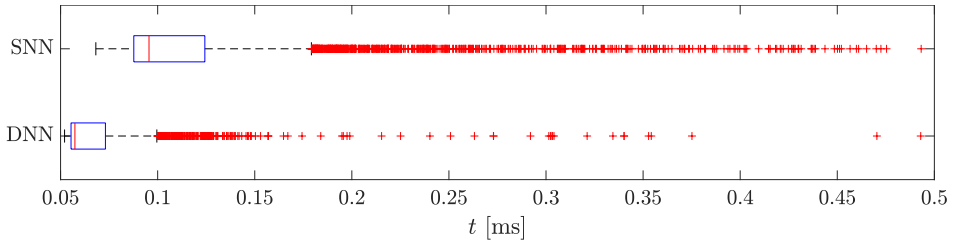
Recalling that, according to the big-Ω notation in (3.17), the number of the hidden layer neurons of a shallow ANN, denoted as L_s, that guarantees the same degree of approximation of a DNN with $k \geq 2$ hidden layers, each with L_d neurons, is given by the following relation

$$\left(\frac{L_d}{n}\right)^{(k-1)n} L_d^n = L_s^n, \tag{5.66}$$

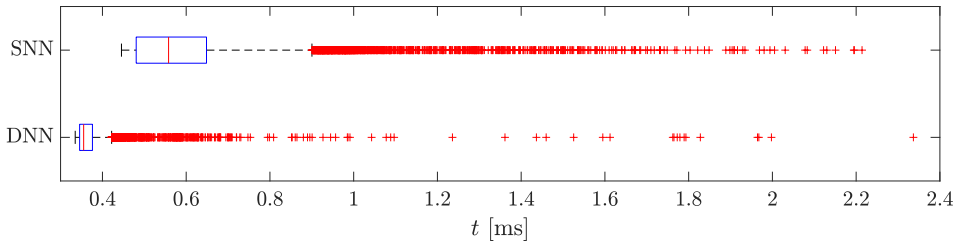
it has been possible to design a test to verify that using larger k_Φ and k_Ψ instead of higher number of neurons in each layer leads to reduced computation time.

In the test, the ANNs $\hat{\Phi}$ and $\hat{\Psi}$ are used to estimate a vector $v \in \mathbb{R}^7$, and a matrix $A \in \mathbb{R}^{7\times 7}$, respectively, starting from an input $x \in \mathbb{R}^{15}$. In the case in which $\hat{\Phi}$ and $\hat{\Psi}$ are DNNs, they are designed so that $k_\Phi = k_\Psi = 8$, with $L_{d,\Phi} = L_{d,\Psi} = 18$. Exploiting (5.66), a shallow version of the $\hat{\Phi}$ and $\hat{\Psi}$, with $L_{s,\Phi} = L_{s,\Psi} = 64$ has been designed.

The test has been performed using C++, recording the execution time using the class `std::chrono::high_resolution_clock`, which returns the approximate CPU time spent in the update procedures [103]. The results of the test are reported in Figure 5.2, which clearly shows how the update procedure is faster in the case of the DNN. In particular, the sample median computed for the DNNs is almost always half the one of the shallow counterparts.



(a) Box plot of the update procedure for $\hat{\Phi}$.



(b) Box plot of the update procedure for $\hat{\Psi}$.

Figure 5.2: Outlier box plot of the recorded times of the full updates of the neural networks. The box indicates the data samples between the 25-th and the 75-th percentile, with the vertical red line indicating the median of the sample distribution. The whiskers of the box plot go from the 5-th to the 25-th and from the 75-th to the 95-th percentile, while the red markers indicate the outliers.

5.4.2 Chattering Reduction

The considered class of systems (5.1) includes the class of electromechanical systems. In real-world implementation, systems that belong to such a class are characterized by actuators that have a finite maximum frequency of operation, or an intrinsic propensity to wearing. Hence, the use of a discontinuous control component, like the one in (5.44), may not be advisable due to the possible presence of chattering. To cope with such a problem, several techniques have been proposed (see, e.g., [41, 40, 44], among others). However, such techniques rely on the concept of HOSM, hence they are not directly applicable to the DNN-ISM control strategy. In the following, two different easy-to-implement chattering reduction schemes, frequently adopted in the practice, are presented.

The first one, introduced in [36, Section 5], exploits the use of the so-called *average-control*, whose aim is to approximate the ideal equivalent control relying on a low pass filter, similarly to what is reported in Section 2.4.2. In particular, the full control laws becomes

$$u(t) = u_n(t) + u_{r,\mathrm{av}}(t), \tag{5.67}$$

where $u_{r,\text{av}}$ is obtained by filtering u_r in (5.44) via a low-pass filter as

$$\mu \dot{u}_{r,\text{av}}(t) = u_r(t) - u_{r,\text{av}}(t), \tag{5.68}$$

where $\mu \in (0, 1)$ is the bandwidth of the filter and $u_{r,\text{av}}(t_0) = 0_m$. To obtain optimal performances, μ should be chosen as small as possible, but large enough not to alter the slow components of u_r.

As for the second technique, it is the *boundary layer control* [27, Section 3.7]. With this approach, the control law is

$$u(t) = u_n(t) + u_{r,\text{bl}}(t), \tag{5.69}$$

where $u_{r,\text{bl}}$ is obtained modifying (5.44) as

$$\mu \dot{u}_{r,\text{bl}}(t) = -\rho \frac{\sigma(x(t))}{\|\sigma(x(t))\| + \varepsilon_\sigma}, \tag{5.70}$$

with $\varepsilon_\sigma \in \mathbb{R}_{>0}$ being an arbitrarily small constant.

5.4.3 Weights initialization

As described earlier in this chapter, the proposed strategy can be divided in two phases. In the first, a constant discontinuous control gain is employed and the weights of the DNNs are adapted until, at time $t_1 \geq t_0$, a practical sliding mode $\|\sigma(x(t))\| \leq \varsigma$ is achieved, while in the second the weights are kept constant and the control gain is adapted until $\sigma(x(t)) = 0_m$.

Intuitively, the enforcement of a practical sliding mode $\|\sigma(x(t))\| \leq \varsigma$ strictly depends on the quality of approximation of $\hat{\Phi}$ and $\hat{\Psi}$. Moreover, the time instant t_1 is strictly related to the initialization of the DNNs. In particular, having good initial estimates allows to enforce a practical sliding mode in a smaller time with respect to the case in which the initial estimates present a large error. Hence, one could exploit previous data about the system to perform an off-line training of $\hat{\Phi}$ and $\hat{\Psi}$ so that a better initialization can be exploited by the DNN-ISM strategy.

Such a philosophy has been implemented in [104], where a simplified case in which only the drift dynamics $f \in \mathbb{R}^6$ of a 3-DoF industrial manipulator was considered unknown. In particular, the weights are initially trained according to the DRL framework using the TD3 learning algorithm, and then the last layer of the actor network is trained according to the DNN-ISM algorithm.

Since the objective was to provide a sufficiently accurate estimate of the drift term, dented as $\hat{f} \in \mathbb{R}^6$, the DRL agent has been trained selecting the state space

$\mathcal{S}$, the action space $\mathcal{A}$, and the instantaneous reward r_t as

$$\mathcal{S} = \{x\}, \quad \mathcal{S} = \{\hat{f}(x)\} \quad r_t = -\Delta\sigma(x(t)),$$

where $x \in \mathbb{R}^6$ is the state vector containing joint positions and velocities, while $\Delta\sigma(x(t)) = \|\sigma(x(t))\| - \|\sigma(x(t^-))\|$, with $\sigma(x(t^-))$ denoting the value of the sliding variable at the previous time instant. Once the DRL training was completed, the weights of the actor network, which corresponds to the DNN estimating the drift dynamics, have been extracted and used as initialization for the DNN-ISM strategy. In [104], this last one has been used to tune only the outer layer of the network, providing benefits from the execution time point of view.

With the same reasoning, a similar technique can be implemented for systems with totally uncertain nominal dynamic model, designing a multi-agent reinforcement learning (MARL) system [105] composed by two agents: one for the drift dynamics and one for the control effectiveness matrix

5.5 Simulations

The efficacy of the DNN-ISM control algorithm has been assessed in simulation on two different systems, whose dynamics is considered fully unknown. The first one is the Duffing oscillator depicted in Figure 2.2 and modeled as in equation (2.3), while the second is the virtualized version of the Franka Emika Panda robot, described in Appendix C.

5.5.1 Duffing Oscillator

The duffing oscillator (2.3) is a mechanical systems. Using Euler-Lagrange modeling, the physical interpretation of the generalized coordinates is position and momentum or velocity, for which the relation is known. For this reason, the DNN $\hat{\Phi}$ is used only for estimating $f_2 \in \mathbb{R}$, having $\hat{\Phi} : \mathcal{X} \to \mathbb{R}$. As for the estimation of the control input, the DNN $\hat{\Psi} : \mathcal{X} \to \mathbb{R}$ is employed. For the simulation, it has been chosen to have $k_\Phi = k_\Psi = 2$ hidden layers for both the DNNs, each characterized by 16 neurons. The adaptation of the weights of the networks is modulated through learning rate matrices $\Gamma_{\Phi_j} = 10\,I$, $\Gamma_{\Psi_j} = 5\,I$, with I denoting an identity matrix with suitable dimensions.

The objective of the simulation, which duration is 40 seconds, is to steer the system state vector towards a time-varying desired equilibrium chosen as $x^\star = \begin{bmatrix} 1.25 & 0 \end{bmatrix}^\top$ for $t \in [0, 20)$ s and $x^\star = \begin{bmatrix} -0.25 & 0 \end{bmatrix}^\top$ for $t \geq 20$ s, while being subject

to a disturbance $h = 0.1\sin(t)$. The stabilizing control law is defined as

$$u_n(t) = -\frac{1}{\hat{\Psi}_{k_\Psi} + \epsilon(t)} \left\{ \hat{\Phi}_{k_\Psi} + \begin{bmatrix} 1 & 2 \end{bmatrix} (x(t) - x^\star(t)) \right\}, \tag{5.71}$$

where $\epsilon \in \mathbb{R}$ is a small design parameter introduced to avoid singularity of the denominator. The integral sliding variable si chosen as $\sigma = \sigma_0 + \hat{z}$, with $\sigma_0 = (x_1 - x_1^\star) + (x_2 - x_2^\star)$. The discontinuous control gain is selected as $\rho = 0.11 + 0.01|u_n|$, while the threshold for the sliding variable that stops the updates of the DNNs and starts the adaptation of the discontinuous gain is $\varsigma = 0.001$. The simulation time-step is set to 10^{-4} seconds.

The results of the simulation are presented in Figure 5.3. From that, it is possible to see how, even applying a relatively small discontinuous control gain, the sliding variable is successfully steered to a very small value (and eventually to zero), after a first transient in which the weights of the DNNs are adapted and during which the transient variable $\hat{z}$ is not properly compensating σ_0. Moreover, the state of the system is successfully controlled to the desired time-varying equilibrium point.

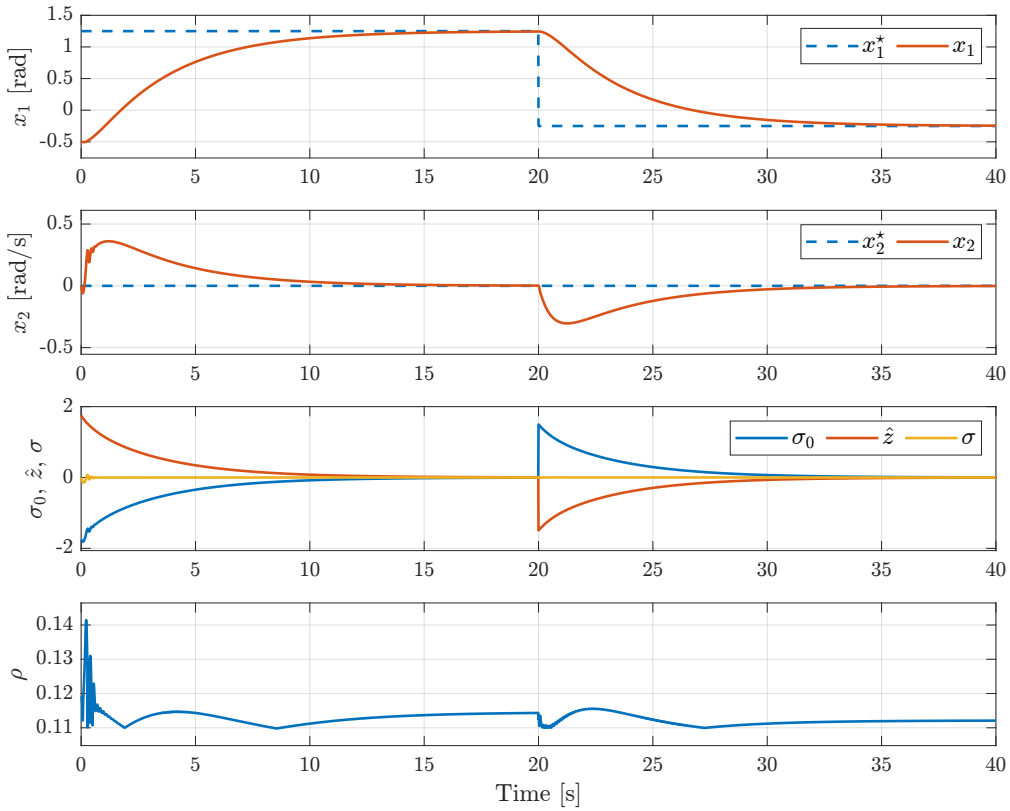


Figure 5.3: Time evolution of the system states, sliding variable components, and discontinuous control gain during the simulation.

Finally, the proposal has been compared with standard SMC. In particular, the

system is controlled to reach the first equilibrium state, defining the sliding variable $\sigma = (x_1 - x_1^\star) + (x_2 - x_2^\star)$ and employing the control law $u(t) = -\rho \operatorname{sign}(\sigma)$, with different constant discontinuous control gains, i.e., $\rho = 0.15$, $\rho = 0.25$, $\rho = 0.5$, and $\rho = 0.75$. The results of the comparison are presented in Figure 5.4, from which it is possible to conclude that, even in the case in which the standard SMC employs a discontinuous control gain more or less 5 time higher than the largest value of the one employed by the DNN-ISM (in Figure 5.3), it is not able to ensure robustness against the disturbance. Since neither DNN-ISM or SMC require the knowledge of the system dynamics, the former perform better than the latter, especially in the cases in which the use of a high discontinuous gain is discouraged, like electromechanical systems.

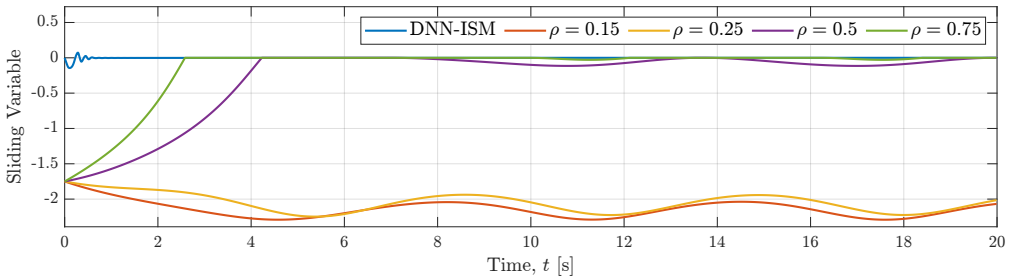


Figure 5.4: Time behavior of the sliding variable when the DNN-ISM is employed with the discontinuous control gain depicted in Figure 5.3, and when SMC is applied with constant gains $\rho = 0.15$, $\rho = 0.25$, $\rho = 0.5$, and $\rho = 0.75$.

5.5.2 Robotic Manipulator

In this simulation, a virtualized model of the Franka Emika Panda is controlled via the DNN-ISM in Figure 5.1, relying on the PyBullet simulator. In particular, the model of the manipulator is the one in described in (C.1), which, defining the state vector $x = \begin{bmatrix} q^\top & \dot{q}^\top \end{bmatrix}^\top \in \mathcal{X} \subset \mathbb{R}^{14}$, can be written in the state-space canonical form

$$\dot{x}(t) = \begin{bmatrix} f_1(x(t), t) \\ f_2(x(t), t) + \bar{B}(x(t), t)u \end{bmatrix} = \begin{bmatrix} \dot{q} \\ -M(q)^{-1}C(q, \dot{q})\dot{q} + M^{-1}(q)\tau \end{bmatrix}, \tag{5.72}$$

having $f_1 = \dot{q} \in \mathbb{R}^7$, $f_2 = M^{-1}(q)C(q, \dot{q})\dot{q} \in \mathbb{R}^7$, $\bar{B} = M^{-1}(q) \in \mathbb{R}^{7\times 7}$, and $u = \tau \in \mathbb{R}^7$. Since f_1 can be obtained directly from robot sensors measurements, only f_2 and $\bar{B}$ are estimated. In particular, the former is estimated by a DNN $\hat{\Phi} : \mathcal{X} \to \mathbb{R}^7$ while the latter by $\hat{\Psi} : \mathcal{X} \to \mathbb{R}^{49}$. The two networks have been designed with $k_\Phi = k_\Psi = 3$ hidden layers, each characterized by 16 neurons. The weights are adapted with adaptation rate matrices $\Gamma_{\Phi_j} = 50I$ and $\Gamma_{\Psi_j} = 250I$.

The objective of the simulation, whose duration is 15 seconds, is to track the desired joint space trajectory

$$q^\star(t) = \begin{bmatrix} \frac{\pi}{5} + \sin(t) & -\frac{\pi}{5} + 0.1\cos(t) & 0.1\sin(2t) & -\frac{7}{10}\pi & 0 & \frac{\pi}{4} + \cos(2t) & -\frac{\pi}{4} \end{bmatrix}^\top$$
$$\dot{q}^\star(t) = \begin{bmatrix} \cos(t) & -0.1\sin(t) & 0.2\cos(2t) & 0 & 0 & -2\sin(2t) & 0 \end{bmatrix}^\top$$
$$\ddot{q}^\star(t) = \begin{bmatrix} -\sin(t) & -0.1\cos(t) & -0.4\sin(2t) & 0 & 0 & -4\cos(2t) & 0 \end{bmatrix}^\top,$$

while being subject to a matched disturbance

$$h(t) = \begin{bmatrix} 0.1\sin(5t) & 0.05 + 0.05\cos(8t) & 0.2 & 0 & 0.075\sin(2t) & 0 & 0.01\cos(10t) \end{bmatrix}^\top.$$

The sliding variable is defined as $\sigma = \sigma_0 + \hat{z}$. The stabilizing control law has been chosen as the feedback linearizing law

$$u_n(t) = -\left(\text{vec}^{-1}\left(\hat{\Psi}_{k_\Psi}\right)\right)^+ \left\{\hat{\Phi}_{k_\Phi} + K_p\Big(q(t) - q^\star(t)\Big) + K_d\Big(\dot{q}(t) - \dot{q}^\star(t)\Big) - \ddot{q}^\star(t)\right\}, \quad (5.73)$$

where the gain matrices are chosen as $K_p = K_d = 5I_7$, while $(\cdot)^+$ denotes the pseudoinverse operation in Definition 4.2. The matrices $C_1 \in \mathbb{R}^7$ and $C_2 \in \mathbb{R}^7$ appearing in (5.5) are chosen as $C_1 = C_2 = I_7$. The discontinuous control gain has been selected as $\rho = 0.5 + 1.5\,\|u_n\|$, while the threshold for the sliding variable that stops the updates of the DNNs and starts the adaptation of the discontinuous gain is $\varsigma = 0.001$. The simulation time-step is 10^{-4} seconds. The results of the simulation are presented in Figure 5.5, from which it is possible to see how, except for a first time transient in which the DNNs are updating, the sliding variable is steered toward zero and the desired trajectory is successfully tracked. Moreover, the control input vector and the discontinuous control gain are depicted in Figure 5.6 and Figure 5.7, respectively. Note that, in Figure 5.6, the values on the vertical axis of each figure has been limited for sake of visibility of the rest of the signal. In fact, due to some poor initial condition of the weights, there are some spikes in the control signal, with a value around 40 Nm. Finally, from Figure 5.7 it is possible to see that, except from an initial time transient in which the discontinuous gain reaches relatively high values, once the DNNs are updated the value of ρ settles around 5.

5.6 Real Robot Experiment

Finally, the DNN-ISM control architecture depicted in Figure 5.1 has been assessed experimentally on a real Franka Emika Panda, whose characteristics are detailed in Appendix C. Since gravity and frictions are automatically compensated from an internal controller (see Appendix C.4), then the model is the one used for the simulations, and the same considerations about the state-space modeling that have been made in the previous section are valid.

The DNN $\hat{\Phi}$ is characterized by $k_\Phi = 2$ hidden layers, each with 16 neurons, and it is adapted with learning rate matrices defined as $\Gamma_{\Phi_j} = 50I$. As for the $\hat{\Psi}$, it has bin designed

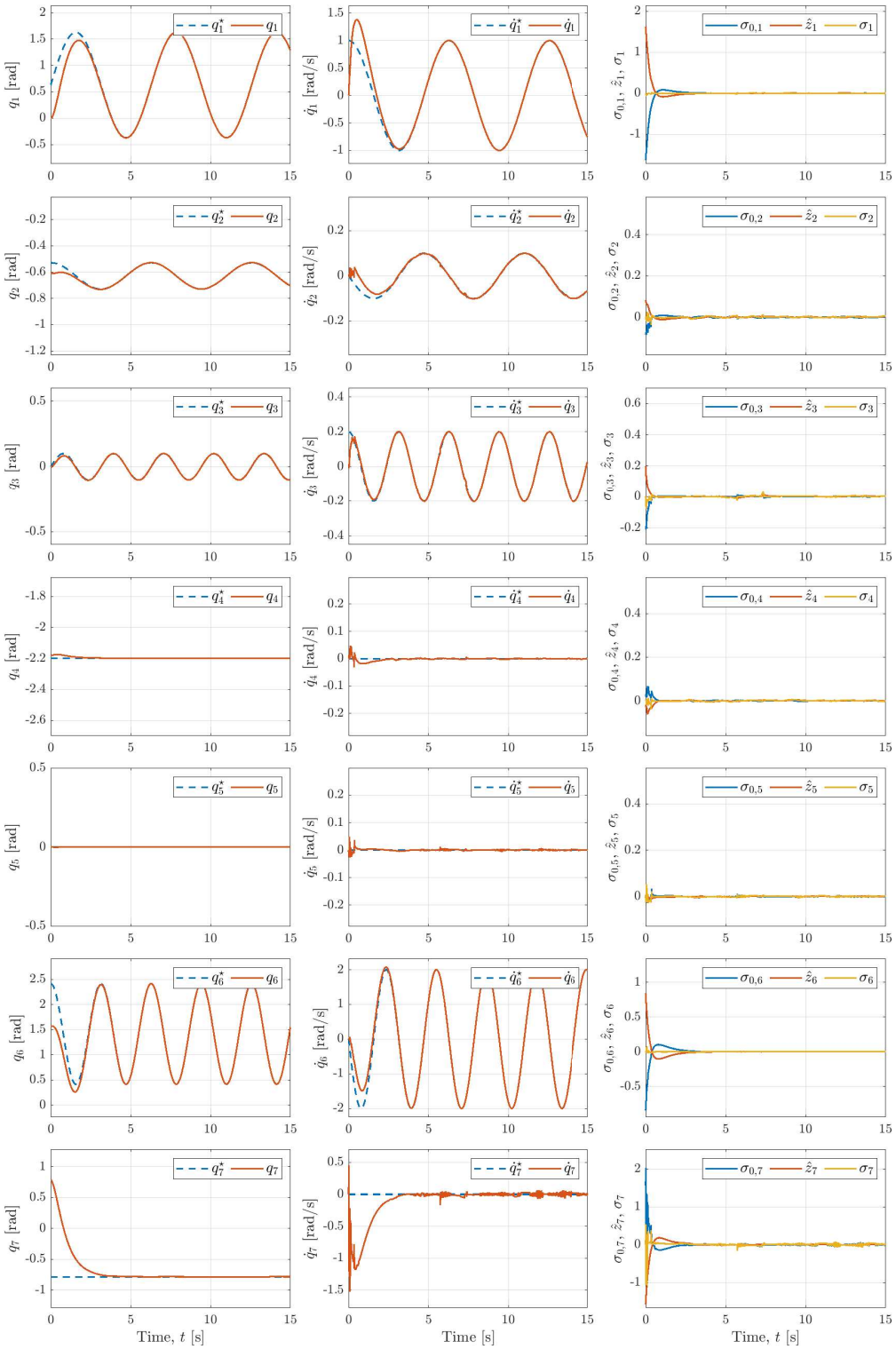


Figure 5.5: Evolution of the joint positions (left column), joint velocities (middle column), and sliding mode components (right column) during the simulation.

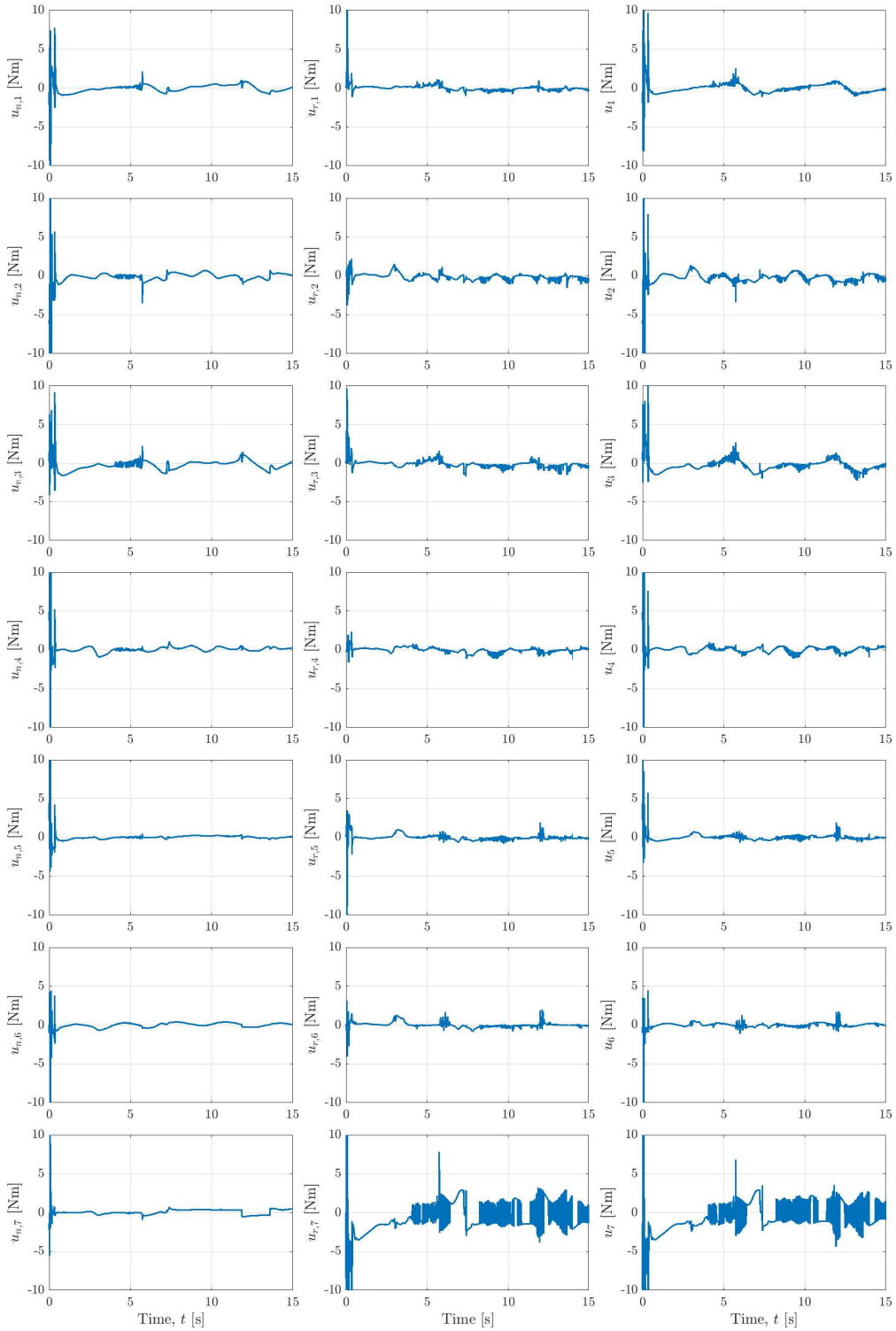


Figure 5.6: Evolution of the nominal control u_n (left column), robustifying control law u_r (middle column), and full control law u (right column) during the simulation.

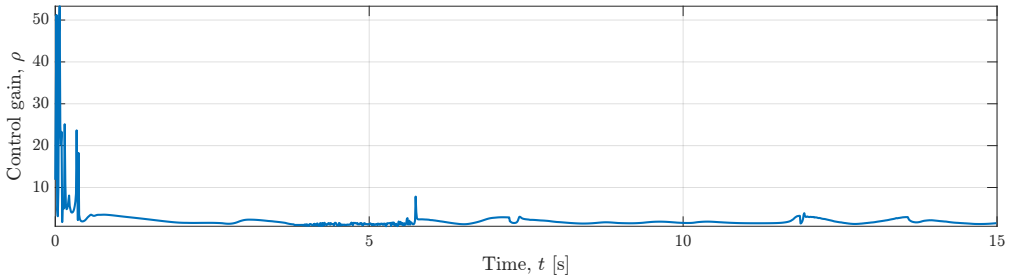


Figure 5.7: Evolution of the discontinuous control gain ρ during the simulation.

with $k_\Psi = 5$ hidden layers with 16 neurons, and it updated with learning rate matrices $\Gamma_{\Psi_j} = 250I$.

The objective of the experiment, whose duration is 60 seconds, is to control the robot so that it tracks the reference trajectory obtained deriving

$$q^\star(t) = \begin{bmatrix} \frac{\pi}{4} & -\frac{\pi}{4} & 0 & -\frac{3}{4}\pi + 0.3\sin(0.2\pi t) & 0.2\sin(\frac{2}{3}\pi t) & \frac{\pi}{2} + 0.2\sin(0.4\pi t) & 0.2\sin(\pi t) \end{bmatrix}^\top,$$

while being subject to a disturbance

$$h(t) = \begin{bmatrix} 0.1\sin(5t) & 0.05 + 0.05\cos(8t) & 0.2 & 0 & 0.075\sin(2t) & 0 & 0.01\cos(10t) \end{bmatrix}^\top.$$

The nominal control law is the DNN based feedback linearizing law in (5.73), with $K_p = K_d = I_7$. The matrices $C_1 \in \mathbb{R}^7$ and $C_2 \in \mathbb{R}^7$ appearing in (5.5) are chosen as $C_1 = C_2 = I_7$. The discontinuous control gain has been selected as $\rho = 1.5 + 0.75\,\|u_n\|$, while the threshold for the sliding variable that stops the updates of the DNNs and starts the adaptation of the discontinuous gain is $\varsigma = 0.001$. The control signal is computed with a frequency more or less equal to 800 Hz. The results of the experiment are presented in Figure 5.8, from which it is possible to see that, after a first transient in which the DNN are adapting, a practical sliding mode is enforced. As one can notice, the sliding variable components are ultimately bounded by values that are higher than the one in the simulation case comes from the fact that the control frequency is more than ten times smaller the one used in the previous section. Nevertheless, the tracking results can be considered more than satisfactorily. Moreover, the control vector components are depicted in Figure 5.9. In order to do not cause damages to the robot motor, the discontinuous component is filtered by a first order filter like the one in (2.75), characterized by filtering constant $\mu = 0.01$.

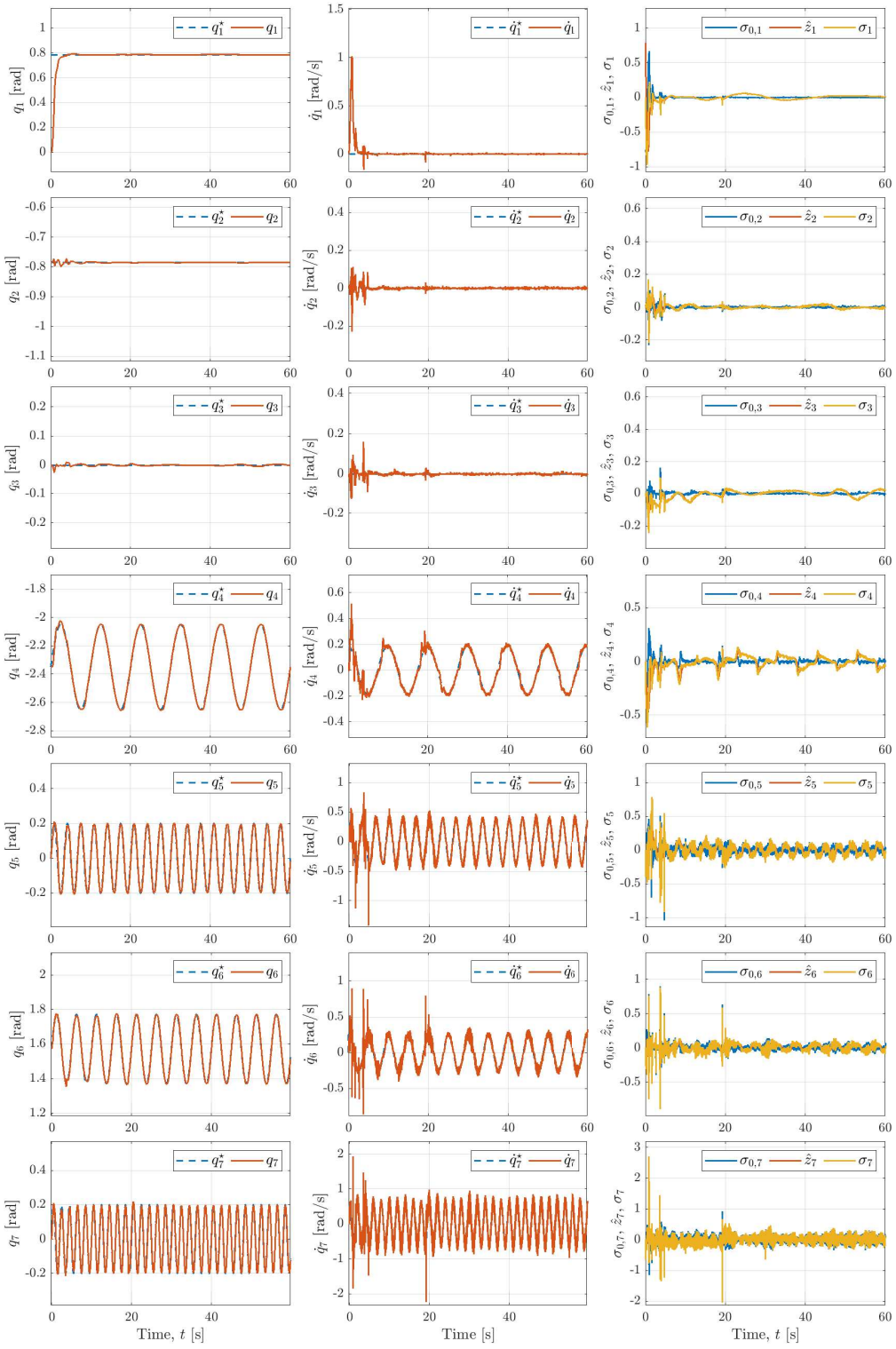


Figure 5.8: Evolution of the joint positions (left column), joint velocities (middle column), and sliding mode components (right column) during the experiment.

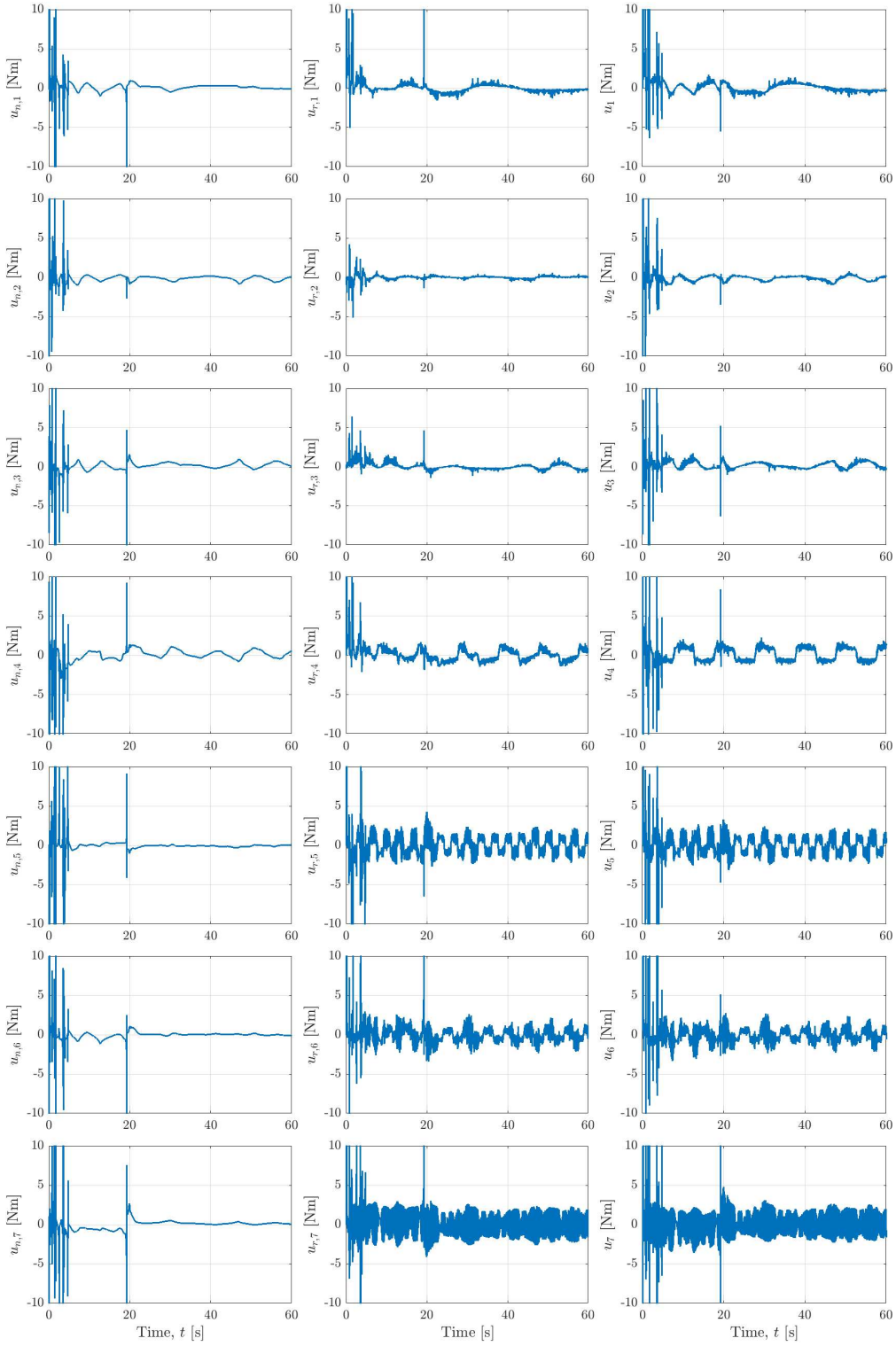


Figure 5.9: Evolution of the nominal control u_n (left column), robustifying control law u_r (middle column), and full control law u (right column) during the experiment.

Chapter 6

DNN-ISM with State and Input Constraints

In this chapter, three different variants of the DNN-ISM framework that allow to take into considerations state and input constraints are presented for some class of systems. In particular, the first scheme modifies the design of the integral sliding manifold and the nominal control law to perform the avoidance of a non admissible portion of the state space, the second one integrates Model Predictive Control (MPC) into the DNN-ISM framework to guarantee the satisfaction of soft state and input constraints, while the last one modifies the original DNN-ISM control integrating Barrier Lyapunov Functions in the DNNs adaptation and generates the nominal control law solving a quadratic programming problem designed using a Control Lyapunov Function (CLF) and a Control Barrier Function (CBF). All the approaches are theoretical analyzed and assessed in simulation on different systems.

6.1 DNN-ISM with State Constraints Avoidance

In this section, the DNN-ISM with a modified integral sliding variable and nominal control law for the avoidance of a non admissible part of the state space is presented. Such a variant has been introduced first in [106] in the case of systems with partially unknown dynamics, and extended to the case of fully unknown model in [101].

6.1.1 Problem Formulation

The considered class of system is the same as in Chapter 5.3. For the reader's convenience, it is briefly recalled in the following.

Consider the nonlinear affected by matched disturbance

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} f_1(x(t), t) \\ f_2(x(t), t) + \bar{B}(x(t), t)u(t) + h(x(t), t) \end{bmatrix}, \tag{6.1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ and $u \in \mathbb{R}^m$, $f_1 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n-m}$ and $f_2 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ are the components of the drift dynamics, $\bar{B} : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{m \times m}$ is the control effectiveness matrix, while $h : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ is the matched perturbation vector. The terms f_1, f_2 and $\bar{B}$ satisfy Assumptions 5.1 and 5.2, while the disturbance h is bounded according to Assumption 5.3.

If one defines a desired trajectory $x^\star(t) \in \mathcal{X}$, it is possible to define the tracking error $e(t) = x(t) - x^\star(t)$, characterized by dynamics

$$\dot{e}(t) = \begin{bmatrix} \dot{x}_1(t) - \dot{x}_1^\star(t) \\ \dot{x}_2(t) - \dot{x}_2^\star(t) \end{bmatrix} = \begin{bmatrix} f_1(x(t), t) - \dot{x}_1^\star(t) \\ f_2(x(t), t) + \bar{B}(x(t), t)u(t) + h(x(t), t) - \dot{x}_2^\star(t) \end{bmatrix}. \tag{6.2}$$

Then, let $\mathcal{X}_c(t) \subset \mathcal{X}$ be a time-dependent set that contains some states that, for some reasons, the nonlinear system (6.1) cannot assume. In the case of mechanical systems, such reasons can be related to the mechanical structure of the system ot the presence of external obstacles. It is convenient to define the nearest non-admissible state $x_c \in \mathcal{X}_c$ and its relative distance $d_c \in \mathbb{R}_{\geq 0}$ from the actual state as

$$x_c(t) = \operatorname*{argmin}_{s \in \mathcal{X}_c} d(x(t), s) \tag{6.3a}$$

$$d_c(t) = \min_{s \in \mathcal{X}_c} d(x(t), s), \tag{6.3b}$$

with $d : \mathcal{X} \times \mathcal{X}_c \to \mathbb{R}_{\geq 0}$ being a distance function chosen, for example, as the squared Euclidean norm, i.e., $d(x(t), s) = \|x(t) - s\|^2$.

The objective is to design an ISM controller which stabilizes the system in (6.1) on the desired trajectory $x^\star$, while rejecting the disturbance h and ensuring $x(t) \notin \mathcal{X}_c(t)$, for all $t \geq t_0$.

Recalling that the integral sliding variable is in general defined as

$$\sigma(x(t)) = \sigma_0(x(t)) - z(x(t)), \tag{6.4}$$

to accomplish the above objective is possible to define the conventional sliding variable $\sigma_0(x(t))$ depending on the the distance $d_c(t)$, i.e.,

$$\sigma_0(x(t)) = \begin{cases} C_{1,a}\Big(x_1(t) - x_{1,s}(t)\Big) + C_{2,a}\Big(x_2(t) - x_{2,s}(t)\Big), & \text{if } d_c(t) \leq d^\star, \\ C_{1,r}\Big(x_1(t) - x_1^\star(t)\Big) + C_{2,r}\Big(x_2(t) - x_2^\star(t)\Big), & \text{if } d_c(t) > d^\star, \end{cases} \tag{6.5}$$

where $C_{1,a} \in \mathbb{R}^{m \times (n-m)}$, $C_{2,a} \in \mathbb{R}^{m \times m}$, $C_{1,r} \in \mathbb{R}^{m \times (n-m)}$, and $C_{2,r} \in \mathbb{R}^{m \times m}$ are design parameters, $d^\star \in \mathbb{R}_{>0}$ is a safety threshold chosen during the design phase , while $x_s(t) = \begin{bmatrix} x_{1,s}^\top(t) & x_{2,s}^\top(t) \end{bmatrix}^\top \in \mathcal{X} \backslash \mathcal{X}_c$ is a trajectory with bounded derivative chosen by the designer to avoid the set of non admissible states. The following assumption about the design matrices $C_{2,a}$ and $C_{2,r}$ is introduced.

Assumption 6.1. *The design matrices $C_{2,a}$ and $C_{2,r}$ are chosen so that $C_{2,a}\bar{B}(x(t), t)$ and $C_{2,r}\bar{B}(x(t), t)$ are symmetric and positive definite, for all $x \in \mathcal{X}$ and $t \geq t_0$.*

Recalling that $\bar{B}(x(t),t)$ satisfies Assumption 5.2, i.e., it is SPD, one can design $C_{2,a} = k_a I_m$ and $C_{2,r} = k_r I_m$, with $k_a, k_r \in \mathbb{R}_{>0}$, for having Assumption 6.1 to hold.

Note that, the expression (6.5) can be written in a more compact manner as

$$\sigma_0(x(t)) = \delta_a(t) \begin{bmatrix} C_{1,a} & C_{2,a} \end{bmatrix} \begin{bmatrix} x_1(t) - x_{1,s}(t) \\ x_2(t) - x_{2,s}(t) \end{bmatrix} + (1 - \delta_a(t)) \begin{bmatrix} C_{1,r} & C_{2,r} \end{bmatrix} \begin{bmatrix} x_1(t) - x_1^\star(t) \\ x_2(t) - x_2^\star(t) \end{bmatrix},$$

where $\delta_a : \mathbb{R}_{\geq 0} \to \{0, 1\}$ is a *constraint avoidance flag* defined as

$$\delta_a(t) = \begin{cases} 1 & \text{if } d_c(t) \leq d^*, \\ 0 & \text{if } d_c(t) > d^*. \end{cases} \tag{6.6}$$

As for the transient function z, it is defined so that

$$\begin{aligned} \dot{z}(x(t)) = \delta_a(t) \begin{bmatrix} C_{1,a} & C_{2,a} \end{bmatrix} \begin{bmatrix} f_1(x(t),t) - \dot{x}_{1,s}(t) \\ f_2(x(t),t) + \bar{B}(x(t),t) u_{n,a}(t) - \dot{x}_{2,s}(t) \end{bmatrix} + \\ + (1 - \delta_a(t)) \begin{bmatrix} C_{1,r} & C_{2,r} \end{bmatrix} \begin{bmatrix} f_1(x(t),t) - \dot{x}_1^\star(t) \\ f_2(x(t),t) + \bar{B} u_{n,r}(t) - \dot{x}_2^\star(t) \end{bmatrix}, \end{aligned} \tag{6.7}$$

with $z(x(t_0)) = \sigma_0(x(t_0))$ and $z(x(t_k)) = \sigma_0(x(t_k))$, where $t_k > t_0$ are the time instants in which $\delta_a(t_k)$ changes its value. The control laws $u_{n,r} \in \mathbb{R}^m$ and $u_{n,a} \in \mathbb{R}^m$ are the continuous control laws which stabilize the system when it is in tracking ($\delta_a = 0$) and avoidance ($\delta_a = 1$) conditions, respectively. In particular, they are designed depending on the task and on the choice of the trajectory $x_s(t)$.

Finally, the complete control law is designed on the ISM control framework, i.e.,

$$u(t) = u_n(t) + u_r(t), \tag{6.8}$$

where the nominal law $u_n \in \mathbb{R}^m$ and the switching law $u_r \in \mathbb{R}^m$ are defined as

$$u_n(t) = (1 - \delta_a(t)) u_{n,r}(t) + \delta_a(t) u_{n,a}(t), \tag{6.9a}$$

$$u_r(t) = -\rho \frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \tag{6.9b}$$

with $\rho \in \mathbb{R}_{>0}$ being the discontinuous control gain and σ chosen as in (6.4).

6.1.2 The DNN-ISM scheme with avoidance capabilities

In the following, a modified version of the DNN-ISM control architecture, depicted in Figure 6.1, allows avoidance of non-admissible states.

Recalling that f_1, f_2, and $\bar{B}$ are not available (see Assumptions 5.1 and 5.2), it is not possible to directly compute the transient variable dynamics in (6.7). Analogously to what done for the DNN-ISM control scheme, it is possible to approximate such terms relying on DNNs $\hat{\Phi}$ and $\hat{\Psi}$. This allows to compute an estimate of z, denoted as $\hat{z}$, and define the integral sliding variable

$$\sigma(x(t)) = \sigma_0(x(t)) - \hat{z}(x(t)), \tag{6.10}$$

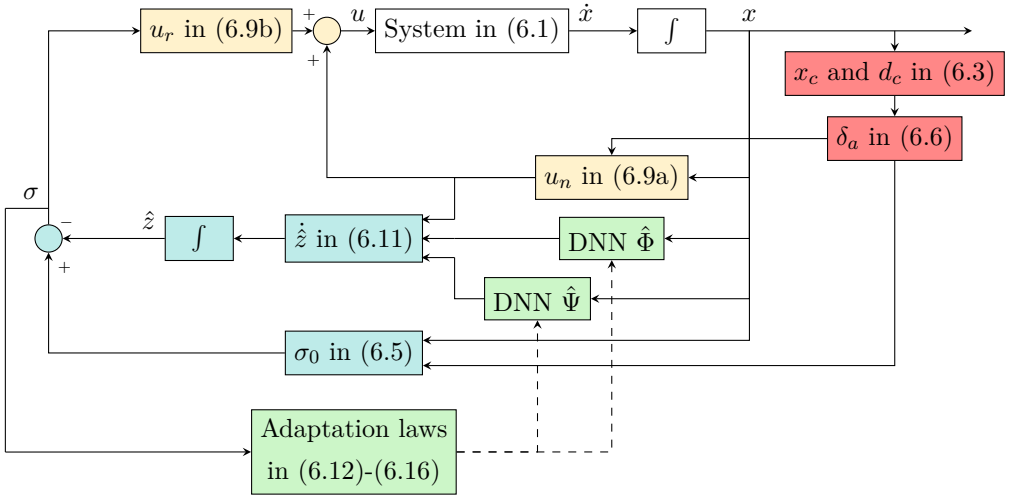


Figure 6.1: Block diagram of the modified DNN-ISM control scheme for the avoidance of non-admissible states. The blocks related to the DNNs are colored in green, the blue ones are associated with the sliding variable, the yellow blocks are the components of the control law, while in red are denoted the blocks responsible for the change of control mode.

where σ_0 is defined as in (6.5) and, as a consequence, $\dot{\hat{z}}$ is computed as

$$\begin{aligned}\dot{\hat{z}}(x(t)) = \delta_a(t) \begin{bmatrix} C_{1,a} & C_{2,a} \end{bmatrix} \begin{bmatrix} \hat{\Phi}^{[1]}_{k_\Phi} - \dot{x}_{1,s}(t) \\ \hat{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_{n,a,i}(t) - \dot{x}_{2,s}(t) \end{bmatrix} + \\ + (1 - \delta_a(t)) \begin{bmatrix} C_{1,r} & C_{2,r} \end{bmatrix} \begin{bmatrix} \hat{\Phi}^{[1]}_{k_\Phi} - \dot{x}^\star_1(t) \\ \hat{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_{n,r,i}(t) - \dot{x}^\star_2(t) \end{bmatrix},\end{aligned} \tag{6.11}$$

where $u_{n,a,i} \in \mathbb{R}$ and $u_{n,r,i} \in \mathbb{R}$ denote, respectively, the i-th components of $u_{n,a}$ and $u_{n,r}$. Similarly to what happens for z, it holds that $\hat{z}(x(t_0)) = \sigma_0(x(t_0))$ and $\hat{z}(x(t_k)) = \sigma_0(x(t_k))$.

Since the integral sliding variable is defined differently from the standard DNN-ISM control scheme, the weight adaptation laws change.

For what concerns the DNN $\hat{\Phi}$, the layers up to the second to last one, i.e., for $j \in \{0, 1, \dots, k_\Phi - 1\}$, the adaptation is done through

$$\operatorname{vec}\left(\dot{\hat{V}}_j\right) = \underset{\mathcal{B}_{\Phi_j}}{\operatorname{proj}}\left(\Gamma_{\Phi_j} \Lambda^\top_{\Phi_j} C^\top_\delta \sigma\right) \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}}, \tag{6.12}$$

where $\Lambda_{\Phi_j} \in \mathbb{R}^{n \times L_{\Phi_j} L_{\Phi_{j+1}}}$ is defined as in in (5.46), while $C_\delta \in \mathbb{R}^{m \times n}$ is given by

$$\begin{aligned} C_\delta &= \begin{bmatrix} C_{1,\delta} & C_{2,\delta} \end{bmatrix} \\ &= \begin{bmatrix} \delta_a C_{1,a} + (1 - \delta_a) C_{1,r} & \delta_a C_{2,a} + (1 - \delta_a) C_{2,r} \end{bmatrix}. \end{aligned} \tag{6.13}$$

Analogously to standard DNN-ISM, the matrix $\Gamma_{\Phi_j} \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}} \times L_{\Phi_j} L_{\Phi_{j+1}}}$ represents the adaptation rate, and it is defined as diagonal with positive entries, while $\operatorname{proj}(\cdot)$ is the

projection operator defined as in Appendix A, with set $\mathcal{B}_{\Phi_j}$ defined in (5.50a). The weights sub-matrices associated with the last layer ($j = k_\Phi$) are characterized by dynamics

$$\operatorname{vec}\left(\dot{\hat{V}}_{k_\Phi}^{[1]}\right) = \underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\operatorname{proj}}\left(\Gamma_{\Phi_{k_\Phi}}^{[1]}(\Lambda_{\Phi_{k_\Phi}}^{[1]})^\top C_{1,\delta}^\top \sigma\right) \in \mathbb{R}^{L_{k_\Phi}(n-m)}, \tag{6.14a}$$

$$\operatorname{vec}\left(\dot{\hat{V}}_{k_\Phi}^{[2]}\right) = \underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\operatorname{proj}}\left(\Gamma_{\Phi_{k_\Phi}}^{[2]}(\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_{2,\delta}^\top \sigma\right) \in \mathbb{R}^{L_{k_\Phi} m}, \tag{6.14b}$$

where $C_{1,\delta} \in \mathbb{R}^{m\times(n-m)}$ and $C_{2,\delta} \in \mathbb{R}^{m\times m}$ are the matrices appearing in (6.13), $\Gamma_{\Phi_{k_\Phi}}^{[1]} \in \mathbb{R}^{(n-m)L_{k_\Phi}\times(n-m)L_{k_\Phi}}$ and $\Gamma_{\Phi_{k_\Phi}}^{[2]} \in \mathbb{R}^{mL_{k_\Phi}\times mL_{k_\Phi}}$ are diagonal matrices with positive entries, while $\Lambda_{\Phi_{k_\Phi}}^{[1]} \in \mathbb{R}^{(n-m)\times(n-m)L_{k_\Phi}}$ and $\Lambda_{\Phi_{k_\Phi}}^{[2]} \in \mathbb{R}^{m\times mL_{k_\Phi}}$ are the ones defined below equation (5.28).

As for $\hat{\Psi}$, for layers $j \in \{0, 1, \dots, k_\Psi - 1\}$, one has that

$$\operatorname{vec}\left(\dot{\hat{U}}_j\right) = \underset{\mathcal{B}_{\Psi_j}}{\operatorname{proj}}\left(\Gamma_{\Psi_j}\left(\sum_{i=1}^{m} u_{n,i}(\Lambda_{\Psi_j}^{[i]})^\top\right) C_{2,\delta}^\top \sigma\right) \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}}, \tag{6.15}$$

where $\Lambda_{\Psi_j}^{[i]} \in \mathbb{R}^{m\times L_{\Psi_j} L_{\Psi_{j+1}}}$ is defined as in (5.37b), $u_{n,i} \in \mathbb{R}$ is the i-th component of the nominal control law in (6.9a) while $\Gamma_{\Psi_j} \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}}\times L_{\Psi_j} L_{\Psi_{j+1}}}$ is diagonal with positive entries. The weight sub-matrices of the last layer are adjusted according to

$$\operatorname{vec}\left(\dot{\hat{U}}_{k_\Psi}^{[i]}\right) = \underset{\mathcal{B}_{\Psi_{k_\Psi}}}{\operatorname{proj}}\left(\Gamma_{\Psi_{k_\Psi}} u_{n,i}(\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top (C_{2,\delta})^\top \sigma\right) \in \mathbb{R}^{L_{k_\Psi} m}, \tag{6.16}$$

with $i \in \{1, 2, \dots, m\}$, where $\Gamma_{\Psi_{k_\Psi}} \in \mathbb{R}^{mL_{k_\Psi}\times mL_{k_\Psi}}$ is diagonal with positive entries, while $\Lambda_{\Psi_{k_\Psi}}^{[i]} \in \mathbb{R}^{m\times mL_{k_\Psi}}$ is the one defined after (5.37).

6.1.3 Sliding mode existence

In the following, the main theoretical results about the modified version of DNN-ISM are presented.

Theorem 6.1. *Consider the nonlinear system in* (5.3)*, controlled via ISM control law* u *in* (6.8)*, with switching law* u_r *in* (6.9b) *with integral sliding variable* σ *defined as in* (6.10)*. Moreover, the DNNs weights are adapted according to the adaptation laws* (6.12) - (6.16)*. If Assumptions 5.1 - 5.6 and Proposition 5.2 hold, and the discontinuous control gain is chose as*

$$\rho > \frac{\|C_{1,\delta}\|\,\bar{r}_{\Phi_1} + \|C_{2,\delta}\|\,(\bar{r}_{\Phi_2} + \bar{h}) + \delta_a\,\|C_{2,a}\|\,\bar{r}_\Psi\,\|u_{n,a}\| + (1-\delta_a)\,\|C_{2,r}\|\,\bar{r}_\Psi\,\|u_{n,r}\| + \bar{\eta}}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}},$$

where $\bar{r}_{\Phi_1} \coloneqq \bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1}$, $\bar{r}_{\Phi_2} \coloneqq \bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2}$, $\bar{r}_\Psi \coloneqq m(\bar{c}_\Psi + \bar{\varepsilon}_\Phi)$*, and* $\bar{\eta} \in \mathbb{R}_{>0}$*, then,* $\sigma(x(t)) \to 0_m$ *for* $t \to \infty$*.*

Proof. See Appendix B.3. □

Similarly to the case of the standard DNN-ISM, the above theorem implies two result. The former one is the achievement of a sliding mode in an ς-vicinity of the sliding manifold

$\sigma(x(t)) = 0_m$ in a finite time, as in Proposition 5.4. The latter is the existence of a discontinuous gain $\bar{\rho}$ that ensures an ideal sliding mode even if the weights of the DNNs have not been tuned.

Proposition 6.1. *If one follows the same steps reported in the proof of Theorem 6.1, and relies on the bounds of the system and the ideal weights for bounding the approximation error, it holds that if*

$$\bar{\rho} = \bar{\rho}_1 + \bar{\rho}_2\Big(\delta_a \left\|u_{n,a}\right\| + (1-\delta_a)\left\|u_{n,r}\right\|\Big) + \frac{\bar{\eta}}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}}, \tag{6.17}$$

where

$$\bar{\rho}_1 = \frac{\|C_{1,\delta}\|\left(\bar{V}^{k_\Phi} + \bar{f}_1\right) + \|C_{2,\delta}\|\left(\bar{V}^{k_\Phi} + \bar{f}_2 + \bar{h}\right)}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}}, \tag{6.18a}$$

$$\bar{\rho}_2 = \frac{\|C_{2,\delta}\|\left(\bar{U}^{k_\Psi} + \bar{\gamma}\right)}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}}, \tag{6.18b}$$

with $\bar{\eta} \in \mathbb{R}_{>0}$, then $\sigma(x(t)) = 0_m$ for $t \geq t_0$.

By virtue of the above results, it is possible to enforce an ideal sliding mode $\sigma(x(t)) = 0_m$ for $t \geq t_2$ by applying the adaptive strategy described below Proposition 5.4. In particular, as soon as a practical sliding mode $\|\sigma(x(t))\| \leq \varsigma$ is enforced, the weight adaptation of the DNNs is stopped and the discontinuous control gain is adapted according to the following theorem.

Theorem 6.2. *Consider the nonlinear system in* (6.2) *and integral sliding variable σ in* (6.10)*. For $t > t_1$, with t_1 being the one defined in Proposition 5.4 , let* (6.2) *be controlled via ISM control law u in* (6.8)*, where the switching law is designed as*

$$u_r(t) = -\left(\hat{\rho}_1(t) + \hat{\rho}_2(t)\Big(\delta_a \left\|u_{n,a}\right\| + (1-\delta_a)\left\|u_{n,r}\right\|\Big)\right)\frac{\sigma(x(t))}{\|\sigma(x(t))\|},$$

where $\hat{\rho}_1(t_1)$ and $\hat{\rho}_2(t_1)$ are so that $\hat{\rho}_1(t_1) + \hat{\rho}_2(t_1)\Big(\delta_a \left\|u_{n,a}\right\| + (1-\delta_a)\left\|u_{n,r}\right\|\Big)$ satisfies Theorem 6.1. Moreover, let $\operatorname{vec}\left(\dot{V}_j\right) = 0$ for $j \in \{0, 1, \ldots, k_\Phi\}$ and $\operatorname{vec}\left(\dot{U}_j\right) = 0$ for $j \in \{0, 1, \ldots, k_\Psi\}$. If Assumptions 5.1 - 5.6 hold and the discontinuous control gain components are adapted according to

$$\dot{\hat{\rho}}_1(t) = \underset{\varrho_1}{\operatorname{proj}}\left(\left(\|C_{2,\delta}\|\,\bar{\gamma} + \alpha_1\right)\|\sigma\|\right), \tag{6.19a}$$

$$\dot{\hat{\rho}}_2(t) = \underset{\varrho_2}{\operatorname{proj}}\left(\left(\|C_{2,\delta}\|\,\bar{\gamma}\Big(\delta_a \left\|u_{n,a} + (1-\delta_a)\left\|u_{n,r}\right\|\right\|\Big) + \alpha_2\right)\|\sigma\|\right), \tag{6.19b}$$

with $\alpha_1, \alpha_2 \in \mathbb{R}_{>0}$ being constants acting as learning rates, $\bar{\gamma}$ being the one appearing in Assumption 5.2, $\varrho_1 \coloneqq \{r \in \mathbb{R}_{>0} : r \leq \bar{\rho}_1\}$ and $\varrho_2 \coloneqq \{r \in \mathbb{R}_{>0} : r \leq \bar{\rho}_2\}$, then a sliding mode $\sigma(x(t)) = 0_m$ is enforced for $t \geq t_2$, with $t_1 < t_2 < \infty$.

Proof. See Appendix B.4. □

6.1.4 Simulations

In this simulation, the control scheme in Figure 6.1 is assessed on a planar version of the Franka Emika Panda panda robot, whose state space model is given in (5.72). In particular, only the joints 2, 4, and 6 are controlled, while the others are kept locked, making the robot planar. Hence, it holds that $q, \dot{q}, \ddot{q} \in \mathbb{R}^3$, $x = \begin{bmatrix} q^\top & \dot{q}^\top \end{bmatrix}^\top \in \mathcal{X} \subset \mathbb{R}^6$, $f_1 : \mathcal{X} \to \mathbb{R}^3$, $f_2 : \mathcal{X} \to \mathbb{R}^3$, $\bar{B} : \mathcal{X} \to \mathbb{R}^{3\times 3}$. Since $f_1 \equiv \dot{q}$ is measurable, only the mappings f_2 and $\bar{B}$ are estimated via the DNNs $\hat{\Phi} : \mathcal{X} \to \mathbb{R}^3$ and $\hat{\Psi} : \mathcal{X} \to \mathbb{R}^9$, respectively. The two networks are characterized by $k_\Phi = k_\Psi = 2$ hidden layers with 16 neurons, and they are updated with adaptation rate matrices $\Gamma_{\Phi_j} = 10\, I$ and $\Gamma_{\Psi_j} = I$, with I denoting an identity matrix with suitable dimensions.

The objective of the simulation, which has a duration of 20 seconds and it is characterized by a time-step of 10^{-4} seconds, is to control the robot so that it tracks a desired trajectory given by

$$
\begin{aligned}
q^\star(t) &= \begin{bmatrix} 0.25\sin(2t) & -\frac{\pi}{2} + 0.75\cos(t) & \frac{\pi}{2} + 0.9\cos(t) \end{bmatrix}^\top \\
\dot{q}^\star(t) &= \begin{bmatrix} 0.5\cos(2t) & -0.75\sin(t) & -0.9\sin(t) \end{bmatrix}^\top \\
\ddot{q}^\star(t) &= \begin{bmatrix} -1\sin(2t) & -0.75\cos(t) & -0.9\cos(t) \end{bmatrix}^\top,
\end{aligned}
$$

while satisfying time-varying constraints on the joint variable vector q, denoted as $\underline{q} \in \mathbb{R}^3$ and $\bar{q} \in \mathbb{R}^3$ and defined as

$$
\underline{q}^\top(t) = \begin{cases} \begin{bmatrix} -1.76 & -3.07 & -0.0175 \end{bmatrix} & \text{if } t < 11 \text{ s} \\ \begin{bmatrix} -1.32 & -2.3 & -0.013 \end{bmatrix} & \text{if } 11 \leq t < 16 \text{ s} \\ \begin{bmatrix} -1.58 & -2.76 & -0.0157 \end{bmatrix} & \text{if } t \geq 16 \text{ s} \end{cases}
$$

$$
\bar{q}^\top(t) = \begin{cases} \begin{bmatrix} 1.76 & -0.06 & 3.75 \end{bmatrix} & \text{if } t < 11 \text{ s} \\ \begin{bmatrix} 1.32 & -0.045 & 2.812 \end{bmatrix} & \text{if } 11 \leq t < 16 \text{ s} \\ \begin{bmatrix} 1.58 & -0.054 & 3.375 \end{bmatrix} & \text{if } t \geq 16 \text{ s.} \end{cases}
$$

Moreover, the robot is affected by a matched disturbance

$$
h(t) = \begin{bmatrix} 0.05 + 0.05\cos(8t) & 0.075\sin(2t) & 0.01\cos(10t) \end{bmatrix}^\top.
$$

To accomplish that, the matrices appearing in the conventional sliding variable (6.5) are chosen as $C_{1,a} = C_{1,r} = C_{2,a} = C_{2,r} = I_3$, while the stabilizing control laws in (6.9a) are designed as

$$
\begin{aligned}
u_{n,r}(t) &= -\left(\mathrm{vec}^{-1}\left(\hat{\Psi}_{k_\Psi}\right)\right)^+ \left\{\hat{\Phi}_{k_\Phi} + K_{1,r}\Big(q(t) - q^\star(t)\Big) + K_{2,r}\Big(\dot{q}(t) - \dot{q}^\star(t)\Big) - \ddot{q}^\star(t)\right\}, \\
u_{n,a}(t) &= -\left(\mathrm{vec}^{-1}\left(\hat{\Psi}_{k_\Psi}\right)\right)^+ \left\{\hat{\Phi}_{k_\Phi} + K_a\Big(x(t) - x_s(t)\Big)\right\},
\end{aligned}
$$

where $K_{1,r} = K_{2,r} = 5\, I_3$, $K_a = 10\, I_6$. The safe state x_s has been selected as

$$
x_s(t) = x(t) + 0.2\frac{x(t) - x_c(t)}{\|x(t) - x_c(t)\|},
$$

while the safety threshold in (6.6) has been chosen as $d^\star = \sqrt{0.05}$. The discontinuous control gain is designed as $\rho = 0.5 + 1.5\,\|u_n\|$, while the threshold that stops the adaptation of the DNNs and triggers the adaptation of the discontinuous gain is $\varsigma = 0.002$.

The results of the simulation are reported in Figure 6.2. From them, it is possible to observe how the robot is successfully controlled toward the reference trajectory, while avoiding the time-varying non-admissible states. As for the sliding variable, it tends to zero after a first time interval in which the transient variable is not correctly updated due to the non accurate approximations produced by the DNNs. It is important to notice that, in the initial phase, it may be possible possible that the non admissible states are not avoided, due to an incorrect values of the weights. In case the avoidance of the non-admissible states is safety-critical, one could mitigate such a problem by relying on an available initial estimate of the DNNs weights.

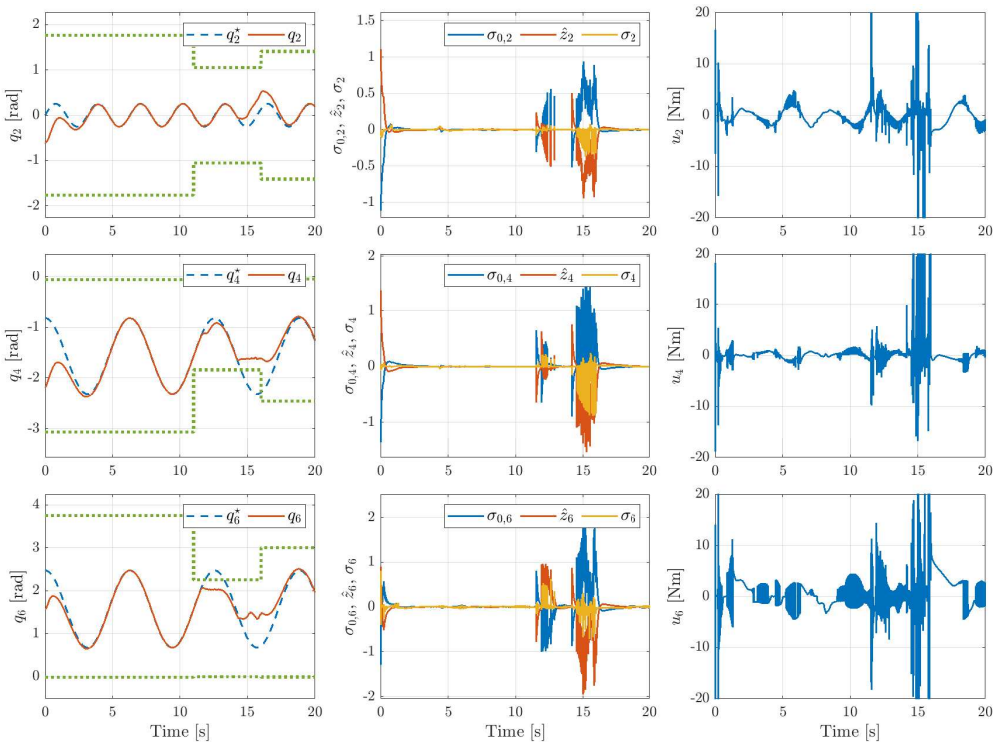


Figure 6.2: Time evolution of the joint positions, sliding variable components and control input when the planar version of the robot is controlled using the control scheme in Figure 6.1.

6.2 DNN based MPC/ISM

The idea of combining MPC and ISM has been introduced and expanded in the last decade in works like [107, 108, 109, 110, 111]. In particular, the idea in those works is to design a MPC/ISM control architecture like the one in Figure 6.3, in which an ISM controller is

employed to counteract the effect of the matched disturbances, while the unmatched ones are dealt via robust MPC with tightened constraints.

The complete knowledge of the nominal dynamics of the system is required for the design both MPC and ISM. Inspired by the DNN-ISM framework, a modified version of the MPC/ISM controller in which the model of the system is estimated online has been proposed in [112] to deal with soft state and input constraints. In this section, such a control architecture is presented and analyzed.

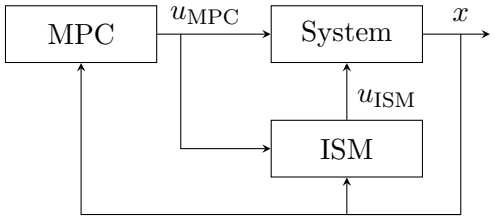


Figure 6.3: Block diagram of the MPC/ISM control architecture.

6.2.1 Problem Formulation

Consider the nonlinear perturbed system in (5.1), with $f_1(x(t)) = x_2(t)$, i.e.,

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ f_2(x(t),t) + \bar{B}(x(t),t)u(t) + h(x(t),t) \end{bmatrix}, \tag{6.20}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$, with $n = 2m$, is the state vector, while $u \in \mathbb{R}^m$ is the input vector. The drift dynamics f_2 and the control effectiveness $\bar{B}$ satisfy Assumption 5.1 and 5.2, respectively. As for the matched disturbance h, it is bounded as in Assumption 5.3.

When controlling the system in (6.20), it can be desirable to define some state and input constraint

$$x \in \mathcal{X}_c \subset \mathcal{X}, \tag{6.21a}$$

$$u \in \mathcal{U} \subset \mathbb{R}^m, \tag{6.21b}$$

with $\mathcal{X}_c$ and $\mathcal{U}$ being compact sets containing the origin. In this case, the set $\mathcal{U}$ is defined by putting constraints on each element of the input vector, i.e.,

$$\mathcal{U} \coloneqq \{u \in \mathbb{R}^m : \underline{u}_i \leq u_i \leq \bar{u}_i, \forall i\}, \tag{6.22}$$

where $\underline{u}_i, \bar{u}_i \in \mathbb{R}$, with $\underline{u}_i < \bar{u}_i$ represent the lower and upper bound for the i-th component of the control vector, with $i \in \{1, 2, \ldots, m\}$.

Inspired by [108, 109, 107], one can control system in (6.20), rejecting the matched disturbance while satisfying the constraints in (6.21b), implementing an MPC controller with ISM generations.

Let $x^\star = \begin{bmatrix}(x_1^\star)^\top & (x_2^\star)^\top\end{bmatrix}^\top \in \mathbb{R}^n$ be a desired state, and define the conventional sliding variable σ_0 in (2.69) as

$$\sigma_0(x(t)) = C_1\Big(x_1(t) - x_1^\star(t)\Big) + C_2\Big(x_2(t) - x_2^\star(t)\Big),$$

with $C_1 \in \mathbb{R}^{m\times m}$ and $C_2 \in \mathbb{R}^{m\times m}$ being design matrices. Moreover, C_2 satisfies Assumption 5.4. As for the transient variable in $z(x(t))$ in (2.69), it is characterized by dynamics

$$\dot{z}(x(t)) = C_1 x_2(t) + C_2\Big(f_2(x(t),t) + \bar{B}(x(t),t)u_{\text{MPC}}(t)\Big), \tag{6.23}$$

with $z(x(t_0)) = \sigma_0(x(t_0))$ and where $u_{MPC} \in \mathbb{R}^m$ is an MPC law, defined later. As for the complete control law, it is given by

$$\begin{aligned} u(t) &= u_n(t) + u_r(t) \\ &= u_{\text{MPC}}(t) - \rho\frac{\sigma(x(t))}{\|\sigma(x(t))\|}. \end{aligned} \tag{6.24}$$

Since part of the control law must be employed for compensating the matched disturbance and, at the same time, it is required that $u(t) \in \mathcal{U}$, for all $t \geq t_0$, then it is convenient to define a reduced constraint set for the input

$$\bar{\mathcal{U}}(t) \coloneqq \left\{u \in \mathbb{R}^m : \underline{u}_i + \rho\frac{\sigma_i(x(t))}{|\sigma_i(x(t))|} \leq u_i \leq \bar{u}_i - \rho\frac{\sigma_i(x(t))}{|\sigma_i(x(t))|}, \forall i\right\} \tag{6.25}$$

with $\sigma_i \in \mathbb{R}$ being the i-th component of the integral sliding variable, and design the MPC such that $u_{\text{MPC}}(t) \in \bar{\mathcal{U}}(t)$.

Since the ISM controller is able to provide robustness against the matched disturbance from the initial time instant, the MPC controller can be designed considering $h(x(t),t) = 0_m$. To this end, one can solve a Finite-Horizon Optimal Control Problem (FHOCP).

In particular, let $t_k \geq t_0$ be the time instant in which the signal u_{MPC} must be computed, then a suitably designed cost function has to be minimized with respect to the control sequence $u_{[t_k,t_{k+N-1}]}$, with $N \in \mathbb{N}_{>0}$ being the prediction horizon. Such a cost function can be defined as

$$J(x(t_k), u_{[t_k,t_{k+N-1}]}, N) = \int_{t_k}^{t_{k+N-1}} l(x(\tau),u(\tau))d\tau + V_f(x(t_{k+N})), \tag{6.26}$$

where $l : \mathcal{X}_c \times \bar{\mathcal{U}} \to \mathbb{R}_{\geq 0}$ is the *stage cost*, chosen as

$$l(x(\tau),u(\tau)) = \|x(\tau) - x^\star\|_Q^2 + \|u(\tau)\|_R^2$$

with $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ being positive definite matrices. As for $V_f : \mathcal{X}_c \to \mathbb{R}_{\geq 0}$, it is the so-called *terminal cost*, instrumental for ensuring the stability of the closed loop system, chosen, for example, as

$$V_f(x(t_{k+N})) = \|x(t_{k+N})\|_\Pi^2, \tag{6.27}$$

where $\Pi \in \mathbb{R}^{n\times n}$ is a positive definite matrix. Moreover, the so-called *terminal constraint* $x(t_{k+N}) \in \mathcal{X}_f$ must be introduced, where the terminal set $\mathcal{X}_f \subseteq \mathcal{X}_c$ is defined as

$$\mathcal{X}_f = \{x : \|x\|_\Pi^2 < \xi\}, \tag{6.28}$$

with $\xi \in \mathbb{R}_{>0}$. A possible choice for the terminal cost and the terminal constraint is the one proposed in [107].

Therefore, MPC problem can be formulated as

$$\begin{aligned} \min_{u_{[t_k,t_{k+N-1}]}} \quad & J(x(t_k), u_{[t_k,t_{k+N-1}]}, N) \\ \text{s.t.} \quad & \text{dynamics (6.20) with } h(x(t),t) = 0_m \\ & x(t_k) \in \mathcal{X}_c,\ \forall t_k \in [t_k, t_{k+N-1}] \\ & u(t_k) \in \bar{\mathcal{U}},\ \forall t_k \in [t_k, t_{k+N-1}] \\ & x(t_{k+N}) \in \mathcal{X}_f, \end{aligned} \tag{6.29}$$

and its result is the optimal control sequence $u^o_{[t_k,t_{k+N-1}]}$. Finally, according to the *Receding Horizon* principle, the control input applied to the system is the first element of the optimal control sequence, denoted by $u_{\text{MPC}}(t) = u^o(t_k)$, $\forall t \in [t_k, t_{k+1})$, with $u^o(t_k)$ resulting from (6.29).

6.2.2 The DNN-ISM based MPC scheme

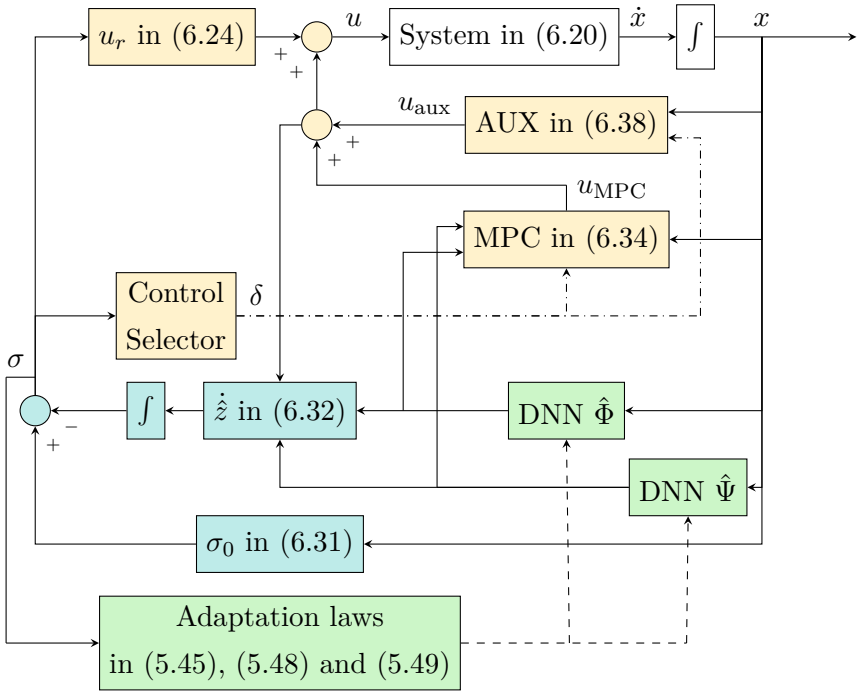


Figure 6.4: Block diagram of the DNN based MPC/ISM control architecture. The blocks associated with the control law are colored in yellow, the ones related to the sliding variable in blue, and the ones corresponding to the DNNs and their update in green.

Since the dynamics of the system (6.20) is assumed unknown, two DNNs are employed

to estimate it, as in (5.14) and (5.17). Hence, the integral sliding variable is defined as

$$\sigma(x(t)) = \sigma_0(x(t)) + \hat{z}(x(t)), \tag{6.30}$$

where σ_0 is defined as

$$\sigma_0(x(t)) = x_2(t), \tag{6.31}$$

while the dynamics of the transient variable depends on the DNNs as

$$\dot{\hat{z}}(x(t)) = \hat{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_{\mathrm{MPC},i}(t), \tag{6.32}$$

with initial conditions $\hat{z}(x(t_0)) = \sigma_0(x(t_0))$.

As for the control law, it is the one in (6.24), with integral sliding variable σ being defined as in (6.30). Recalling that $f_1(x(t)) = x_2(t)$ and, from (6.31), $C_2 = I_m$, one can choose the discontinuous control gain ρ as in Theorem 5.1. In particular, under the assumptions of the same theorem, selecting ρ as

$$\rho > \frac{\|I_m\| \left\{\bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \|u_{\mathrm{MPC}}\| + \bar{h}\right\} + \bar{\eta}}{\underline{\lambda}(C_2)\underline{\gamma}}, \tag{6.33}$$

with $\bar{\eta} \in \mathbb{R}_{>0}$, implies that $\sigma(x(t)) \to 0_m$ for $t \to \infty$. As a consequence, as introduced in Proposition 5.4, there exists $t_1 \geq t_0$ such that a practical sliding mode $\|\sigma(x(t))\| \leq \varsigma$, with $\varsigma \in \mathbb{R}_{>0}$.

Due to the particular choice of (6.20), the enforcement of a practical sliding mode $\|\sigma(x(t))\| \leq \varsigma$ implies that, in the worst case, the error on x_2 provided by the DNNs is equal ς. Such a fact is clear if one expands σ in the above practical sliding mode condition as

$$\begin{aligned}
\|\sigma(x(t))\| &= \left\| \int_{t_0}^{t} \dot{\sigma}_0(x(\tau)) - \dot{\hat{z}}(x(\tau)) d\tau \right\| \\
&= \left\| \int_{t_0}^{t} \dot{x}_2(\tau) - \hat{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_{\mathrm{MPC},i}(\tau) d\tau \right\| \\
&= \left\| \int_{t_0}^{t} \tilde{\Phi}^{[2]}_{k_\Phi} + \varepsilon^{[2]}_\Phi + \sum_{i=1}^{m} \left(\Psi^{[i]}_{k_\Psi} + \varepsilon^{(i)}_\Phi\right) u_i(\tau) + \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_{\mathrm{MPC},i}(\tau) d\tau \right\| \leq \varsigma.
\end{aligned}$$

This allows to formulate a reduced constraints set $\bar{\mathcal{X}}_c \subset \mathcal{X}_c$, obtained from the set $\mathcal{X}_c$ with a tightened boundary of $(t_{k+N-1} - t_k)\varsigma$ in the x_1 direction and ς in the x_2 direction. Then, the DNN-ISM based MPC problem can be formulated as

$$\begin{aligned}
\min_{u_{[t_k, t_{k+N-1}]}} \quad & J(x(t_k), u_{[t_k, t_{k+N-1}]}, N) \\
\text{s.t.} \quad & \dot{x} = \begin{bmatrix} x_2 \\ \hat{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_i \end{bmatrix} \\
& x(t_k) \in \bar{\mathcal{X}}_c, \, \forall t_k \in [t_k, t_{k+N-1}] \\
& u(t_k) \in \bar{\mathcal{U}}, \, \forall t_k \in [t_k, t_{k+N-1}] \\
& x(t_{k+N}) \in \mathcal{X}_f,
\end{aligned} \tag{6.34}$$

where the cost J and the terminal set $\mathcal{X}_f$ are chosen as in (6.26) and (6.28), respectively. Then, similarly as in the ideal case, the control input is chosen as the first element of the optimal control sequence $u^o_{[t_k,t_{k+N-1}]}$, denoted as $u^o(t_k)$, resulting from (6.34), i.e.,

$$u_{\text{MPC}}(t) = u^o(t_k), \forall t \in [t_k, t_{k+1}]. \tag{6.35}$$

If, on the one hand, one can formulate the MPC problem in (6.34) from time instant t_1 in which the practical sliding mode $\|\sigma(x(t))\| \leq \varsigma$ is enforced, on the other it is difficult to formulate the tightened set $\bar{\mathcal{X}}_c$ for $t \in [t_0, t_1)$.

In [112], a solution that relies on the use of an auxiliary control law, depicted in Figure 6.4, has been proposed. In particular, the control law in (6.24) is modified as

$$\begin{aligned} u(t) &= u_n(t) + u_r(t) \\ &= \delta(t)u_{\text{MPC}}(t) + (1-\delta(t))u_{\text{aux}}(t) - \rho\frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \end{aligned} \tag{6.36}$$

where $u_{\text{aux}} \in \mathbb{R}^m$ is an auxiliary control law that stabilizes the system while the weights of the DNNs are adapting. As for $\delta \in \{0, 1\}$, it is defined as

$$\delta(t) = \begin{cases} 0 & \text{if } \|\sigma(x(t))\| > \varsigma, \\ 1 & \text{if } \|\sigma(x(t))\| \leq \varsigma, \end{cases} \tag{6.37}$$

having ς determining the trading-off utilization time and the conservativeness if the MPC controller. In particular, an higher value of ς implies a lower value of t_1, but a tighter set $\bar{\mathcal{X}}_c$.

The choice of the auxiliary control law u_{aux} depends on the behavior that one wants to impose while the weights of the DNNs are adapting. A choice could be the feedback linearizing control law

$$u_{\text{aux}}(t) = -\left(\text{vec}^{-1}\left(\hat{\Psi}_{k_\Psi}\right)\right)^+ \left(\hat{\Phi}^{[2]}_{k_\Phi} + K(x(t) - x^\star)\right), \tag{6.38}$$

where $K \in \mathbb{R}^{m\times n}$ is a suitably chosen design matrix, while the operator $(\cdot)^+$ represents the Moore-Penrose pseudoinverse operator defined in Definition 4.2. Clearly, the law (6.38) does not provide guarantees on the satisfaction on the state and input constraints are provided in the case of $\|\sigma\| > \varsigma$. Hence, the use of an auxiliary control law is not suitable for safety-critical constraints. Nevertheless, the DNN based MPC/ISM presented in this section is very useful when one wants to apply soft constraints on a system with fully unknown dynamics and it represents a first step forward toward formulating robust DNN based MPC/ISM control schemes for uncertain systems.

6.2.3 Simulations

The DNN-ISM based MPC control architecture in Figure 6.4 has been assessed in simulation on the Duffing oscillator, described by the dynamics in (2.3) and subject to soft state and input constraints defined as

$$\mathcal{X}_c \coloneqq \{(x_1, x_2) \in [-1.8, 1.8] \times [-1.57, 1.57]\}$$

$$\mathcal{U} \coloneqq \{u \in \mathbb{R} : |u| \leq 10\},$$

respectively. The model is assumed fully unknown and it is estimated by means of the two networks in Section 5.5.1.

The objective of the simulation, whose duration is 30 seconds, is to control the system such that it reaches a time-varying equilibrium state defined as $x^\star = \begin{bmatrix} 0.5 & 0 \end{bmatrix}^\top \in \mathcal{X}_c$ for $t \in [0, 10)$ seconds, $x^\star = \begin{bmatrix} 1.95 & 0 \end{bmatrix}^\top \notin \mathcal{X}_c$ for $t \in [10, 20)$ seconds, and $x^\star = \begin{bmatrix} 0.35 & 0 \end{bmatrix}^\top \in \mathcal{X}_c$ for $t \geq 20$ seconds, while being subject to a matched disturbance $h(t) = 0.1 \sin(t)$. To accomplish that, the control law is defined as in (6.36), where the MPC has a cost which is quadratic and characterized by weights $Q = 100 I_2$ and $R = 5$, the prediction horizon is chosen as $N = 20$, with a discretization step $t_{k+1} - t_k = 0.05$ seconds. The auxiliary control law u_{aux} is chosen as in (5.71), while the flag δ is chosen as in (6.37), with $\varsigma = 0.005$. As for the discontinuous control gain, it has been chosen as $\rho = 0.15 + 0.05 \|u_n\|$. The simulation time-step is set to 10^{-4} seconds.

The results of the simulation are presented in Figure 6.5, in which the time evolution of the components of the sliding variable, the state trajectory, and the constrained control input are presented. Moreover, in Figure 6.6 is depicted a zoom of the norm of the sliding variable the first instants in which DNNs are being adapted. From that it is possible to see that after 0.04 seconds the MPC controller is always employed since $\|\sigma\| \leq \varepsilon$. These results highlight the capability of the control algorithm in Figure 6.4 of satisfactorily solving the constrained optimization problem although the system model is completely unavailable.

6.3 DNN-ISM with Barrier Functions

The last DNN-ISM based architecture that takes into account constraints, presented in this section, relies on the use of a Barrier Lyapunov Function (BLF) for limiting the raise of the sliding variable during the learning transients, while the nominal control law of the ISM is generated solving a Quadratic Programming (QP) problem whose constraints are defined a CLF and a CBF.

6.3.1 Problem Formulation

Consider the nonlinear control-affine system

$$\dot{x}(t) = f(x(t), t) + \bar{B}(x(t), t)u(t) + h(x(t), t), \tag{6.39}$$

where $x \in \mathcal{X} \subset \mathbb{R}^m$ is the state vector, $u \in \mathbb{R}^m$ is the control input vector, while $f : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ and $\bar{B} : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{m \times m}$ are, respectively, the drift and control effectiveness term, satisfying Assumption 5.1 and Assumption 5.2. As for $h : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$, it represents the matched disturbance acting on the system and it satisfies Assumption 5.3.

The objective is to control the system in (6.39) toward the origin, while maintaining the system states in an *admissible set* $\mathcal{X}_a \subset \mathcal{X}$, having $x(t) \in \mathcal{X}_a$, for all $t \geq t_0$ or, in other

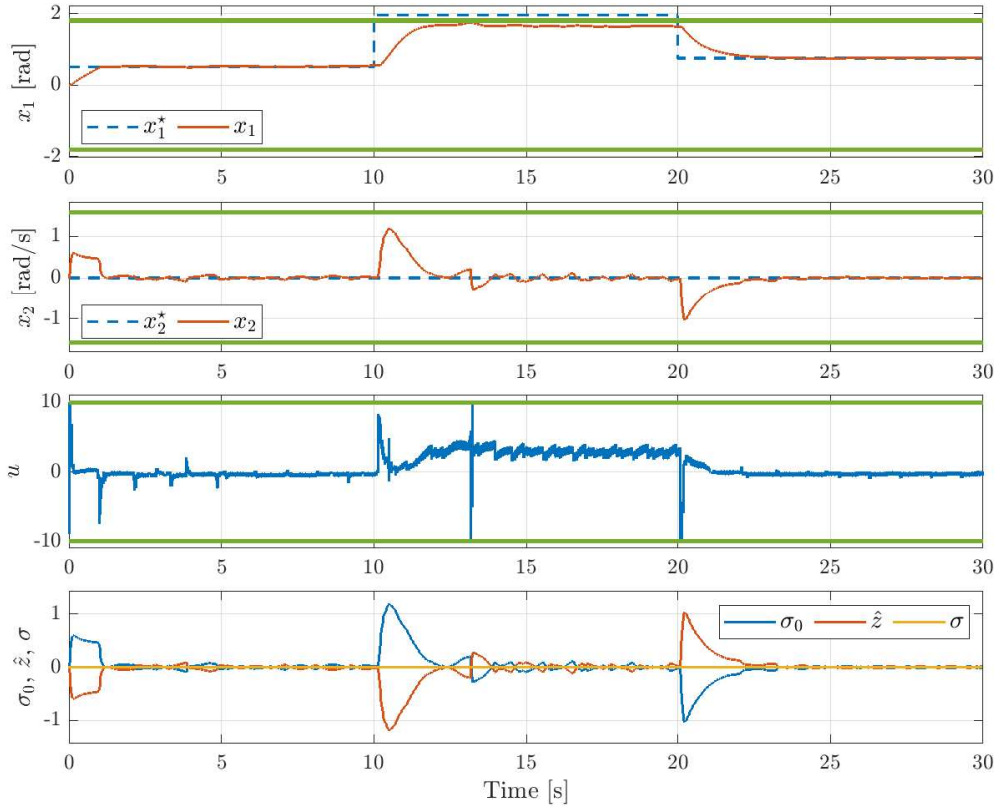


Figure 6.5: Time evolution of the system states, control input, and sliding variable components during the simulation. The state and input constraints $\mathcal{X}$ and $\bar{\mathcal{U}}$ are depicted with green solid lines.

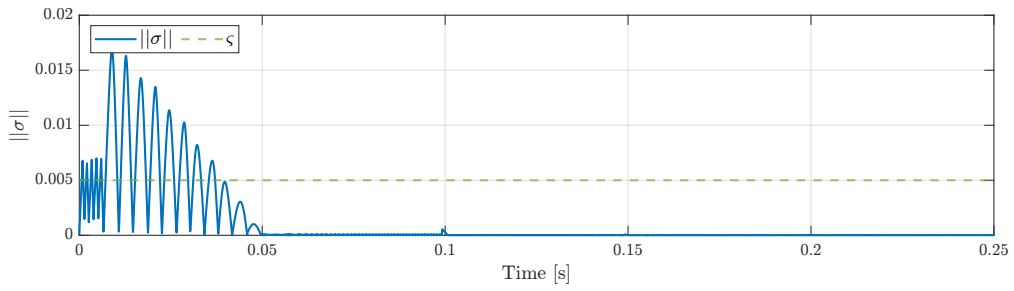


Figure 6.6: Zoom of the norm of the sliding variable during the adaptation transient, along the MPC activation threshold ς

words, to make the set $\mathcal{X}_a$ *forward invariant*. To accomplish that, it is possible to design an ISM control law

$$u(t) = u_n(t) + u_r(t) = u_{\text{qp}}(t) - \rho \frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \tag{6.40}$$

where the integral sliding variable $\sigma(x(t))$ is defined as the difference between the conven-

tional sliding variable σ_0, and the transient variable z. In this case

$$\sigma_0(x(t)) = x(t), \tag{6.41}$$

leading to

$$\begin{aligned}\sigma(x(t)) &= \sigma_0(x(t)) - z(x(t)) \\ &= x(t) - x(t_0) - \int_{t_0}^{t} f(x(\tau),\tau) + \bar{B}(x(\tau),\tau)u_{\text{qp}}(\tau)d\tau.\end{aligned} \tag{6.42}$$

The signal u_{qp} is a nominal control law that, in the case $h(x(t),t) = 0_m$, stabilizes the state in the origin, while ensuring $x(t) \in \mathcal{X}_a$, for all $t \geq t_0$. Such a signal is obtained is used by solving an QP problem which includes a CLF and a CBF [113].

Since the components of the dynamics (6.39) are not available, the DNN-ISM method described in Chapter 5 could be employed. However, this last one should be modified so that it is possible to ensure that $x(t) \in \mathcal{X}_a$. Such a modification involves the use of a so-called BLF [114].

6.3.2 Preliminaries on BLFs, CLFs, and CBFs

Before introducing the complete control methodology, some preliminary concepts about CLFs, CBFs, and BLFs, are presented in the following.

Barrier Lyapunov Functions

Let $\Omega \subset \mathbb{R}^m$ be an an open set containing the origin as an interior point. A BLF is a smooth scalar positive definite function $\beta : \Omega \to \mathbb{R}$ such that

$$\lim_{y \to \partial\Omega} \beta(y) = +\infty. \tag{6.43}$$

where $\partial\Omega$ denotes the boundary of Ω. In particular, a BLF satisfies that $\beta(y(t)) \leq \bar{\beta}$, for $t \geq t_0$ along the trajectories of $y(t)$, with $\bar{\beta} \in \mathbb{R}_+$ if $y(0) \in \Omega$. A BLF can be used to certify stability of a desired equilibrium point, together with the invariance property of Ω [114, 115].

Control Lyapunov and Control Barrier Functions

In many cases it is possible to describe the admissible region $\mathcal{X}_a \subset \mathbb{R}^m$ via the superlevel set of a suitably smooth function $\vartheta : \mathbb{R}^m \to \mathbb{R}$, i.e.,

$$\mathcal{X}_a \coloneqq \{x \in \mathbb{R}^m : \vartheta(x) \geq 0\}.$$

Definition 6.1 (CBF [113]). *Consider the admissible set $\mathcal{X}_a \subset \mathbb{R}^m$. Then, the function $\vartheta : \mathbb{R}^m \to \mathbb{R}$ is a CBF if there exists a control law $u(t)$ such that $\dot{\vartheta}(x,u) \geq -\alpha(\vartheta(x))$ or, in terms of Lie derivatives*

$$\sup_{u \in \mathcal{U}} L_f\vartheta(x) + \sum_{i=1}^{m} L_{\bar{B}^{(i)}}\vartheta(x)u_i \geq -\alpha(\vartheta(x)), \tag{6.44}$$

where $\alpha : \mathbb{R} \to \mathbb{R}$ *is an extended class* $\mathcal{K}_\infty$ *function, i.e., strictly increasing function on* $\mathbb{R}$ *with* $\alpha(0) = 0$.

From Definition 6.1, it is possible to identify the set of control vectors that makes the set $\mathcal{X}_a$ invariant for every point in the state space as

$$\mathcal{I}(x) = \left\{ u \in \mathcal{U} : L_f \vartheta(x) + \sum_{i=1}^{n} L_{\bar{B}^{(i)}} \vartheta(x) u_i \geq -\alpha(\vartheta(x)) \right\}, \tag{6.45}$$

where $\mathcal{U}$ is the set of input constraints. In the following, this last one is defined as a ball centered in the origin

$$\mathcal{U} \coloneqq \left\{ u \in \mathbb{R}^m : \|u\| \leq \bar{U} \right\}. \tag{6.46}$$

Similarly, one can define the set of controllers that stabilize system (6.39). Such a set is defined with respect to a Lyapunov function $\mathcal{V} : \mathcal{X} \to \mathbb{R}_{>0}$, which certifies stability when $u(t)$ satisfies $\dot{\mathcal{V}}(x, u) < 0$. Hence, the set of stabilizing controllers with respect to v can be expressed in function of its Lie derivatives as

$$\mathcal{S}(x) = \left\{ u \in \mathcal{U} : L_f \mathcal{V}(x) + \sum_{i=1}^{m} L_{\bar{B}^{(i)}} \mathcal{V}(x) u_i \leq -\xi(\mathcal{V}(x)) \right\}, \tag{6.47}$$

where $\xi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}_\infty$ function.

The following lemma shows how it is possible to find a controller which, at the same time, stabilizes the system (6.39) and makes $\mathcal{X}_a$ forward invariant.

Lemma 6.1 (CBF/CLF QP [116]). *It is possible to generate controller* $u^\star \in \mathcal{I}(x) \cap \mathcal{S}(x)$, *with* $\mathcal{I}(x)$ *and* $\mathcal{S}(x)$ *defined as in* (6.45) *and* (6.47), *respectively, by solving the QP problem*

$$u^\star = \underset{u,\delta}{\operatorname{argmin}} \quad J(u, \delta) = \frac{1}{2} \|u\|_R^2 + \frac{l}{2} \delta^2 \tag{6.48a}$$

$$\text{s.t.} \quad L_f \mathcal{V}(x) + \sum_{i=1}^{m} L_{\bar{B}^{(i)}} \mathcal{V}(x) u_i \leq \delta - \xi(\mathcal{V}(x)) \tag{6.48b}$$

$$L_f \vartheta(x) + \sum_{i=1}^{m} L_{\bar{B}^{(i)}} \vartheta(x) u_i \geq -\alpha(\vartheta(x)) \tag{6.48c}$$

$$(u, \delta) \in \mathcal{U} \times \mathbb{R}_{\geq 0} \tag{6.48d}$$

where $R \in \mathbb{R}^{m \times m}$ *is the symmetric positive definite input weight matrix, while* $l \in \mathbb{R}_{>0}$ *is the weight associated with the optimization variable* δ, *which is the so-called* slack variable, *i.e., a relaxation variable that can assume positive values.*

Note that, the solution of (6.48) has $\delta > 0$ when the forward invariance of $\mathcal{X}_a$ is favored over the convergence rate of $\mathcal{V}(x)$ imposed by $-\xi(\mathcal{V}(x))$. Moreover, if for some $x \in \mathbb{R}^m$ it holds that $\mathcal{I}(x) \cap \mathcal{S}(x) = \emptyset$, meaning that constraints (6.48b) and (6.48c) are conflicting when $\delta = 0$, then $\delta > 0$ allows to find a feasible solution which is makes $\mathcal{X}_a$ forward invariant, but $\mathcal{V}(x)$ non-decreasing.

The QP problem (6.48) can be employed as a filter for a stabilizing control control law $u_a \in \mathcal{S}(x)$ and possibly $u_a \notin \mathcal{I}(x)$, referred to as *auxiliary controller*, such that $u_a \in \mathcal{I}(x)$, as described in the following lemma.

Lemma 6.2 (CBF/CLF QP as filter [113]). *Assume that there exists a known stabilizing controller $u_a \in \mathcal{S}(x)$ which may not make $\mathcal{X}_a$ forward invariant. Then, if $\mathcal{I}(x) \cap \mathcal{S}(x) \neq \emptyset$, one can generate a control signal $u^\star \in \mathcal{I}(x) \cap \mathcal{S}(x)$ applying the minimum perturbation to u_a obtained solving the QP problem* (6.48) *with the modified cost function*

$$J(u, \delta) = \frac{1}{2} \|u - u_a\|_R^2 + \frac{l}{2}\delta^2. \tag{6.49}$$

6.3.3 The DNN-ISM scheme with CBFs, CLFs, and BLFs

In the following, the DNN-ISM control scheme that relies on CBFs, CLFs, and BLFs, depicted in Figure 6.7, is presented. Such a scheme is composed by a BLF-based DNN-ISM controller that, relying on estimates DNNs, reject the disturbance. The stabilizing law of the DNN-ISM is instead generated by solving a CBF/CLF QP problem which takes into account the approximation uncertainties.

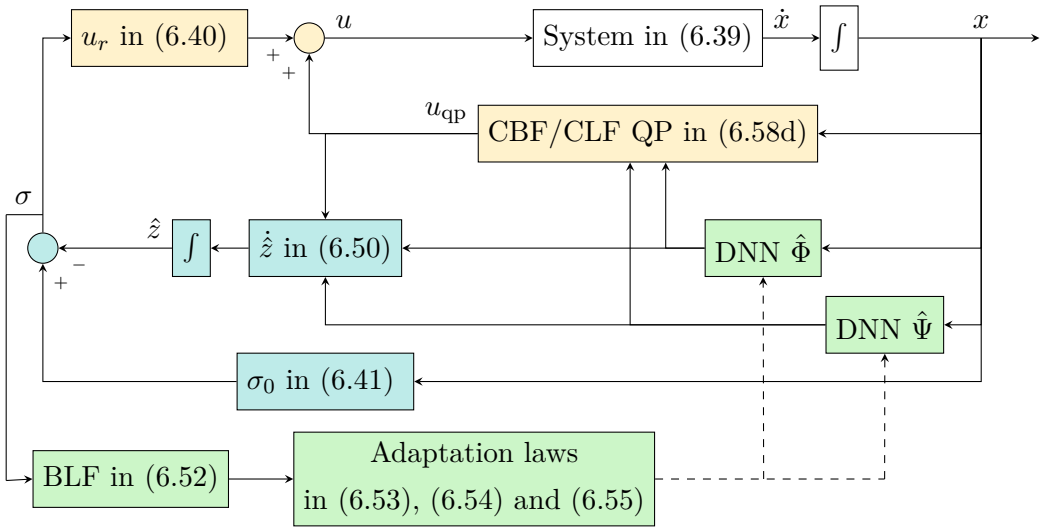


Figure 6.7: Block diagram of the BLF based DNN-ISM control architecture with CBF/CLF based QP. The yellow blocks are related to the control law, the green ones to the DNNs and their adaptation, while the blue ones are associated with the sliding variable.

The BLF-based DNN-ISM scheme

Since the dynamics terms f and $\bar{B}$ in (6.39) are unknown, they are estimated using two DNNs $\hat{\Phi}$ and $\hat{\Psi}$, as in (5.14) and (5.17). Note that, due to the structure of the considered

system, it is not required to divide the components of the last layer of $\hat{\Phi}$ as did in Chapter 5, hence $\hat{\Phi}_{k_\Phi}^{[1]} \equiv \hat{\Phi}_{k_\Phi}^{[2]} \equiv \hat{\Phi}_{k_\Phi}$.

The estimates are then employed for the computation of the integral sliding variable, choosing

$$\dot{\hat{z}}(x(t)) = \hat{\Phi}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}_{k_\Psi}^{[i]} u_{\mathrm{qp,i}}(t), \tag{6.50}$$

with $\hat{z}(x(t_0)) = \sigma_0(x(t_0))$. Recalling that $\sigma_0(x(t))$ is defined as in (6.41), it holds that

$$\begin{aligned}\sigma(x(t)) &= \sigma_0(x(t)) - z(x(t)) \\ &= x(t) - x(t_0) - \int_{t_0}^{t} \hat{\Phi}_{k_\Phi} + \sum_{i=1}^{m} \hat{\Psi}_{k_\Psi}^{[i]} u_{\mathrm{qp,i}}(\tau) d\tau.\end{aligned} \tag{6.51}$$

In order to successfully define the CBF/CLF QP, described in the next subsection, which is required to generate u_{qp} so that the controlled system is stable and the set $\mathcal{X}_a$ is forward invariant, it required that, at every time instant, the approximation error of the DNNs is bounded by a known constant.

By virtue of the integral sliding variable definition in (6.51), this can be achieved by designing a BLF $\beta : \mathbb{R}^m \to \mathbb{R}$ which limits σ into the set

$$\Omega_\sigma \coloneqq \{\sigma \in \mathbb{R}^m : \|\sigma\| \leq \varepsilon_\sigma\},$$

with $\varepsilon_\sigma \in \mathbb{R}_{>0}$. In particular, β must be chosen so that

$$\lim_{\|\sigma\| \to \varepsilon_\sigma} \beta(\sigma(x(t))) = +\infty. \tag{6.52}$$

The BLF is directly employed in the design of the weights adaptation law. In particular, the of the DNN $\hat{\Phi}$ are updated according to

$$\mathrm{vec}\left(\dot{\hat{V}}_j\right) = \underset{\mathcal{B}_{\Phi_j}}{\mathrm{proj}}\left(\Gamma_{\Phi_j} (\Lambda_{\Phi_j})^\top \frac{d\beta}{d\|\sigma\|^2}\sigma\right), \tag{6.53}$$

for $j \in \{0, 1, \ldots, k_\Phi\}$, where $\Gamma_{\Phi_j} \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}} \times L_{\Phi_j} L_{\Phi_{j+1}}}$ is the adaptation rate matrix, while $\Lambda_{\Phi_j} \in \mathbb{R}^{m \times L_{\Phi_j} L_{\Phi_{j+1}}}$ is the one defined in (5.26b).

As for the $\hat{\Psi}$, for $j \in \{0, 1, \ldots, k_\Psi - 1\}$, the weights are adapted according to

$$\mathrm{vec}\left(\dot{\hat{U}}_j\right) = \underset{\mathcal{B}_{\Psi_j}}{\mathrm{proj}}\left(\Gamma_{\Psi_j}\left(\sum_{i=1}^{m} u_{\mathrm{qp},i} (\Lambda_{\Psi_j}^{[i]})^\top\right) \frac{d\beta}{d\|\sigma\|^2}\sigma\right), \tag{6.54}$$

with $\Gamma_{\Psi_j} \in \mathbb{R}^{L_{\Psi_j} L_{\Psi_{j+1}} \times L_{\Psi_j} L_{\Psi_{j+1}}}$ is the adaptation rate matrix, $\Lambda_{\Psi_j} \in \mathbb{R}^{m \times L_{\Psi_j} L_{\Psi_{j+1}}}$ is the one in (5.37b). For $j = k_\Psi$, the weight sub-matrices are characterized by dynamics

$$\mathrm{vec}\left(\dot{\hat{U}}_j^{[i]}\right) = \underset{\mathcal{B}_{\Psi_j}}{\mathrm{proj}}\left(\Gamma_{\Psi_{k_\Psi}} u_{\mathrm{qp},i} (\Lambda_{\Psi_j}^{[i]})^\top \frac{d\beta}{d\|\sigma\|^2}\sigma\right), \tag{6.55}$$

with $i \in \{1, 2, \ldots, m\}$, where $\Gamma_{\Psi_{k_\Psi}} \in \mathbb{R}^{m L_{k_\Psi} \times m L_{k_\Psi}}$ is the diagonal matrix of the adaptation rates, while $\Lambda_{\Psi_{k_\Psi}}^{[i]} \in \mathbb{R}^{m \times m L_{k_\Psi}}$ is the one defined below equation (5.37).

Theorem 6.3. *Consider the nonlinear system in* (6.39)*, controlled via ISM control law* u *in* (6.40)*, with integral sliding variable* σ *defined as in* (6.51)*. Moreover, the DNNs weights are adapted according to the adaptation laws* (6.53)*,* (6.54)*, and* (6.55)*. If Assumptions 5.1 - 5.6 and Proposition 5.2 hold,* β *is chosen so that it satisfies* (6.52)*, and*

$$\rho = \frac{\bar{c}_\Phi + \bar{\varepsilon}_\Phi + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi)\,\|u_{\mathrm{qp}}\| + \bar{h} + \bar{\eta}}{\underline{\gamma}},$$

with $\bar{\eta} \in \mathbb{R}_{>0}$ *being a design parameter, then,* $\sigma(x(t)) \to 0_m$ *for* $t \to \infty$*, and it is guaranteed that* $\|\sigma(x(t))\| \leq \varepsilon_\sigma$*.*

Proof. See appendix B.5 □

Before introducing the QP problem that is solved to generate the control law u_{qp}, the following assumption needs to be introduced.

Assumption 6.2. *The set* $\mathcal{X}_a$ *can be made forward invariant by the control* $u_{\mathrm{qp}} \in \bar{\mathcal{U}} \subset \mathcal{U}$*, with* $\mathcal{U}$ *being the one in* (6.46) *and* $\bar{\mathcal{U}}$ *defined as*

$$\bar{\mathcal{U}} \coloneqq \left\{ u \in \mathbb{R}^m : \|u\| \leq \frac{\bar{U} - \rho_1}{1 + \rho_2} \right\}, \tag{6.56}$$

where $\rho_1 = \frac{\bar{c}_\Phi + \bar{\varepsilon}_\Phi + \bar{h} + \bar{\eta}}{\underline{\gamma}} \in \mathbb{R}_{>0}$ *and* $\rho_2 = \frac{m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi)}{\underline{\gamma}} \in \mathbb{R}_{>0}$ *are derived from the discontinuous control gain* ρ *in Theorem 6.3.*

The modified CBF/CLF QP problem

Since the dynamics of (6.39) is unknown, one has to rely on the estimates obtained from the neural networks to solve the QP in (6.48). Recalling that the integral sliding variable in (6.51) represents the approximation error of the network and that Theorem 6.3 ensures that $\|\sigma(x(t))\| \leq \varepsilon_\sigma$, the original barrier function $\vartheta(x)$ has to be modified in order to prevent the system trajectory to leave the set $\mathcal{X}_a$.

Theorem 6.4. *Consider system in in* (6.39)*, controlled with BLF-based DNN-ISM controller with gain chosen as in Theorem 6.3. Moreover, consider the reduced input constraints set* $\bar{\mathcal{U}}$ *defined in Assumption 6.2 and let* $\hat{\vartheta} : \mathbb{R}^m \to \mathbb{R}$ *be a CBF such that a tightened admissible set* $\hat{\mathcal{X}}_a \subset \mathcal{X}_a$ *can be defined as*

$$\hat{\mathcal{X}}_a \coloneqq \left\{ \hat{\vartheta}(x) \geq 0 \right\} = \mathcal{X}_a \setminus \{ x \in \mathcal{X}_a : \mathrm{dist}(x, \partial\mathcal{X}_a) \leq \varepsilon_\sigma \}. \tag{6.57}$$

If the the signal u_{qp} *is generated solving the QP problem*

$$u_{\mathrm{qp}} = \underset{u,\delta}{\mathrm{argmin}}\ \frac{1}{2}\|u\|_R^2 + \frac{l}{2}\delta^2 \tag{6.58a}$$

$$\text{s.t. } L_{\hat{\Phi}_{k_\Phi}} \mathcal{V}(x) + \sum_{i=1}^{m} L_{\hat{\Psi}^{[i]}_{k_\Psi}} \mathcal{V}(x) u_i \leq \delta - \xi(\mathcal{V}(x)) \tag{6.58b}$$

$$L_{\hat{\Phi}_{k_\Phi}} \hat{\vartheta}(x) + \sum_{i=1}^{m} L_{\hat{\Psi}^{[i]}_{k_\Psi}} \hat{\vartheta}(x) \geq -\alpha(\hat{\vartheta}(x)) \tag{6.58c}$$

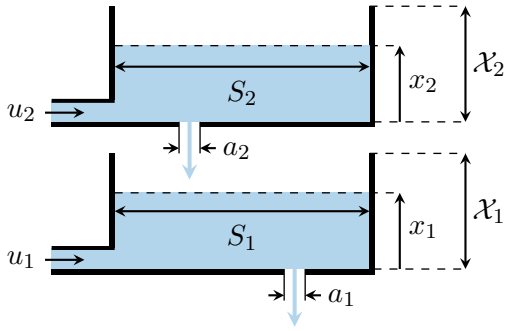


Figure 6.8: Graphical representation of the double tank system employed in the simulation.

$$(u, \delta) \in \bar{\mathcal{U}} \times \mathbb{R}_{\geq 0} \tag{6.58d}$$

where $\xi : \mathbb{R} \to \mathbb{R}$ and $\alpha : \mathbb{R} \to \mathbb{R}$ are, respectively, class $\mathcal{K}_\infty$ and extended class $\mathcal{K}_\infty$ functions, and $\mathcal{V} : \mathcal{X} \to \mathbb{R}_{>0}$ is a Lyapunov function that ensures the stability of the controlled system, then the forward invariance of the original admissible set $\mathcal{X}_a$ is guaranteed.

Proof. See Appendix B.6. □

6.3.4 Simulations

The control scheme in Figure 6.7 has been tested on the double tank system (inspired by [117]), depicted in Figure 6.8 and described by

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{a_1}{S_1}\sqrt{2gx_1(t)} + \frac{a_2}{S_2}\sqrt{2gx_2(t)} + \frac{1}{S_1}u_1(t) + h_1(t) \\ -\frac{a_2}{S_2}\sqrt{2gx_2(t)} + \frac{1}{S_2}u_2(t) + h_2(t) \end{bmatrix} \tag{6.59}$$

where x_1, x_2 [m] are the water levels of the two tanks, $S_1 = 0.017\,[\mathrm{m}^2]$, $S_2 = 0.031\,[\mathrm{m}^2]$ are the tanks cross-sections, $a_1 = a_2 = 0.007[\mathrm{m}^2]$ are the sections of the output valves, and $g = 9.81[\mathrm{m/s}^2]$ is the gravitational acceleration. The input variables $u_1, u_2\,[\mathrm{m}^3/\mathrm{s}]$ represent the input flows, and the disturbance is chosen as $h_1 = 0.025\sin(6\pi t)$, $h_2 = 0.015\cos(8\pi t)$.

The system in (6.59) can be expressed in the form (6.39), with

$$f(x) = \begin{bmatrix} -\frac{a_1}{S_1}\sqrt{2gx_1} + \frac{a_2}{S_2}\sqrt{2gx_2} \\ -\frac{a_2}{S_2}\sqrt{2gx_2} \end{bmatrix}, \quad \bar{B}(x) = \begin{bmatrix} \frac{1}{S_1} & 0 \\ 0 & \frac{1}{S_2} \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad h = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}.$$

The drift dynamics and the control effectiveness matrix are assumed unknown and they are estimated using two DNNs $\hat{\Phi}$ and $\hat{\Psi}$, characterized by $k_\Phi = k_\Psi = 2$ hidden layers with 8 neurons each. The weights of such DNNs are updated relying on adaptation matrices $\Gamma_{\Phi_j} = \Gamma_{\Psi_j} = 10\,I$, while the and BLF is chosen as

$$\beta(\sigma) = -\log\left(\frac{\varepsilon_\sigma^2}{\varepsilon_\sigma^2 - \|\sigma\|^2}\right),$$

with $\varepsilon_\sigma = 0.05$.

The objective of the simulation, whose duration is 1.5 seconds with a time-step of $5{\cdot}10^{-4}$ seconds, is to follow a piece-wise constant reference signal $x^\star \in \mathbb{R}^2$, defined as

$$x^\star(t) = \begin{cases} \begin{bmatrix} 0.8 & 0.4 \end{bmatrix}^\top \notin \mathcal{X}_a & \text{if } t \in [0, 0.25) \\ \begin{bmatrix} 0.4 & 0 \end{bmatrix}^\top \notin \mathcal{X}_a & \text{if } t \in [0.25, 0.6) \\ \begin{bmatrix} 0.5 & 0.185 \end{bmatrix}^\top \in \mathcal{X}_a & \text{if } t \geq 0.6 \end{cases}$$

while satisfying state and input constraints, defined as $\mathcal{X}_a \coloneqq \left\{x \in \mathcal{X} : x \in [0.05, 0.7]^2\right\}$ and $\mathcal{U} \coloneqq \{\|u\| \leq 0.5\}$, respectively. For the smooth set in (6.45), the CBF has been chosen as $\vartheta(x) = 0.325^4 - (x_1 - 0.375)^4 - (x_2 - 0.375)^4$. The robust CBF used for defining the tightened state constraints set (6.57), is $\hat{\vartheta}(x) = 0.275^4 - (x_1 - 0.375)^4 - (x_2 - 0.375)^4$ with $\alpha(\hat{\vartheta}) = \hat{\vartheta}$, while the CLF in (6.58b) is $\mathcal{V} = \|x - x^\star\|^2$ with $\xi(\mathcal{V}) = 0.01\mathcal{V}$. The weights of the cost (6.58a) have been chosen as $R = 100\, I_2$ and $l = 100$. The discontinuous control gain has been chosen setting $\rho_1 = \rho_2 = 0.05$, having the tightened input set being defined as $\bar{\mathcal{U}} \coloneqq \{u \in \mathbb{R}^2 : \|u\| \leq 0.4285\}$.

The results of the simulation are reported in Figure 6.9 and Figure 6.10. In particular, the former shows how the states are successfully steered toward the desired equilibrium when this last one belongs to the admissible set $\mathcal{X}_a$, while in the other case the states stops at $\partial\hat{\mathcal{X}}_a$, defined by the CBF $\hat{\vartheta}$. Moreover, the third and fourth plot of Figure 6.9 show how both the complete control signal u and the nominal control law q_{qp} resulting from (6.58) satisfy the constraint sets $\mathcal{U}$ and $\bar{\mathcal{U}}$, respectively. Finally, from Figure 6.10 it is possible to see how, even during the first transient in which the DNNs are adapting, the BLF maintains the norm of the sliding variable bounded.

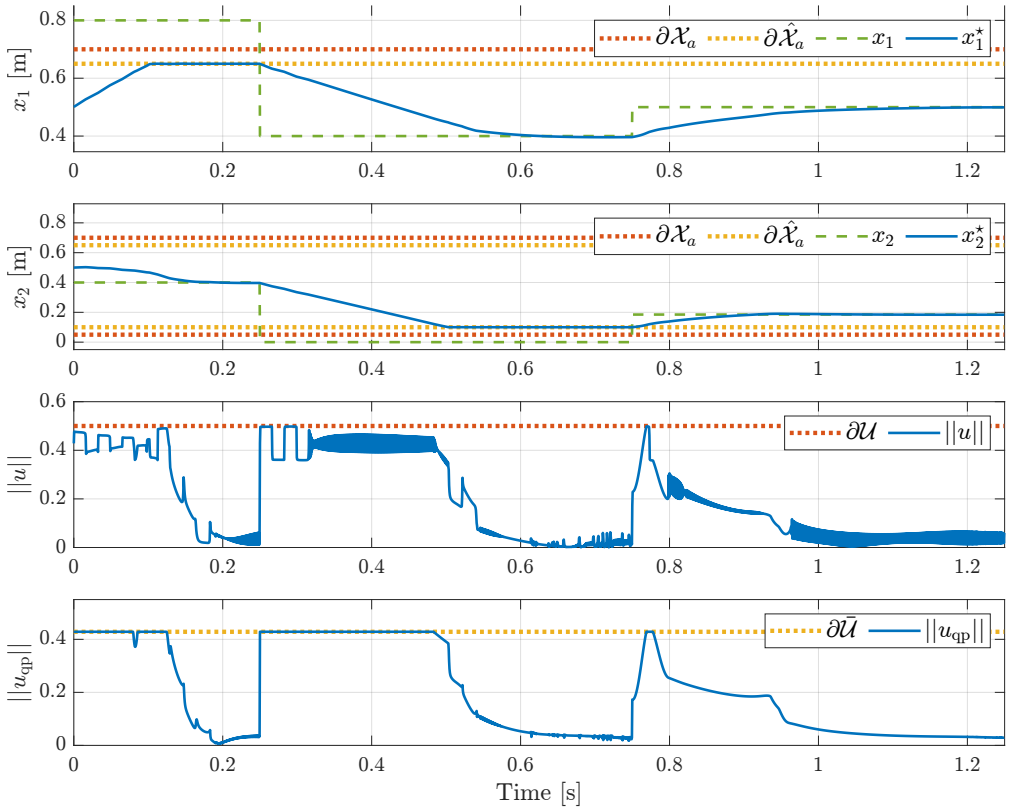


Figure 6.9: Time evolution of the system states (first and second plots), norm of the full control input (third plot), and norm of the nominal control input resulting from the QP problem (6.58)(fourth plot).

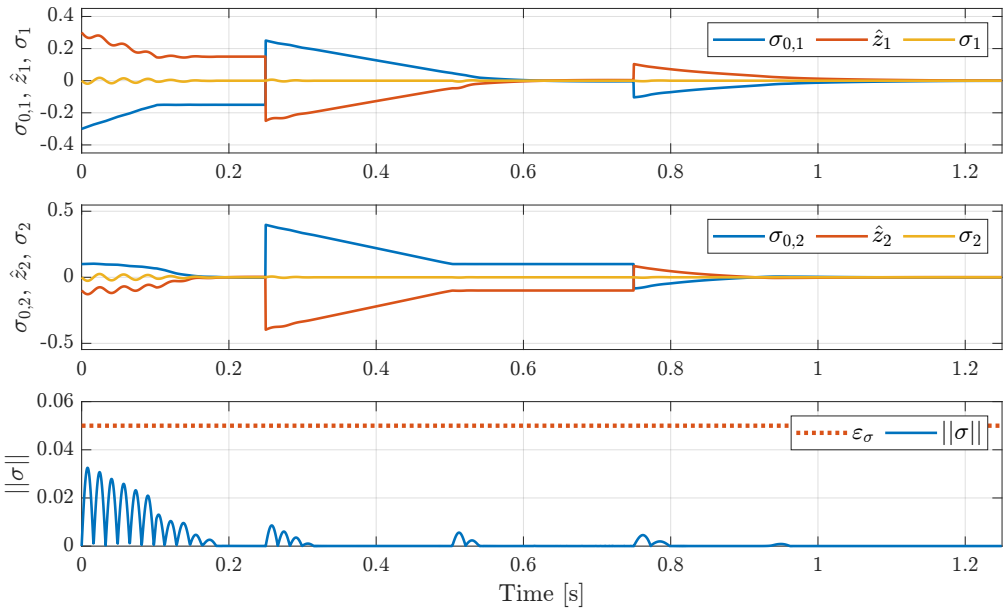


Figure 6.10: Time evolution of the sliding variable components (first and second plots) and norm of the sliding variable vector, along with the bound imposed by the BLF (third plot).

Part III

Fault Diagnosis via Neural Networks and Sliding Mode Observers

Chapter 7

Fault Diagnosis via DNN-ISM based UIO

Fault Diagnosis (FD) in an essential operation that aims to reduce the risk of damages of systems affected by faults. In particular, FD determines the causes of deviation of the control status from the desired behavior, and interprets such status given the measurements from sensors, or on the basis of the process model [118]. The FD is divided in three different procedures. Specifically, the *detection* procedure allows to understand when a fault occurs in the system, without any knowledge on the specific faulty component, the *isolation* allows to understand which one is the faulty component of the system, while the *identification* task reconstructs the fault signals. In general, FD methodologies can be divided in *passive* and *active*. To the former category, in which, independently of any fault information, an input signal is fed into the actual process and also into its nominal model, and then the corresponding output signals are analyzed, belong the techniques presented in works like [119] and [120]. While, to the latter, which exploit the injection of specific signals in order to improve the detectability of faults, belong approaches like [121, 122, 123], among many others. Other methodologies rely on robust control approaches, like $\mathcal{H}_\infty$ [124], multi-model approaches [125], LQR [126], and MPC [127], to cite a few.

During the years, SMC based techniques have been developed and adopted in several works (see, e.g., [128] , [129], and [130]) in different fault scenarios. With the objective of increasing the robustness of SMC, having also the possibility to define a specific dynamics of the observation error, one could rely on an ISM based Unknown Input Observer (UIO). However, as detailed in Section 2.4, ISM requires the knowledge of the system dynamics.

In this chapter, the FD scheme proposed in [131] which relies on a DNN-ISM based UIO, is presented. In particular, such a scheme is able to perform diagnosis on the control input even when the nominal model of the system is partially unavailable, estimating online the unknown part of the dynamics.

7.1 The considered faulted system

Considered a nonlinear control affine system affected by actuator fault, expressed in the canonical reduced form

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} f_1(x(t), t) \\ f_2(x(t), t) + \bar{B}(x(t), t)\Big(u(t) + \Delta u(x(t), t)\Big) \end{bmatrix}, \tag{7.1}$$

where $x \in \mathcal{X}$ is the system state vector, $f_1 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n-m}$ and $f_2 : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ are the components of the drift term, $\bar{B} : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{m \times m}$ is the control effectiveness matrix. The last three elements are bounded as in Assumption 5.1 and 5.2. As for $u \in \mathbb{R}^m$ and $\Delta u : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$, they represent, respectively, the control input, and the actuator fault.

Note that, the actuator fault is modeled as an additive disturbance entering the system through the input channel and it satisfies the following assumption.

Assumption 7.1. *There exists a known constant $\bar{\delta} \in \mathbb{R}_{>0}$ such that*

$$\sup_{x \in \mathcal{X}, t \in \mathbb{R}_{\geq 0}} \|\Delta u(x(t), t)\| \leq \bar{\delta}.$$

Before introducing the concept of ISM UIO, it is worth noticing that, in the rest of the chapter $u(t) = \kappa(x(t))$, with $\kappa : \mathcal{X} \to \mathbb{R}^m$ being any suitable control law for the system without faults, i.e., $\Delta u(t) = 0_m$.

7.2 ISM Unknown Input Observer

Assume that the dynamics components of system (7.1), i.e., f_1, f_2, and $\bar{B}$ are available. Then, since the objective is to perform fault diagnosis in presence of actuator fault, one can design a UIO of the form

$$\dot{\hat{x}}(t) = \begin{bmatrix} \dot{\hat{x}}_1(t) \\ \dot{\hat{x}}_2(t) \end{bmatrix} = \begin{bmatrix} f_1(\hat{x}(t), t) \\ f_2(\hat{x}(t), t) + \bar{B}(\hat{x}(t), t)\Big(u(t) + v(t)\Big) \end{bmatrix}, \tag{7.2}$$

with $\hat{x}(t_0) = x(t_0)$ and where $v \in \mathbb{R}^m$ is the input of the observer, designed according the ISM strategy as

$$v(t) = v_n(t) + v_r(t), \tag{7.3}$$

where $v_n \in \mathbb{R}^m$ is chosen to stabilize the observer error $e(t) = x(t) - \hat{x}(t) \in \mathbb{R}^n$ in the case $\Delta u(x(t), t) = 0_m$. As for $v_r \in \mathbb{R}^m$, it is chosen as

$$v_r = \rho \frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \tag{7.4}$$

where $\sigma : \mathcal{X} \to \mathbb{R}^m$ is the integral sliding variable, defined as in (2.69), i.e., $\sigma(x(t)) = \sigma_0(x(t)) - z(x(t))$. The conventional sliding variable is defined as the linear combination of the observation error, having

$$\sigma_0(x(t)) = C_1\Big(x_1(t) - \hat{x}_1(t)\Big) + C_2\Big(x_2(t) - \hat{x}_2(t)\Big), \tag{7.5}$$

with $C_1 \in \mathbb{R}^{m\times(n-m)}$ and $C_2 \in \mathbb{R}^{m\times m}$ being design matrices. In particular, C_2 must be chosen so that it satisfies Assumption 5.4.

As for the transient variable, it is chosen so that $z(x(t_0)) = \sigma_0(x(t_0))$, ensuring $\sigma(x(t_0)) = 0_m$. Specifically, it is characterized by dynamics

$$\dot{z} = C_1\Big\{f_1(x) - f_1(\hat{x})\Big\} + C_2\Big\{f_2(x) - f_2(\hat{x}) + \Big(\bar{B}(x) - \bar{B}(\hat{x})\Big)u - \bar{B}(\hat{x})v_n\Big\}, \tag{7.6}$$

where dependence on time is omitted for sake of readability.

As detailed in the following, properly selecting the discontinuous control gain ρ in (7.4), one can enforce a sliding mode $\sigma(x(t)) = 0_m$ for $t \geq t_0$ and exploit the concept of equivalent control to provide an estimate $\Delta u(x(t), t)$.

Theorem 7.1. *Consider the faulty system* (7.1) *and the UIO in* (7.2)*, characterized by input* v *in* (7.3)*. If Assumption 7.1 holds, and* ρ *in* (7.4) *is chosen so such that*

$$\rho > \frac{\|C_2\|\,\bar{\gamma}}{\underline{\lambda}(C_2)\underline{\gamma}}\bar{\delta},$$

then a sliding mode $\sigma(x(t)) = 0_m$ *is enforced for* $t \geq t_0$*. Moreover, letting* v_{eq} *be the equivalent control signal, it holds that*

$$v_{\text{eq}}(t) = (B(\hat{x}))^{+}B(x)\Delta u(t).$$

Proof. See Appendix B.7. □

As detailed in [36], v_{eq} is often obtained using a first-older filter with in input the discontinuous signal v_r. In particular, $v_{\text{eq}}(t) \approx \hat{v}_{\text{eq}}(t)$, with this last one coming from

$$\mu\dot{\hat{v}}_{\text{eq}}(t) = v_r(t) - \hat{v}_{\text{eq}}(t), \tag{7.7}$$

with $\hat{v}_{\text{eq}}(t_0) = 0_m$ and where $\mu \in (0, 1)$ is a filtering constant.

Note that, the results presented in this section are valid only in the case in which the model of the system (7.1) is completely available. In the following, the dynamics of (7.1) is considered partially unknown.

7.3 DNN-ISM Unknown Input Observer

Inspired by the DNN-ISM control strategy presented in Chapter 5, it is possible to design an UIO when part of the dynamics of (7.1) is not available and design. In particular, the drift dynamics components f_1 and f_2 are considered unknown, while the control effectiveness matrix $\bar{B}$ is available. Hence, coherently with the DNN-ISM framework, f_1 and f_2 are estimated by a DNN Φ as in (5.14).

With these premises, consider the UIO defined as

$$\dot{\hat{x}}(t) = \begin{bmatrix} 0_{n-m} \\ \bar{B}(\hat{x}(t), t)\Big(u(t) + v(t)\Big) \end{bmatrix}, \tag{7.8}$$

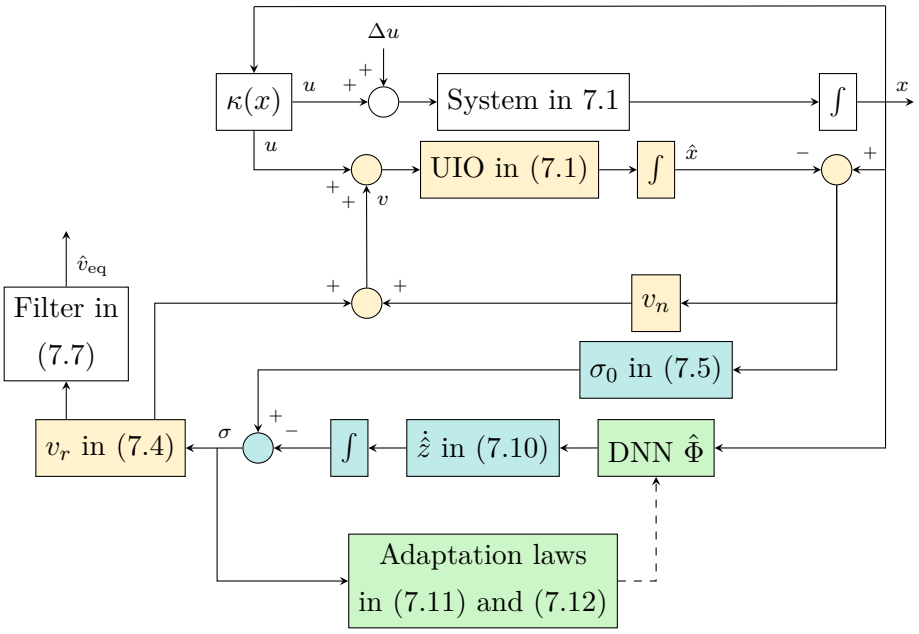


Figure 7.1: Block diagram of the DNN-ISM based UIO. The blocks associated with the sliding variable are colored in blue, the ones related to the DNNs in green, while the ones in yellow are related to the UIO and its input.

where v is the one in (7.3), with modified integral sliding variable

$$\sigma(x(t)) = \sigma_0(x(t)) - \hat{z}(x(t)). \tag{7.9}$$

In this case, the transient variable $\hat{z}$ is characterized by dynamics dependent on the estimate provided by the DNN, having

$$\dot{\hat{z}} = C_1 \hat{\Phi}^{[1]}_{k_\Phi} + C_2 \Big\{ \hat{\Phi}^{[2]}_{k_\Phi} + \Big(\bar{B}(x) - \bar{B}(\hat{x}) \Big) u - \bar{B}(\hat{x}) v_n \Big\} \tag{7.10}$$

Before going on with the analysis, it is worth noticing that the input of the DNN is state of (7.1), and not the one of the observer, i.e., $\hat{\Phi}_{k_\Phi} = \hat{\Phi}(x)$.

Analogously to what was previously done for the DNN-ISM control strategy, in this case the weights of the DNN $\hat{\Phi}$ are adapted according to law derived from Lyapunov stability analysis. In particular, for layers $j \in \{0, 1, \dots, k_\Phi - 1\}$, the weights are adjusted according to

$$\operatorname{vec}\left(\dot{\hat{V}}_j\right) = \underset{\mathcal{B}_{\Phi_j}}{\operatorname{proj}} \left(\Gamma_{\Phi_j} \Lambda^\top_{\Phi_j} C^\top \sigma \right) \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}}}, \tag{7.11}$$

where $\Lambda_{\Phi_j} \in \mathbb{R}^{n \times L_{\Phi_j} L_{\Phi_{j+1}}}$ is the one defined in (5.46). $\Gamma_{\Phi_j} \in \mathbb{R}^{L_{\Phi_j} L_{\Phi_{j+1}} \times L_{\Phi_j} L_{\Phi_{j+1}}}$ is the adaptation rate matrix, defined as diagonal with positive entries. The weights sub-matrices associated with the last layer $j = k_\Phi$ are adjusted according to

$$\operatorname{vec}\left(\dot{\hat{V}}^{[1]}_{k_\Phi}\right) = \underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\operatorname{proj}} \left(\Gamma^{[1]}_{\Phi_{k_\Phi}} (\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top C_1^\top \sigma \right) \in \mathbb{R}^{L_{k_\Phi}(n-m)}, \tag{7.12a}$$

$$\operatorname{vec}\left(\dot{\hat{V}}_{k_\Phi}^{[2]}\right) = \underset{\mathcal{B}_{\Phi_{k_\Phi}}}{\operatorname{proj}}\left(\Gamma_{\Phi_{k_\Phi}}^{[2]}(\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_2^\top \sigma\right) \in \mathbb{R}^{L_{k_\Phi} m}, \tag{7.12b}$$

where $\Gamma_{\Phi_{k_\Phi}}^{[1]} \in \mathbb{R}^{(n-m)L_{k_\Phi}\times(n-m)L_{k_\Phi}}$ and $\Gamma_{\Phi_{k_\Phi}}^{[2]} \in \mathbb{R}^{mL_{k_\Phi}\times mL_{k_\Phi}}$ are diagonal matrices with positive entries, while $\Lambda_{\Phi_{k_\Phi}}^{[1]} \in \mathbb{R}^{(n-m)\times(n-m)L_{k_\Phi}}$ and $\Lambda_{\Phi_{k_\Phi}}^{[2]} \in \mathbb{R}^{m\times mL_{k_\Phi}}$ are the ones defined below equation (5.28).

Theorem 7.2. *Consider the system* (7.1)*, the UIO* (7.8)*, with input* (7.3) *and integral sliding variable* (7.9)*. Then, let the weights of the DNN be updated according to* (7.11)*,* (7.12a)*, and* (7.12b)*. If Assumptions 5.1, 5.2, 5.6, and 7.1 hold, and the discontinuous control gain is chosen as*

$$\rho > \frac{\|C_1\|\{\bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1}\} + \|C_2\|\{\bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + \bar{\delta}\} + \bar{\eta}}{\underline{\lambda}(C_2)\underline{\gamma}},$$

with $\bar{\eta} \in \mathbb{R}_{>0}$, then $\sigma(x(t)) \to 0_m$ for $t \to \infty$.

Proof. See Appendix B.8 □

The above theorem only guarantees that a sliding mode $\sigma(x(t)) = 0_m$ is achieved asymptotically. However, such a condition is required to exploit the equivalent control and provide an estimate of the fault acting on the system. However, similar considerations done for the DNN-ISM in Chapter 5 can be made. In particular, from Theorem 7.2 follows that a practical sliding mode condition $\|\sigma(x(t))\| \leq \varsigma$ is satisfied for $t \geq t_1 > t_0$. Moreover, from the same theorem it is possible to compute the discontinuous control gain which would enforce an ideal sliding mode $\sigma(x(t)) = 0_m$, independently from the initialization of the weights.

Proposition 7.1. *Following the same reasoning reported in the proof of Theorem 7.2, and relying on the bounds of the system and the ideal weights for bounding the approximation error, it holds that if*

$$\bar{\rho} = \frac{\|C_1\|\left(\bar{V}^{k_\Phi} + \bar{f}_1\right) + \|C_2\|\left(\bar{V}^{k_\Phi} + \bar{f}_2 + \bar{\delta}\right)}{\underline{\lambda}(C_2)\underline{\gamma}} + \frac{\bar{\eta}}{\underline{\lambda}(C_2)\underline{\gamma}}, \tag{7.13}$$

with $\bar{\eta} \in \mathbb{R}_{>0}$, then $\sigma(x(t)) = 0_m$ for $t \geq t_0$.

The above results legitimate the use of the strategy presented below Proposition 5.4 to enforce an ideal sliding mode $\sigma(x(t)) = 0_m$ in finite time. In particular, let $t_1 > t_0$ be the time instant in which the practical sliding mode condition $\|\sigma(x(t))\| \leq \varsigma$ is satisfied, then, the adaptation of the weights of the DNN is interrupted and the discontinuous control gain is adjusted according to the law presented in the following theorem.

Theorem 7.3. *Consider the system* (7.1)*, the UIO* (7.8)*, with input* (7.3) *and integral sliding variable* (7.9)*. For $t \geq t_1$, with t_1 being the one defined in Proposition 5.4, let the input of UIO* (7.8) *be the one in* (7.3)*, where v_r is designed as*

$$v_r(t) = \hat{\rho}(t)\frac{\sigma(x(t))}{\|\sigma(x(t))\|},$$

with $\hat{\rho}(t_1)$ and $\hat{\rho}_2(t_1)$ chosen so that it satisfies Theorem 7.2. Moreover, let $\operatorname{vec}\left(\dot{V}_j\right) = 0$ *for $j \in \{0, 1, \dots, k_\Phi\}$. If Assumptions 5.1, 5.2, - 5.6, and 7.1 hold and the discontinuous control gain is adapted according to*

$$\dot{\hat{\rho}}(t) = \operatorname*{proj}_{\varrho}\left(\left(\bar{\lambda}(C_2)\bar{\gamma} + \alpha\right)\|\sigma\|\right), \tag{7.14a}$$

with $\alpha \in \mathbb{R}_{>0}$ being a constant acting as a learning rate, $\bar{\gamma}$ being the one appearing in Assumption 5.2 and $\varrho \coloneqq \{r \in \mathbb{R}_{>0} : r \leq \bar{\rho}\}$, then a sliding mode $\sigma(x(t)) = 0_m$ is enforced for $t \geq t_2$, with $t_1 < t_2 < \infty$.

Moreover, the equivalent control signal is

$$\begin{aligned} v_{\text{eq}}(t) = &\left(C_2\bar{B}(\hat{x}(t), t)\right)^{-1}\left\{C_1\left[\tilde{\Phi}^{[1]}_{k_\Phi, t_1} + \varepsilon^{[1]}_\Phi(x(t))\right] + C_2\left[\tilde{\Phi}^{[2]}_{k_\Phi, t_1} + \varepsilon^{[2]}_\Phi(x(t))\right]\right\} + \\ &+ \left(C_2\bar{B}(\hat{x}(t), t)\right)^{-1} C_2\bar{B}\Delta u(t), \end{aligned} \tag{7.15}$$

where $\tilde{\Phi}^{[1]}_{k_\Phi, t_1}$ and $\tilde{\Phi}^{[2]}_{k_\Phi, t_1}$ denote the approximation error between the ideal DNN and the one characterized by weights $\hat{V}_j(t_1)$, for $j \in \{0, 1, \dots, k_\Phi\}$.

Proof. See Appendix B.9. □

From (7.15) it is possible to conclude that, when using the DNN to approximate the unknown drift term of the system, the quality of the estimation of the fault is reduced. In particular, the better is the approximation of the weights at time t_1, the more the equivalent control signals follows the fault.

Recall that, once again, a continuous approximation for v_{eq} is possible via the filter defined in (7.7).

7.4 Simulations

The fault diagnosis scheme depicted in Figure 7.1 is assessed in simulation relying on the Duffing Oscillator model in (2.3), assumed partially unknown. In particular, the control effectiveness $\bar{B}$ is assumed available, while the drift term f_2 is assumed unknown and estimated with the DNN described in Section 5.5.1.

The objective of the simulation, whose duration is 20 seconds, is to provide an estimate of un unknown fault Δu acting on the system, which is controlled to reach a desired state $x^\star = \begin{bmatrix} 1.5 & 0 \end{bmatrix}^\top$, defining u as simple PI controller. The fault is defined as $\Delta u(t) = 0.1\sin(2\pi t) + 0.05\cos(\pi t)$ for $t \in [0, 15)$ seconds, and $\Delta u(t) = 0.1\sin(2\pi t)$ for $t \geq 15$ seconds. The conventional sliding variable has been chosen as the linear combination of the observation error, with coefficients equal to one. As for the UIO input, it is defined by a nominal law defined as

$$v_n(t) = \begin{bmatrix} 2 & 2 \end{bmatrix}(x(t) - \hat{x}(t)) + \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}\int_0^t (x(\tau) - \hat{x}(\tau))d\tau,$$

while the discontinuous control gain is chosen as $\rho = 0.175$. The equivalent control is approximated by filtering the discontinuous part of the observer input v_r by means of the low pass filter $\dot{\hat{v}}_{\text{eq}}(t) = 100(v_r(t) - \hat{v}_{\text{eq}}(t))$, with $\hat{v}_{\text{eq}}(0) = 0$.

The results of the simulation are depicted in Figure 7.2, which illustrates the time evolution of the system and observer states, the components of the sliding variable, and the approximated equivalent control compared with the actuator fault Δu acting on the system. In particular, it is possible to see how, once the practical sliding mode is enforced, $\hat{v}_{\text{eq}}$ satisfactorily estimates the fault Δu, and the UIO states $\hat{x}$ practically converge to the real states of the system.

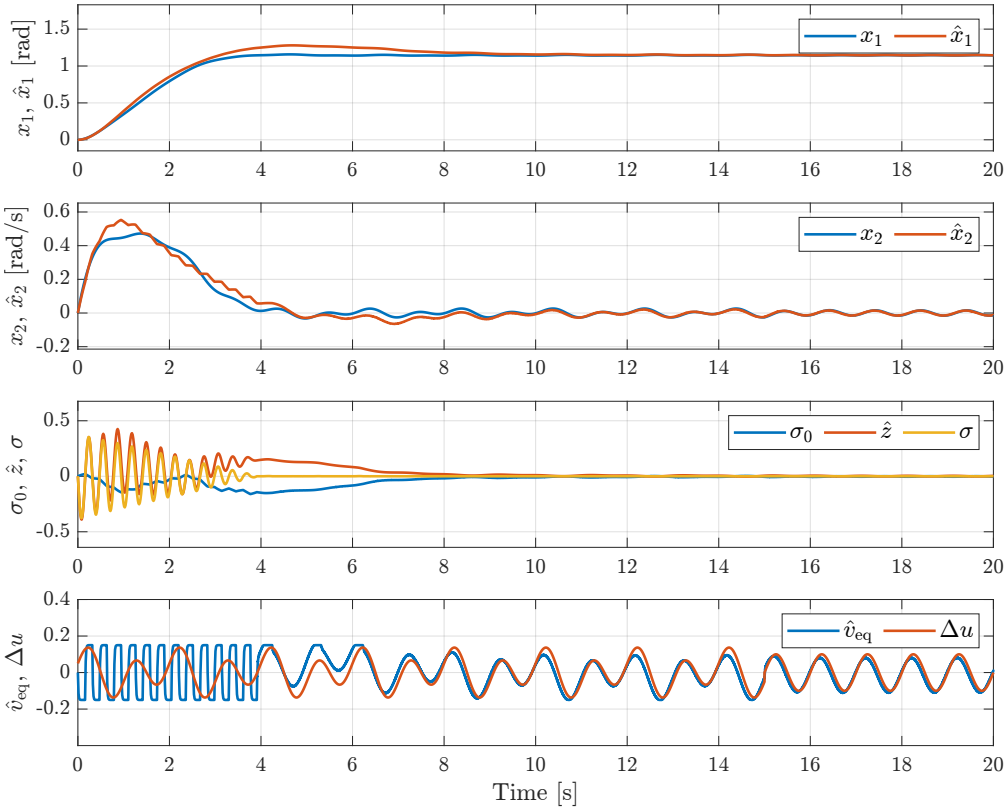


Figure 7.2: Time evolution of the system and UIO states, the sliding variable components, the actual fault injected into the system, and its estimate.

Chapter 8

SM based Fault Diagnosis with DRL add-ons for Redundant Manipulators

In general, robots may be susceptible to different types of faults which can occur simultaneously on both sensors and actuators, perturbing the robot operations [132]. the model of a robot manipulator is nonlinear and presents strong coupling of the states, making the effect of the different faults coupled, hence the application of standards FD techniques (e.g., [133, 134, 135]), which neglect coupling effects, is not suitable in many cases.

In order to decouple the effects of fault signals acting on sensors and actuators, it is possible to add external vision sensors, like proposed in [136]. In particular, such a paper introduced a vision sensor to enhance the FD capability of the proposed scheme SMC based observers. Such a work has been extended in [137], in which a nonlinear coupled planar manipulator is transformed into a set of decoupled linear systems using the so-called inverse dynamics approach. Then, sensors fault diagnosis is done relying on a low-cost IP camera, which extrapolates the correct position of the end-effector, while actuator fault diagnosis is done via a battery of Suboptimal Second Order Sliding Mode (SSOSM) [33] UIOs.

The aim of this chapter is to present the FD scheme introduced in [138], in which a DRL agent and a battery of Second Order Sliding Mode (SOSM) [44] UIOs are employed to perform sensor and actuator FD, respectively. Differently from previous works, the manipulator considered in [138] is not planar and it is characterized by kinematic redundancy. In particular, the output of the model-free DRL agent is employed to correct the corrupted sensor measurements before they are used by the UIOs.

8.1 Problem Formulation

Consider a redundant robot manipulator, i.e., one that is characterized by $n > p$ revolute joints, with $p \in \mathbb{N}_{>0}$ being the dimension of the operational space. The kinematic and dynamic model of the manipulator are derived according to the methodology presented in Chapter 4. For convenience, the main elements of the modeling are reported in the following.

8.1.1 Robot Model

Coherently with the notation introduced in Chapter 4, the vector of the joint variables is denoted as $q \in \mathbb{R}^n$. Then, defining a base reference frame $O_b - x_b y_b z_b$ fixed in space and a reference frame $O_e - x_e y_e z_e$ attached to the end-effector, it is possible to express the forward kinematics of the manipulator by means of the homogeneous transformation matrix

$$T_e^b(q) = \begin{bmatrix} R_e^b(q) & p_e^b(q) \\ 0_3^\top & 1 \end{bmatrix}, \tag{8.1}$$

with $p_e^b \in \mathbb{R}^3$ and $R_e^b \in SO(3)$ being the position and orientation of O_e with respect to O_b, respectively.

As for the dynamics of the manipulator, it is described by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\text{sign}(\dot{q}) + g(q) = \tau, \tag{8.2}$$

where $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C(q,\dot{q})\dot{q} \in \mathbb{R}^n$ is the vector modeling the Coriolis and centrifugal forces, $F_v\dot{q} \in \mathbb{R}^n$ and $F_s\text{sign}(\dot{q}) \in \mathbb{R}^n$ represent viscous and Coulomb friction, respectively, $g(q) \in \mathbb{R}^n$ is the vector of gravitational torques, while $\tau \in \mathbb{R}^n$ is the input torque. Moreover, if one defines the quantity

$$\nu(q,\dot{q}) = C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\text{sign}(\dot{q}) + g(q), \tag{8.3}$$

the model in (8.2) can be expressed in a more compact form as

$$M(q)\ddot{q} + \nu(q,\dot{q}) = \tau. \tag{8.4}$$

The following assumption about the robot model holds.

Assumption 8.1. *The inertia matrix $M \in \mathbb{R}^{n\times n}$ and the vector $\nu \in \mathbb{R}^n$ are known.*

8.1.2 Faults Modeling

Let $\delta_q : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and $\delta_\tau : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ be the time-varying vectors containing sensor and actuator faults, respectively. Then, it is possible to define the quantities

$$\tilde{q}(t) \coloneqq q(t) + \delta_q(t), \tag{8.5a}$$

$$\tilde{\tau}(t) \coloneqq \tau(t) + \delta_\tau(t), \tag{8.5b}$$

that describe, respectively, the joint positions measured by the faulted sensors, and the faulted input that will be applied to the robot.

Moreover, the following assumption about the fault vectors holds.

Assumption 8.2. *There exist some known constants* $\bar{d}_q, \bar{d}_\tau \in \mathbb{R}_{>0}$ *such that*

$$\sup_{t\in\mathbb{R}_{\geq 0}} \|\delta_q(t)\| \leq \bar{d}_q, \qquad \sup_{t\in\mathbb{R}_{\geq 0}} \|\delta_\tau(t)\| \leq \bar{d}_\tau.$$

8.1.3 Problem Statement

Consider any task in the operational space that can be described by $p < n$ variables. Assume that all the joint variables are measurable from the encoders embedded in the robot joints, and these are susceptible to faults, while external sensors (e.g., a vision sensors) are capable to directly retrieve only the end-effector position p_e^b without being affected by any possible fault. Furthermore, assume that, during the task execution, any Fault Event (FE) from the ones in Table 8.1 can occur.

FE #	Description
1	Single fault on an actuator
2	Single fault on a sensor
3	Multiple faults on actuators
4	Multiple faults on sensors
5	Multiple mixed faults

Table 8.1: Possible fault events (FEs) during the task execution.

The goal is to design a FD scheme capable of providing, at each time instant t, the estimates $\hat{\delta}_q(t) \in \mathbb{R}^n$ and $\hat{\delta}_\tau(t) \in \mathbb{R}^n$ of the sensor and actuator faults with the objective of minimizing the performance index

$$\mathcal{I}(t) \coloneqq \mathcal{I}_s(t) + \mathcal{I}_a(t), \tag{8.6}$$

with the terms $\mathcal{I}_s \in \mathbb{R}_{\geq 0}$ and $\mathcal{I}_a \in \mathbb{R}_{\geq 0}$ being defined as

$$\mathcal{I}_s(t) \coloneqq \left\| \hat{\delta}_q(t) - \delta_q(t) \right\|, \tag{8.7}$$

$$\mathcal{I}_a(t) \coloneqq \left\| \hat{\delta}_\tau(t) - \delta_\tau(t) \right\|. \tag{8.8}$$

8.2 Inverse Dynamics control

In order to design the fault diagnosis scheme, the dynamics of the controlled robots must be decoupled so that it is in the form of a perturbed double integrator. To accomplish that, the so-called *inverse dynamics control* [24], whose architecture is depicted in Figure 8.1, can be applied.

Assume, for now, that no fault is affecting neither the sensors or the actuators of the robot, i.e., $\delta_q(t) = 0_n$ and $\delta_\tau(t) = 0_n$. Then, the inverse dynamics of a robot manipulator

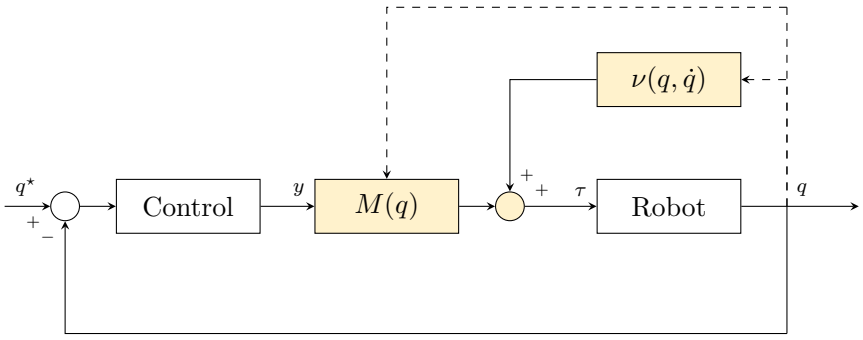


Figure 8.1: Inverse dynamic control architecture.

with dynamics in (8.4), can be expressed as a nonlinear relationship between inputs and outputs. In particular, let $y \in \mathbb{R}^n$ be an auxiliary control vector that describes some desired behavior to the robot in terms of joint accelerations. Then, designing the control torque as

$$\tau = M(q)y + \nu(q, \dot{q}) \tag{8.9}$$

and substituting it into (8.4), the dynamics of the controlled robot becomes

$$\ddot{q}(t) = y(t), \tag{8.10}$$

that is a chain of n decoupled double integrators, one for each joint of the robot.

Consider now $\delta_q(t) \neq 0_n$ and $\delta_q(t) \neq 0_n$. Then, the control law (8.9) becomes

$$\tau = M(\tilde{q})y + \nu(\tilde{q}, \dot{\tilde{q}}) + \delta_\tau. \tag{8.11}$$

Substituting it into (8.4), one has that

$$\begin{aligned}\ddot{q} &= M^{-1}(q)\Big(\tau - \nu(q,\dot{q})\Big) \\ &= M^{-1}(q)M(\tilde{q})y + M^{-1}(q)\nu(\tilde{q},\dot{\tilde{q}}) + M^{-1}(q)\delta_\tau - M^{-1}(q)\nu(q,\dot{q}). \\ &= G(q,\tilde{q})y + \tilde{\nu}(q,\dot{q},\tilde{q},\dot{\tilde{q}}) + \delta_y \end{aligned} \tag{8.12}$$

where $G(q,\tilde{q}) = M^{-1}(q)M(\tilde{q}) \in \mathbb{R}^{n\times n}$, $\tilde{\nu}(q,\dot{q},\tilde{q},\dot{\tilde{q}}) = M^{-1}(q)\Big(\nu(\tilde{q},\dot{\tilde{q}}) - \nu(q,\dot{q})\Big) \in \mathbb{R}^n$, while $\delta_y \in \mathbb{R}^n$ denotes the acceleration fault induced by the actuator fault δ_τ and it is defined as

$$\delta_y(q,t) = M^{-1}(q)\delta_\tau(t). \tag{8.13}$$

The following assumption about $G(q,\tilde{q}) = M^{-1}(q)M(\tilde{q})$, $\tilde{\nu}(q,\dot{q},\tilde{q},\dot{\tilde{q}})$, and δ_y needs to be introduced.

Assumption 8.3. *There exist some known constants $\tilde{g}, \bar{\tilde{\nu}} \in \mathbb{R}_{\geq 0}$ such that the quantities $G(q,\tilde{q})$ and $\tilde{\nu}(q,\dot{q},\tilde{q},\dot{\tilde{q}})$ are bounded as*

$$\sup_{q,\tilde{q}\in\mathbb{R}^n} \|G(q,\tilde{q})\| \leq \tilde{g}, \qquad \sup_{q,\dot{q},\tilde{q},\dot{\tilde{q}}\in\mathbb{R}^n} \left\|\tilde{\nu}(q,\dot{q},\tilde{q},\dot{\tilde{q}})\right\| \leq \bar{\tilde{\nu}}.$$

Assumption 8.4. *There exists a known constant* $\bar{d}_y \in \mathbb{R}_{>0}$ *such that the acceleration fault* δ_y *is bounded as*

$$\sup_{q \in \mathbb{R}^n, t \in \mathbb{R}_{\geq 0}} \|\delta_y(q,t)\| \leq \bar{d}_y.$$

Note that, even though the control of the faulted robot is beyond the scope of this chapter, Assumptions 8.3 and 8.4 are instrumental for designing a controller capable of guaranteeing the closed-loop stability in presence of model mismatches.

8.3 The Fault Diagnosis Scheme

In order to solve the problem introduced in Section 8.1.3, a fault diagnosis scheme which relies on SOSM UIOs and a DRL agent, depicted in Figure 8.2, has been introduced in [138].

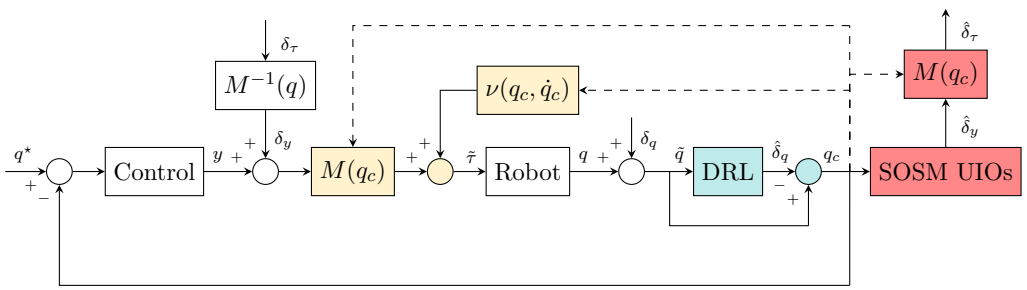


Figure 8.2: Block diagram of the FD scheme. The blocks in yellow are related to the inverse dynamics, the ones in blue are associated with the estimation and compensation of the sensor faults, while the blocks responsible for the actuator faults estimation are colored in red.

8.3.1 Sensor fault diagnosis with DRL

To perform diagnosis of sensor faults concurrently occurring on multiple joints, a model-free DRL agent, trained with the TD3 algorithm described in Algorithm 3, has been designed. Although the application of the proposed architecture to an intrinsically redundant robot might increase the computational complexity for the DRL, it has the advantage to overcome possible issues due to the kinematic inversion which could lead to multiple (possibly infinite) solutions.

Before describing the DRL agent, the following assumption must be introduced.

Assumption 8.5. *The vector of the end-effector position* $p_e^b \in \mathbb{R}^3$ *is available for measurement for all* $t \in \mathbb{R}_{\geq 0}$.

Note that, such an assumption is reasonable since, in the practice, p_e^b can be retrieved using, vision sensors, e.g., multiple stereo cameras or an Red Green Blue Depth (RGBD) camera.

Making reference to the notation introduced in Section 3.2, the environment is represented by the state space $\mathcal{S}$ which depends on the end-effector position vector $p_e^b \in \mathbb{R}^3$, assumed known in Assumption 8.5, and the joint variable vector affected by faults $\tilde{q} \in \mathbb{R}^n$, defined as in (8.5a). In particular, $\mathcal{S}$ is given by

$$\mathcal{S} \coloneqq \{p_b^e, \tilde{q}\} \subset \mathbb{R}^{n+3}. \tag{8.14}$$

Since the the agent is to provide an estimate of the sensor fault, the bounded action space $\mathcal{A}$ is defined as

$$\mathcal{A} \coloneqq \left\{\hat{\delta}_{q,1}, \hat{\delta}_{q,2}, \ldots, \hat{\delta}_{q,n}\right\} \subset \mathbb{R}^n, \tag{8.15}$$

with $\hat{\delta}_{q,i} \in \mathbb{R}$ being the estimate of the fault on the i-th sensor. Moreover, each component of the action is bounded as

$$\underline{a}_i \leq \hat{\delta}_{q,i} \leq \bar{a}_i, \tag{8.16}$$

for $i \in \{1, 2, \ldots, n\}$, with $\underline{a}_i, \bar{a}_i \in \mathbb{R}_{>0}$ being design parameters depending on $\bar{d}_q$ introduced in Assumption 8.2.

Since the action space in (8.15) is defined so that the output is limited as in (8.16), and the sensor fault vector δ_q is bounded as in Assumption 8.2, the following holds.

Proposition 8.1. *There exists a constant $E \in \mathbb{R}_{\geq 0}$ such that the sensor fault estimation error $\hat{\delta}_q - \delta_q$ is bounded as*

$$\sup_{t \in \mathbb{R}_{\geq 0}} \left\| \hat{\delta}_q(t) - \delta_q(t) \right\| \leq E.$$

The DRL agent has been trained with the TD3 algorithm, relying on a realistic virtualization of the robot (see Appendix C.3). In particular, during the training phase different sensor fault signals δ_q are injected into the system, and the reward function, used to assign a value to each action, is designed as

$$r_t = -\mathcal{I}_s(t), \tag{8.17}$$

where $\mathcal{I}_s$ is defined as in (8.7).

During the training phase, the robot is controlled to follow predefined trajectories in the joint space and at each time step, data from the simulated sensors is retrieved and constant faults $\delta_{q,i}$, with, $i = \{1, 2, \ldots n\}$, each characterized by random amplitude, are added. The result, along with the end-effector position, is given as input to the DRL agent, which is trained as described in Algorithm 3, using (8.17) as an instantaneous reward.

Once the training phase is completed, the weights of the DNNs of the DRL are kept constant for the online test phase, in which the robot is controlled to follow a specific trajectory, while being subject to faults. Note that, since the control y does not belong to the state space $\mathcal{S}$, it does not affect the sensor fault estimate $\hat{\delta}_q$.

Such an estimate is then employed to perform both detection and isolation by building a vector of flags $f_s \in \{0,1\}^n$, whose elements are defined as

$$f_{s,i} = \begin{cases} 0, & \text{if } \underline{\Delta}_{s,i} \leq \hat{\delta}_{q,i} \leq \bar{\Delta}_{s,i} \\ 1, & \text{otherwise} \end{cases} \tag{8.18}$$

where $\underline{\Delta}_{s,i} \in \mathbb{R}$ and $\bar{\Delta}_{s,i} \in \mathbb{R}$, with $\underline{\Delta}_{s,i} < \bar{\Delta}_{s,i}$, are design constants indicating the i-th lower and upper thresholds, respectively. Note that, in an ideal scenario, $\underline{\Delta}_{s,i} = \bar{\Delta}_{s,i} = 0$, but possible estimation errors, caused also by imprecisions coming from the DRL agent, may occur.

Moreover, by analyzing the time evolution of $\hat{\delta}_q$, it is possible to compensate the fault and use a "corrected" version of the signals coming from the faulted sensors to build the control loop. Specifically, let $q_c \in \mathbb{R}^n$ be defined as

$$q_{c,i} = \begin{cases} \tilde{q}_i - \hat{\delta}_{q,i}, & \text{if } f_{s,i} = 1, \\ \tilde{q}_i, & \text{otherwise} \end{cases} \tag{8.19}$$

for $i \in \{1, 2, \ldots, n\}$, meaning that the signal coming from the sensor is corrected only if the a fault is detected on the same sensor.

Then, q_c is used to perform the Inverse dynamic control

$$\tau = M(q_c)y + \nu(q_v, \dot{q}_c) + \delta_\tau, \tag{8.20}$$

as shown in Figure 8.2.

8.3.2 Actuator FD with SOSM UIOs

Exploiting the nominal integrator chain model of the robot in (8.10) and employing the corrected joint variables measures q_c for the inverse dynamics control, it holds that

$$\ddot{q}_c(t) = y(t) + \delta_y(q, t) \tag{8.21}$$

with δ_y in (8.13). Hence, it is possible to design an battery of UIOs, one for each joint, to provide an estimate of actuators faults.

In particular, define the UIO model as

$$\ddot{\hat{q}}_c(t) = y(t) + v(t), \tag{8.22}$$

where $\hat{q}_c \in \mathbb{R}^n$ are the states of the observer and $v \in \mathbb{R}^n$ is the observer input.

Let $e_1(t) = q_c(t) - \hat{q}_c(t)$ be the observation error and $e_2(t) = \dot{e}_1(t)$ its first time-derivative. The objective is to design a sliding mode based observer that ensures convergence to zero of the estimation error within an optimal reaching time or, in the case in which the worst realization of the uncertain terms occurs, within a minimum time. Making reference to [44], a SOSM law to accomplish such an objective can be designed.

Define the sliding variable $\sigma_1 \in \mathbb{R}^n$ as the combination of the estimation error and its derivative, with each component defined as

$$\sigma_{1,i}(t) = \beta e_{1,i}(t) + e_{2,i}(t), \tag{8.23}$$

for $i \in \{1, 2, \ldots, n\}$, where $\beta \in \mathbb{R}_{>0}$ is a design constant. If one computes the first time-derivative of (8.23) relying on (8.22) its relative degree is equal to 1, so that a first order sliding mode law would apply.

Having as a goal the design of the SOSM law in [44], an auxiliary system, with relative degree 2, can be defined as

$$\begin{cases} \dot{\sigma}_{1,i}(t) = \sigma_{2,i}(t) \\ \dot{\sigma}_{2,i}(t) = f(e_{2,i}, q_{c,i}, \dot{y}_i) - w_i(t) \\ w_i(t) = \dot{v}_i(t) \end{cases} \tag{8.24}$$

where $w_i \in \mathbb{R}$ is the new observer input, while the drift term $f : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is defined as

$$f(e_{2,i}, q_{c,i}, \dot{y}_i) = \beta_i \dot{e}_{2,i}(t) + \frac{d^{(3)} q_{c,i}(t)}{dt^3} - \dot{y}_i.$$

By virtue of the mechanical nature of the robotic system, the following assumption holds.

Assumption 8.6. *There exists a known constant $F_i \in \mathbb{R}_{>0}$ such that the drift term $f(e_{2,i}, q_{c,i}, \dot{y}_i)$ is bounded as*

$$\sup_{e_{2,i}, q_{c,i}, \dot{y}_i \in \mathbb{R}} |f(e_{2,i}, q_{c,i}, \dot{y}_i)| \leq F_i$$

for $i \in \{1, 2, \ldots, n\}$.

Then, the auxiliary observer input w_i is designed as

$$w_i(t) = \alpha_i \text{sign}\left(\sigma_{1,i}(t) + \frac{\sigma_{2,i}(t)|\sigma_{2,i}(t)|}{2\alpha_{r,i}}\right), \tag{8.25}$$

with $\alpha_i > F_i$ being the control gain, and $\alpha_{r,i} \in \mathbb{R}_{>0}$ the so-called *reduced* control amplitude defined as

$$\alpha_{r,i} \coloneqq \alpha_i - F_i, \tag{8.26}$$

which represents the minimum control amplitude to cope with the worst realization of the drift uncertain term.

The following result can be introduced convergence of the sliding variables to zero.

Theorem 8.1. *Consider the robotic system described by* (8.4)*, controlled by the inverse dynamics law in* (8.20)*, which relies on the estimates provided by the DRL agent. If Assumption 8.6 holds and $f(e_{2,i}, q_{c,i}, \dot{y}_i) = -F_i sign(w_i)$, then all the components $\sigma_{1,i}(t)$ of the auxiliary system* (8.24)*, controlled by* (8.25)*, are steered to zero in minimum time.*

Proof. See Appendix B.10 □

Theorem 8.1 ensures the finite-time convergence to zero of the sliding variable and its first time-derivative, implying that the error signals $e_{1,i}(t)$ and $e_{2,i}(t)$ exponentially decay to zero constrained to $\sigma_{1,i}(t) = 0$ for $t \geq t_r$, with $t_r \geq t_0$ being the convergence time.

Once in sliding mode, it is possible to exploit equivalent control to provide an estimate of the acceleration fault δ_y in (8.13).

Let $v_{\text{eq}} \in \mathbb{R}^n$ be the equivalent control vector. Then, its i-th component is derived solving $\dot{\sigma}_{1,i} = 0$ for v_i. In particular

$$\begin{aligned}\dot{\sigma}_{1,i}(t) &= \beta e_{2,i}(t) + \dot{e}_{2,i}(t) \\ &= \beta e_{2,i}(t) + \ddot{q}_{c,i}(t) - \ddot{\hat{q}}_{c,i}(t) \\ &= \beta e_{2,i}(t) + y_i(t) + \delta_{y,i}(q,t) - y_i(t) - v_i(t) \\ &= \beta e_{2,i}(t) + \delta_{y,i}(q,t) - v_{\text{eq},i}(t) = 0.\end{aligned}$$

Then, since the condition $\sigma_{1,i} = \sigma_{2,i} = 0$ implies that $e_{2,i}$ goes exponentially to zero, it holds that

$$\lim_{t\to\infty} v_{\text{eq},i}(t) = \delta_{y,i}(q,t). \tag{8.27}$$

Since the observer input is continuous, one can write the estimate of the i-th component acceleration fault vector as

$$\hat{\delta}_{y,i}(t) = v_i(t) = \int_0^t w_i(z)\,\mathrm{d}z, \tag{8.28}$$

meaning that the filtering operation is performed by the integrator in (8.28), and it is possible to conclude that the signal $v_i(t)$ can be used as an estimate of the acceleration fault δ_y induced by the actuator fault δ_τ.

Analogously to what is done for sensor FD, it is possible to define a vector $f_a \in \{0,1\}^n$, whose elements are given by

$$f_{a,i} = \begin{cases} 0, & \text{if } \underline{\Delta}_{a,i} \leq \hat{\delta}_{y,i} \leq \bar{\Delta}_{a,i} \\ 1, & \text{otherwise} \end{cases} \tag{8.29}$$

where $\underline{\Delta}_{a,i} \in \mathbb{R}$ and $\bar{\Delta}_{a,i} \in \mathbb{R}$, with $\underline{\Delta}_{a,i} < \bar{\Delta}_{a,i}$ being the lower and upper thresholds.

Note that, since δ_τ is related to δ_y via (8.13), one has that

$$\hat{\delta}_\tau(t) = M(q_c)\hat{\delta}_y(t), \tag{8.30}$$

which means that, even though the the above strategy is perfectly able to detect and isolate the acceleration fault induced by the actuator fault, the estimate of this last one is indeed related to the quality of the sensor fault estimate, which must be sufficiently accurate, having the error $q - q_c$ sufficiently small.

8.4 Simulations

The performances of the FD scheme depicted in Figure 8.2 have been assessed in simulation relying on the virtualized model of the Franka Emika Panda, a robot characterized by $n = 7$

joints, described in Appendix C. In particular, the robot is controlled to reach a desired configuration $q^\star \in \mathbb{R}^7$ using a PD controller characterized by gain matrices $K_p = 3\, I_7$ and $K_d = 2\, I_7$, while being subject to one of the FE listed in Table 8.1. In all the simulations, the parameters of the SOSM UIOs are selected as $\alpha_i = 100$, for $i \in \{1, 2, \dots, n\}$, and $\beta = 1$. Moreover, in order to emulate a more realistic scenario, a white noise with amplitude equal to 0.2 degrees has been added to the sensor measures. The thresholds in (8.29) are selected as $\underline{\Delta}_{a,i} = -0.5$ and $\bar{\Delta}_{a,i} = 0.5$, while those in (8.18) are selected as $\underline{\Delta}_{s,i} = -0.026$ and $\bar{\Delta}_{s,i} = 0.026$.

The DRL agent employed to perform sensor fault diagnosis relies on the TD3 algorithm described in in Algorithm 3, with state space $\mathcal{S} \subset \mathbb{R}^{10}$, action space $\mathcal{A} \subset \mathbb{R}^7$ and reward r_t being defined as in (8.14), (8.15), and 8.17, respectively. The training procedure is divided into episodes, and each episode has a fixed duration of 5 seconds such that at each time-step the simulation advances of 4.2 milliseconds. During each episode, a vector of random constant bounded sensor faults δ_q is injected into the system. In particular, it holds that $|\delta_{q,i}| \leq 0.35$, for $i \in \{1, 2, \dots, n\}$. Both the actor and critic are approximated by using DNNs with an input layer with 10 neurons, 2 hidden layers with 64 neurons each, and an output layer with 7 neurons. In Figure 8.3 it is illustrated the average cumulative reward R_t during the training procedure. One can notice that, as expected, the reward tends to increase during the learning phase, meaning that the agent learns how to perform the fault diagnosis whenever a sensor failure occurs.

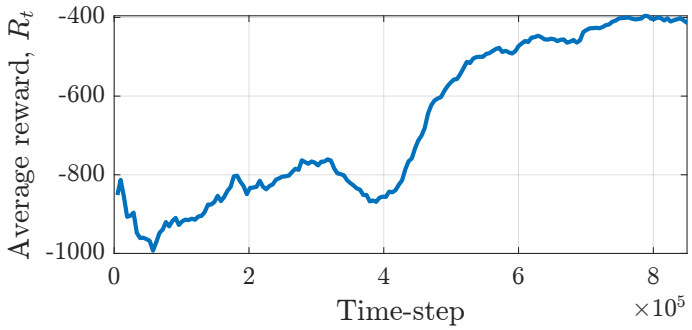
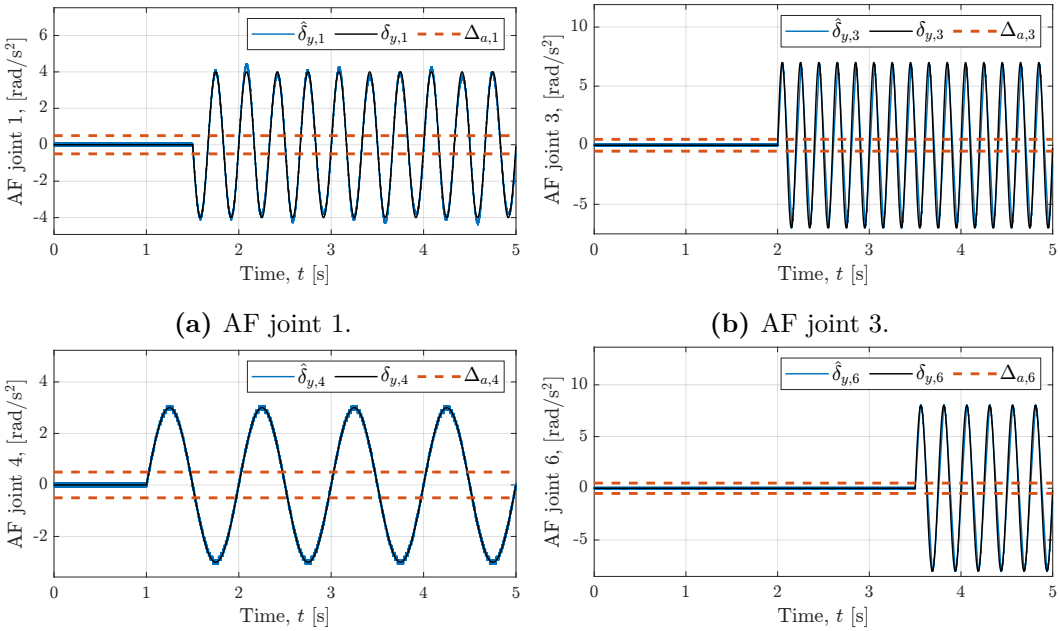


Figure 8.3: Evolution of the average cumulative reward during the training procedure.

Scenario FE3: Faults on multiple actuators In this scenario, joints 1, 3, 4, and 6 are affected by actuator fault, having

$$\delta_y(t) = \begin{bmatrix} 4\sin(6\pi t) & 0 & 7\sin(10\pi t) & 3\sin(2\pi t) & 0 & 8\sin(4\pi t) & 0 \end{bmatrix}^\top. \tag{8.31}$$

The result of the simulation, depicted in Figure 8.4, show that the battery of SOSM UIOs is capable of identifying in finite time the fault on the corrupted joints. Then, the Root Mean Square (RMS) of the estimation error has been computed for all the considered actuators, and the results are presented in Table 8.2.



(a) AF joint 1. **(b)** AF joint 3.

(c) AF joint 4. **(d)** AF joint 6.

Figure 8.4: Time evolution of the actual and estimated actuator faults for joints 1, 3, 4 and 6 in the case of scenario FE3.

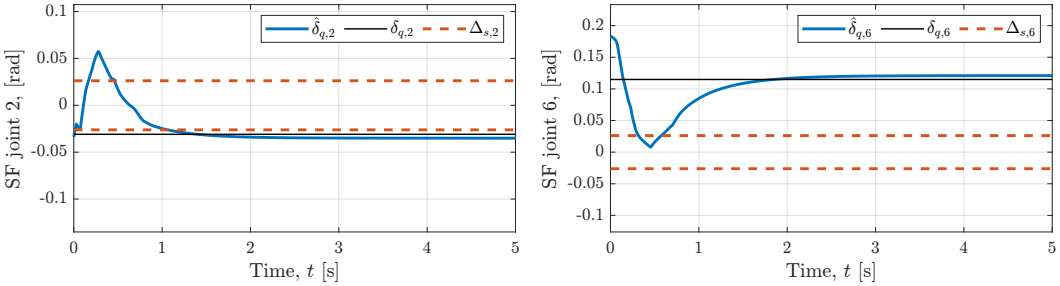
Joint #	RMS $(\hat{\delta}_y - \delta_y)$ [rad s^{-2}]
1	0.2107
3	0.2698
4	0.5392
6	0.2234

Table 8.2: RMS estimation errors for scenario FE3.

Scenario FE4: Faults on multiple sensors In this scenario, the effectiveness of the proposed DRL algorithm, is assessed. In particular, constant fault signals are injected into joints 2 and 6, having $\delta_{q,2} = -0.0309$ and $\delta_{q,6} = 0.1149$, with the two quantities expressed in radians.

The results are satisfactory and illustrated in Figure 8.5, while in Table 8.3 the RMS estimation errors are reported to confirm the effectiveness of the proposed approach.

Scenario FE5: Faults on multiple sensors and actuators In this scenario, actuator and sensor faults contemporaneously occurs on the same joint. In particular, joint 1 is affected by a constant sensor fault $\delta_{q,1} = -0.247$ radiance for $t \geq 1.5$ and by a sinusoidal acceleration fault $\delta_{y,1} = 3\sin(8\pi t)\,\text{rad}\,\text{s}^{-2}$ starting for $t \geq 1$, while on joint 7 only an acceleration fault $\delta_{y,7} = 5\sin(12\pi t)\,\text{rad}\,\text{s}^{-2}$ occurs from $t = 3$. The time evolution of the



(a) SF joint 2. **(b)** SF joint 6.

Figure 8.5: Time evolution of the actual and estimated sensor faults for joints 2 and 6 in the case of scenario FE4.

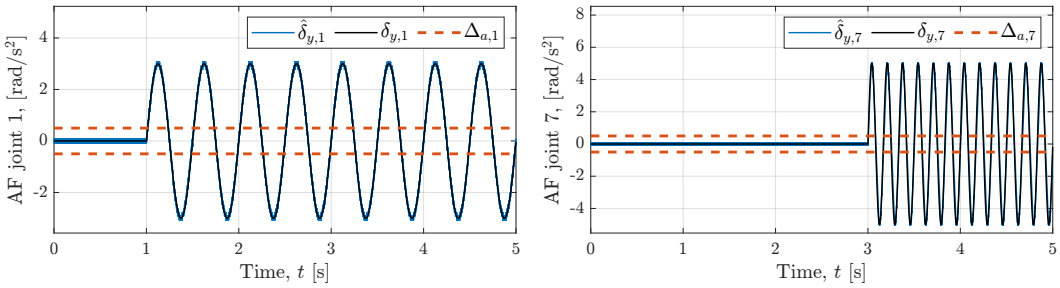
Joint #	RMS $(\hat{\delta}_q - \delta_q)$ [rad]
2	0.021
6	0.032

Table 8.3: RMS estimation errors for scenario FE4.

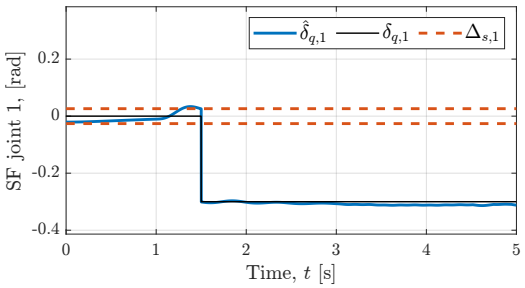
actual and estimated faults is depicted in Figure 8.6, while the RMS od the estimation errors is presented in Table 8.4.

Joint #	RMS $(\hat{\delta}_q - \delta_q)$ [rad]	RMS $(\hat{\delta}_y - \delta_y)$ [rad s^{-2}]
1	0.031	0.3226
7	-	0.194

Table 8.4: RMS estimation errors for scenario FE5.



(a) AF joint 1. **(b)** AF joint 7.



(c) SF joint 1.

Figure 8.6: Time evolution of the actual and estimated sensor and actuator faults for joints 1 and 7 in the case of scenario FE5.

Part IV

Applications to Human-Robot Interaction

Chapter 9

Human-Robot Ergonomic Handover via Adaptive DNN-ISM

The number of situations in which humans and robots share their workspace has greatly increased. When there is physical Human Robot Interaction (pHRI) [139], it is fundamental that such an interaction causes the minimum psycho-physical stress to the human operator. Indeed, it has been studied how Muscoloskeletal Disorders (MSDs), which are mainly caused by repetitive work and poor posture, constitute more or less the thirty percent of all occupational diseases in the United states, Nordic countries, and Japan [140, 141].

Hence, it is fundamental to design comfortable workspaces putting particular attention into ergonomics [142]. When there is pHRI, this can be done by controlling the robot so that it adapts its motion to the human operator movements, while still performing the task proficiently. One of the most common operations that involves collaboration between humans and robots is the so-called *handover*, which consists into an exchange of objects directly from the human to the machine, or vice versa. Usually, the handover operation is performed by specifying a fixed location and orientation in space. However, such a combination may not be the most comfortable for the operator and could lead to bad posture and, if performed for long periods of time, to an increased psycho-physical stress and cause MSDs.

Such a problem has been addressed in several works, with the objective of developing strategies for ensuring *ergonomic handover*. For example, [143] proposes a methodology that learns the most ergonomic way of passing objects to a person relying on data gathered during the interaction, while in [144] a whole-body dynamic model of the human operator is employed with the aim of optimizing the location of the co-manipulation task inside the workspace.

In general, when an object is grasped by a robotic manipulator, it exerts a torque on

the joints of this last one, acting as a disturbance that, if not taken into consideration in the controller design, could interfere with the completion of the task. In such situations, SMC has been proved to be an effective technique, as detailed in Chapter 2. In order compensate a disturbance acting on the robotic system, one could rely on ISM control, which, described in Section 2.4, relies on the complete knowledge of the system dynamics.

In robotic applications, such a knowledge may be not fully available, due to the difficulty in modeling the Coriolis term and the friction vectors. For this reason, one could rely on the DNN-ISM framework presented in Chapter 5, using a DNN to estimate the combination of the aforementioned term. However, as detailed in Chapter 5, this would require the knowledge of the bounds of the DNN approximation error and on the worst possible realization of the disturbance induced by the grasped object, which, in the case of the handover operation, has an unknown mass and shape.

With these premises, the aim of this chapter is to present the work contained in [145], in which a version of the DNN-ISM with adaptive discontinuous gain is proposed to develop a strategy for performing human-robot handover operation in an ergonomic way, relying on an Inertial Measurement Unit (IMU) placed on the back of the operator's hand.

9.1 Problem Formulation

Consider an open-chain robotic manipulator characterized by $n \in \mathbb{N}_{>0}$ joints. As detailed in Section 4.4, its dynamics are described by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(q,\dot{q}) + g(q) = \tau + \tau_h, \tag{9.1}$$

where q, $\dot{q}$, $\ddot{q} \in \mathbb{R}^n$ are the vectors of the joint positions, velocities, and accelerations, respectively, $M : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the inertia matrix, which is symmetric, positive definite, and bounded as in (4.18a), $C : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the matrix of the Coriolis and centripetal effects, $F : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is the vector of the friction terms, $g : \mathbb{R}^n \to \mathbb{R}^n$ represents the gravity components, $\tau \in \mathbb{R}^n$ is the vector of input torques, while $\tau_h \in \mathbb{R}^n$ represents the torques induced by external forces acting on the robot during the handover operation.

If one defines the state vector $x = \begin{bmatrix} q^\top & \dot{q}^\top \end{bmatrix}^\top \in \mathcal{X}$, with $\mathcal{X} \subset \mathbb{R}^{2n}$ being a compact set containing that depends on the mechanical limits of the robot joints, it is possible to express (9.1) in the canonical reduced form (2.17), having

$$\dot{x}(t) = \begin{bmatrix} \dot{q}(t) \\ \ddot{q}(t) \end{bmatrix} = \begin{bmatrix} \dot{q}(t) \\ M(q(t))^{-1}\Big(\tau_h(t) - \nu(q(t),\dot{q}(t))\Big) + M(q(t))^{-1}\tau(t) \end{bmatrix}, \tag{9.2}$$

where $\nu(q,\dot{q}) = C(q,\dot{q})\dot{q} + F(q,\dot{q}) + g(q)$.

Since the dynamical parameters of the object manipulated by the robot during the handover are not available, the term $M(q)^{-1}\tau_h$ act as a disturbance. Due to the nature of the object, and from the fact that $M(q)$ is bounded as in (4.18a), the following assumption holds

Assumption 9.1. *There exists a constant* $\bar{\delta} \in \mathbb{R}_{>0}$ *such that*

$$\sup_{x \in \mathcal{X}} \left\| M(q)^{-1} \tau_h \right\| \leq \bar{\delta}. \tag{9.3}$$

Let the model in (9.1) be fully available, with exception of τ_h. Then, with the objective of tracking a desired trajectory $x^\star = \begin{bmatrix} (q^\star)^\top & (\dot{q}^\star)^\top \end{bmatrix}^\top \in \mathcal{X}$, while compensating the uncertain term $M^{-1}(q)\tau_h$, one could design an ISM controller as presented in Section 2.4.

Considering the robot model in (9.2), the ISM controller would be

$$\tau(t) = \tau_n(t) + \tau_r(t), \tag{9.4}$$

where $\tau_n \in \mathbb{R}^n$ is a control law that stabilizes the system on the desired trajectory in the case of $\tau_h = 0_n$, while $\tau_r \in \mathbb{R}^n$ is the discontinuous robustifying term defined as

$$\tau_r(t) = -\rho \frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \tag{9.5}$$

with $\rho \in \mathbb{R}_{>0}$ being the discontinuous control gain, and $\sigma : \mathcal{X} \to \mathbb{R}^n$ being the so-called integral sliding variable. This last one is defined as

$$\sigma(x(t)) = \sigma_0(x(t)) - z(x(t)). \tag{9.6}$$

In this case, the conventional sliding variable $\sigma_0 : \mathcal{X} \to \mathbb{R}^n$ is defined as

$$\sigma_0(x(t)) = C_1 \Big[q(t) - q^\star(t) \Big] + C_2 \Big[\dot{q}(t) - \dot{q}^\star(t) \Big], \tag{9.7}$$

with C_1, $C_2 \in \mathbb{R}^{n \times n}$ being symmetric and positive-definite design matrices. As for the transient variable $z : \mathcal{X} \to \mathbb{R}^n$, it is characterized by the dynamics

$$\dot{z}(x(t)) = C_1 \Big[\dot{q}(t) - \dot{q}^\star(t) \Big] + C_2 \Big[M(q(t))^{-1} \nu(q(t), \dot{q}(t)) + M(q)^{-1} \tau_n(t) - \ddot{q}^\star(t) \Big], \tag{9.8}$$

with $z(x(t_0)) = \sigma_0(x(t_0))$.

As detailed in [36], if the discontinuous control gain in (9.5) is designed so that it dominates the worst realization of the external disturbance $M(q)^{-1}\tau_h$, which is $\bar{\delta}$, then a sliding mode $\sigma(x(t)) = 0_n$ is established for each $t \geq t_0$.

The objective is to design a control strategy that allows to perform ergonomic handover in condition of partially unknown dynamics, in the case in which bounds on the disturbance or on the DNN approximation error are not available.

9.2 Adaptive DNN-ISM for ergonomic handover

The strategy can be divided into two parts, i.e., the *reference generation*, whose aim is to provide a reference trajectory for the robot that enables ergonomic handover, and the *control* via adaptive DNN-ISM, whose aim is to track the aforementioned reference, without relying on conservative bounds.

9.2.1 Reference generation

In the following, it is assumed, that the objective is to control the robot so that it reaches a pose comfortable for the human operator, grasps an object with unknown shape and mass from the operator's hand, and places it at a predefined location.

Hence, it is possible to divide the handover task into two different phases, i.e., the *hand reaching* phase and the *object placement* phase. The former one, summarized in in the block diagram in Figure 9.1, is the described in the following.

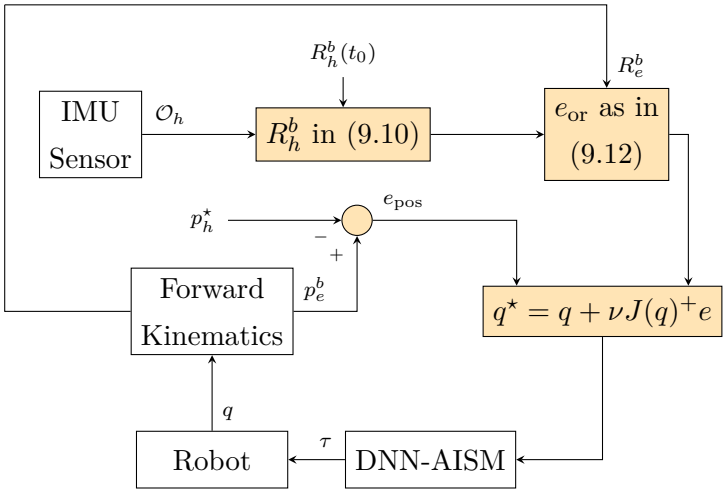


Figure 9.1: Block diagram of the hand reaching strategy. The blocks responsible for the computation of the joints reference are colored in yellow.

Let $O_b - x_b y_b z_b$ be a fixed reference frame and $O_e - x_e y_e z_e$ a reference frame attached to the end-effector. Then, it is possible to express the forward kinematics of the manipulator by means of the homogeneous transformation matrix

$$T_e^b(q) = \begin{bmatrix} R_e^b(q) & p_e^b(q) \\ 0_3^\top & 1 \end{bmatrix}, \tag{9.9}$$

with $p_e^b \in \mathbb{R}^3$ and $R_e^b \in SO(3)$ being the position and orientation of O_e with respect to O_b, respectively. Moreover, let $O_h - x_h y_h z_h$ be a reference frame attached to the IMU sensor placed on the back of the human operator's hand. Then, the orientation of O_h with respect to O_b is denoted by the matrix $R_h^b(t) \in SO(3)$.

The following assumption about the desired interaction position is introduced.

Assumption 9.2. *The position $p_h^\star \in \mathbb{R}^3$, expressed with respect to the base frame O_b, in which the contact between the human operator hand and the robot end-effector happen, is defined a before the execution of the handover task.*

The above assumption is reasonable, since $p_h^\star$ could be defined before the start of the task by hand guiding the robot end-effector into a position which is considered comfortable for the operator.

The same cannot be said for the orientation with which the robot should approach for the handover operation. In fact, it depends on different aspect, including the shape of the object exchanged during the interaction. For this reason, such an orientation is defined relying on measurements of an IMU sensor placed on the back of the operator's hand, which, at each time instant, provides the set of orientation angles

$$\mathcal{O}_h(t) = \{\phi(t), \theta(t), \psi(t)\},$$

expressed with respect to its own frame, with this last one being defined at the instant in which the sensor calibration occurred, i.e., $t = t_0$. The orientation of the operator's hand with respect to the fixed base frame O_b is given by

$$R_h^b(t) = R_h^b(t_0) R_{z_h}(\psi(t)) R_{y_h}(\theta(t)) R_{x_h}(\phi(t)), \tag{9.10}$$

where R_{x_h}, R_{y_h}, $R_{z_h} \in SO(3)$ represent the basic rotations around the sensor axes, while $R_h^b(t_0) \in SO(3)$ is the orientation of the IMU with respect to the base frame at the initial time instant.

In order to compute the reference for the robot joints, the position error and the orientation error should be computed. The former one can be computed easily as

$$e_{\text{pos}}(t) = p_e^b(t) - p_h^\star. \tag{9.11}$$

As for the orientation error, it is computed according to the following procedure. First, the matrix that describes the orientation of O_h with respect to O_e is defined as

$$\begin{aligned} R_h^e(t) &= (R_e^b(t))^{-1} R_h^b(t) \\ &= (R_e^b(t))^\top R_h^b(t). \end{aligned}$$

Then, an axis-angle representation of of R_h^e is computed according to Algorithm 4, obtaining an angle $\alpha \in \mathbb{R}$ and a rotation axis $\hat{w} \in \mathbb{R}^3$. Finally, the rotation error $e_{\text{or}} \in \mathbb{R}^3$, expressed in the frame O_b is obtained as

$$e_{\text{or}}(t) = -\alpha(t) R_e^b(t) \hat{w}(t). \tag{9.12}$$

The vector of the pose error can be then defined as $e = \begin{bmatrix} e_{\text{pos}}^\top & e_{\text{or}}^\top \end{bmatrix}^\top \in \mathbb{R}^6$, and the reference $q^\star \in \mathbb{R}^n$ for the joint positions is computed according to

$$q^\star(t) = q(t) + \kappa J(q(t))^+ e(t), \tag{9.13}$$

where $\kappa \in \mathbb{R}_{>0}$ is a design parameter, $J \in \mathbb{R}^{n \times 6}$ is the geometric Jacobian of the robot, computed as in (4.8).

The joint reference (9.13) is passed to the adaptive DNN-ISM controller, described in the next section. As soon as the condition $\|e\| \leq \epsilon_r$, with $\epsilon_r \in \mathbb{R}_{>0}$ being a small design constant, the end-effector is commanded so that it grasps the object. If the grasping procedure is successfully carried out, the robot is controlled reach a configuration $q_{\text{place}} \in \mathbb{R}^n$ and place the object. Hence, in this phase, a reference $q^\star = q_{\text{place}}$ is provided to the adaptive

DNN-ISM controller. As soon as $\|q - q^\star\| \leq \epsilon_p$, with $\epsilon_p \in \mathbb{R}_{>0}$ being a small design constant, the end-effector is commanded to release the object. As soon this last one is released, the robot reference is generated again according to the block diagram depicted in Figure 9.1, until the task is completed.

9.2.2 Adaptive DNN-ISM control

The adaptive DNN-ISM controller, depicted in Figure 9.2, and employed to drive the robot towards the desired configuration in the case of fully unknown dynamics of the object and partially unknown dynamics of the robot, is presented in the following.

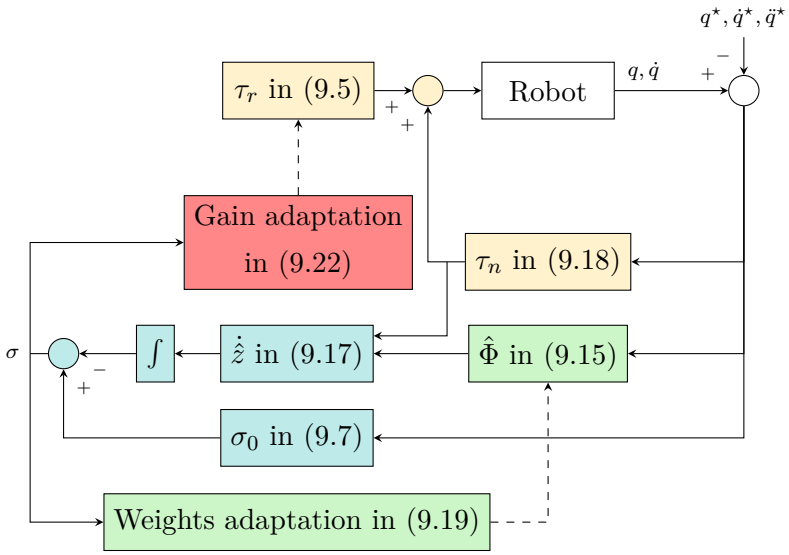


Figure 9.2: Block diagram of the adaptive DNN-ISM control scheme. The yellow blocks are associated with the control law, the blue and the green ones are related to the sliding variable and the DNN, respectively.

Assume the components $\nu(q, \dot{q})$ and τ_h appearing in (9.2) to be unknown in structure and unmeasurable. Differently from what done in Section 9.1, in which τ_h is treated as a disturbance, in the following it is considered as a part of the dynamics that must be learned. Thus, as described in Chapter 5, there exists a DNN $\Phi : \mathcal{X} \to \mathbb{R}^n$ characterized by ideal weights such that

$$\tau_h - \nu(q, \dot{q}) = \Phi_{k_\Phi} + \varepsilon_\Phi. \tag{9.14}$$

Note that the weights and the approximation error ε_Φ satisfies Assumption 5.6. Similarly, a version of the same network, but with estimated weights, denoted as $\hat{\Phi}$, allows to write

$$\tau_h - \nu(q, \dot{q}) = \hat{\Phi}_{k_\Phi}. \tag{9.15}$$

Then, it is possible to define the integral sliding variable as

$$\sigma(x(t)) = \sigma_0(x(t)) - \hat{z}(x(t)), \tag{9.16}$$

where σ_0 is defined as in (9.7), while the dynamics of the transient variable $\hat{z}$ depends on the estimate (9.15), having

$$\dot{\hat{z}}(x(t)) = C_1\Big[\dot{q}(t) - \dot{q}^\star(t)\Big] + C_2\Big[M(q(t))^{-1}\hat{\Phi}_{k_\Phi} + M(q)^{-1}\tau_n(t) - \ddot{q}^\star(t)\Big], \tag{9.17}$$

with $\hat{z}(x(t_0)) = \sigma_0(x(t_0))$.

The nominal part τ_n of the control law in (9.17) and (9.4) is chosen so that it stabilizes the robot around the trajectory coming defined in Section 9.2.1. In particular, τ_n is designed as

$$\tau_n(t) = -M(q(t))\left(K_p\Big(q(t) - q^\star(t)\Big) + K_d\Big(\dot{q}(t) - \dot{q}^\star(t)\Big) - \ddot{q}^\star(t)\right) + \hat{\Phi}_{k_\Phi}, \tag{9.18}$$

with K_p, $K_d \in \mathbb{R}^{n\times n}$ being design matrices with positive entries.

The weights of $\hat{\Phi}$ are updated according to the dynamics given by

$$\operatorname{vec}\left(\dot{\hat{V}}_j\right) = \operatorname*{proj}_{\mathcal{B}_{\Phi_j}}\left(\Gamma_{\Phi_j}\Lambda_{\Phi_j}^\top M^{-\top}C_2^\top\sigma\right), \tag{9.19}$$

where $\Gamma_{\Phi_j} \in \mathbb{R}^{L_{\Phi_j}L_{\Phi_{j+1}}\times L_{\Phi_j}L_{\Phi_{j+1}}}$ is the diagonal matrix with positive entries which represent the learning rate, $\Lambda_{\Phi_j} \in \mathbb{R}^{n\times L_{\Phi_j}L_{\Phi_{j+1}}}$ is the one defined in (5.26b), while $M^{-\top} \equiv (M^{-1})^\top$.

If one applies the reasoning behind Theorem 5.1, it would be possible to conclude that, if the discontinuous control gain in (9.5) is chosen as $\rho = \rho^\star$, with

$$\rho^\star > \frac{\left\|C_2M^{-1}\right\|(\bar{c}_\Phi + \bar{\varepsilon}_\Phi) + \eta^\star}{\underline{\lambda}(C_2M^{-1})}, \tag{9.20}$$

where $\bar{c}_\Phi$ comes from Proposition 5.2, $\bar{\varepsilon}_\Phi$ is the one in Assumption 5.6, and $\eta^\star \in \mathbb{R}_{>0}$, then a sliding mode $\sigma(x(t)) = 0_n$ is enforced for $t \to \infty$.

However, $\bar{c}_\Phi$ and $\bar{\varepsilon}_\Phi$ may be derived by over conservative bounds, resulting in a large control gain that could cause damage to the robot joints. For this reason, in the following such quantities are assumed to be unknown and the robustifying control law τ_r is modified as

$$\tau_r(t) = -\hat{\rho}(t)\frac{\sigma(x(t))}{\|\sigma(x(t))\|}, \tag{9.21}$$

with $\hat{\rho} \in \mathbb{R}_{>0}$ being now an adaptive parameter.

Theorem 9.1. *Consider the robotic manipulator* (9.2)*, the control law* $\tau = \tau_n + \tau_r$*, with* τ_n *and* τ_r *defined as in* (9.18) *and* (9.21)*, respectively, the integral sliding variable in* (9.16)*, the weight adaptation laws* (9.19)*. If the discontinuous control gain in* (9.21) *is adapted as*

$$\dot{\hat{\rho}} = \mu\,\|\sigma\|\left\|C_2M^{-1}\right\| sign\left(\|\sigma\| - \varepsilon_\sigma\right), \tag{9.22}$$

with $\hat{\rho}(t_0) = 0$*,* $\mu \in \mathbb{R}_{>0}$ *being the adaptation rate, and* $\varepsilon_\sigma \in \mathbb{R}_{>0}$ *being a leaking factor which allows* $\hat{\rho}$ *to decrease when* $\|\sigma\| < \varepsilon_\sigma$*, then* σ *is ultimately bounded into the set*

$$\Omega_\sigma \coloneqq \{\sigma \in \mathbb{R}^n : \|\sigma\| \le \varepsilon_\sigma\}. \tag{9.23}$$

Proof. See Appendix B.11 □

9.3 Experiment

The ergonomic handover strategy presented in Figure 9.1 and Figure 9.2 has been validated experimentally on the Franka Emika Panda, a collaborative robot with $n = 7$ DoF described in Appendix C. The orientation of the hand are retrieved via the MTw Awinda IMU, worn by the human operator by means of a glove.

Specifically, such a DNN is characterized by $k_\Phi = 2$ hidden layers with 16 neurons, all activated using the hyperbolic tangent function. The weights are adjusted according to (9.19), setting each matrix Γ_{Φ_j} equal to an identity matrix with suitable dimensions. The gain matrices of the nominal control law in (9.18) are chosen as $K_p = K_d = 5\, I_7$. The parameters of the conventional sliding variable in (9.7) are chosen as $C_1 = C_2 = I_7$, while the discontinuous control gain $\hat{\rho}$ is updated according to (9.22), with $\mu = 0.85$ and $\varepsilon_\sigma = 0.3$. Finally, the reference $q^\star$ in (9.13) has been generated considering $p_h^\star = \begin{bmatrix} 0.47 & 0.18 & 0.47 \end{bmatrix}^\top$ [m], and choosing $\kappa = 0.6$.

During the experiment, whose duration is 90 seconds, the human operator is required to perform the handover of four different objects, depicted in Figure 9.3, characterized by different shape and mass. In particular, the masses of the objects vary from few grams up to 1.5 kilograms. The results of the experiment are depicted in Figure 9.4. In particular from the first plot is is possible to see how the the robot pose is successfully driven to the desired one, dictated by the IMU measurements, thanks to the reference generation scheme in Figure 9.1. The yellow areas indicate the time instants in which the the flag $\delta_g \in \{0, 1\}$, defined as $\delta_g = 1$ if an object has been grasped and $\delta_g = 0$ otherwise, is equal to one. The second plot depicts the norm of the integral sliding variable σ. From it, it is possible to conclude that, except for the times in which objects are grasped or released, resulting in a change of dynamics and requiring an adaptation of the DNN, a practical sliding mode on the set Ω_σ is achieved successfully. Finally, the third plot depicts the time behavior of the adaptive discontinuous control gain, showing that, even with reasonably a small value, it is able to ensure a practical sliding mode.

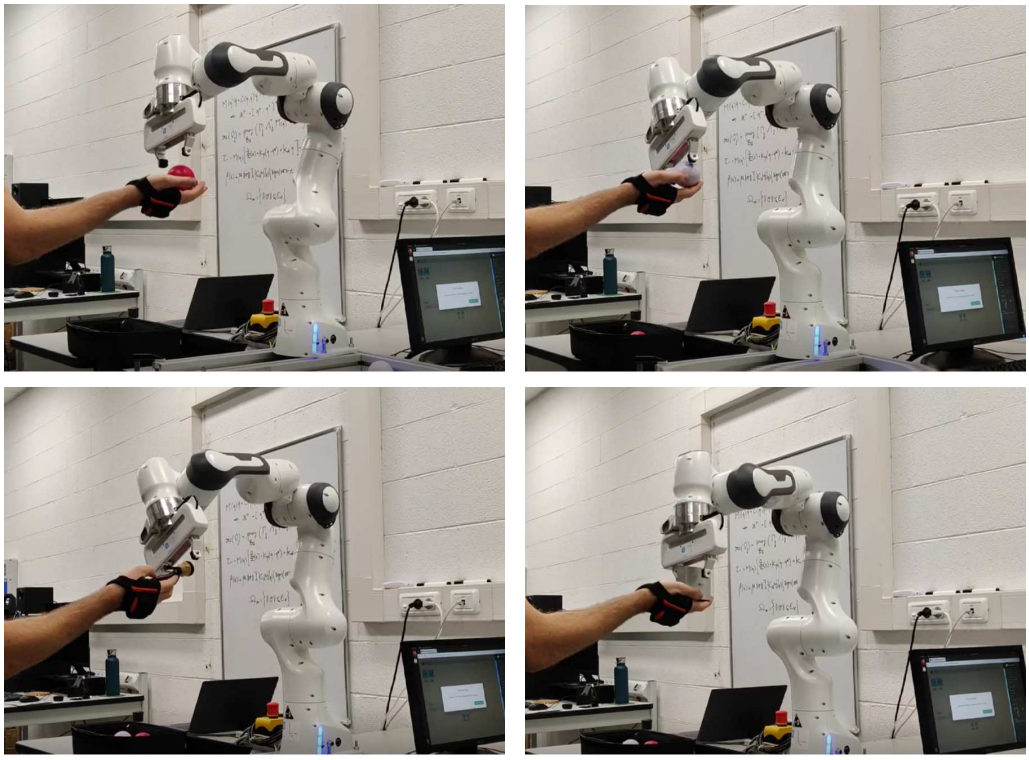


Figure 9.3: Instants in which the handover operation is performed. The robot adapts the pose of its end-effector so that it follows the orientation of the IMU sensor.

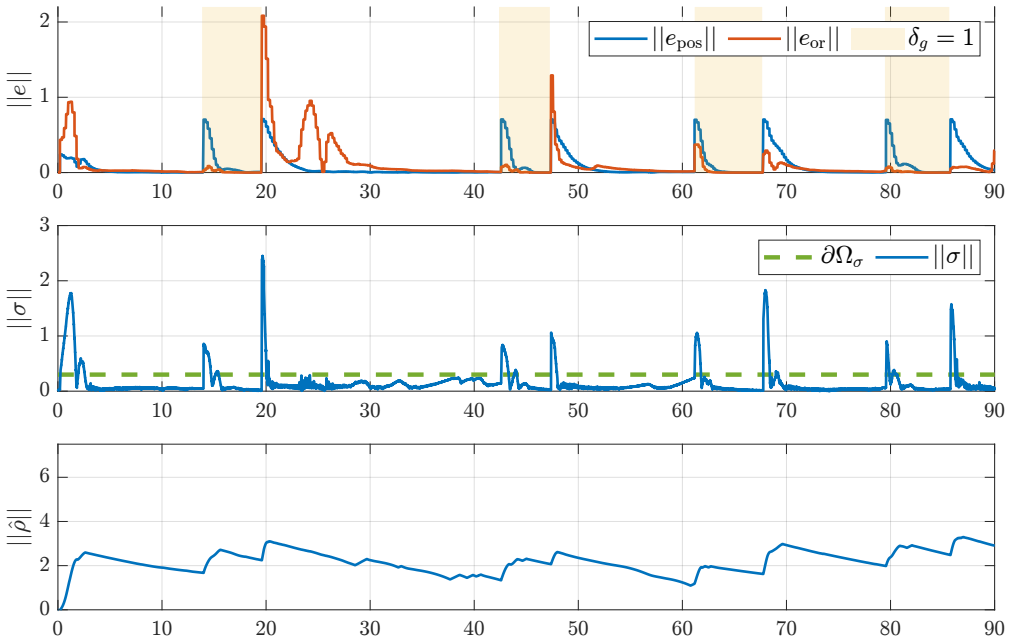


Figure 9.4: Time evolution of the pose error (first plot), norm of the sliding variable (second plot), and adaptive control gain (third plot).

Chapter 10

Vision-based Collision Avoidance with ISM control for Collaborative Robots

The task of obstacle detection, i.e., estimation of pose and size of volumes present in the robot workspace that are different from the target manipulation volume and the collaborative robot (cobot) itself, is fundamental to implement a collision avoidance scheme. One of the most convenient ways to perform obstacle detection involves the use of RGBD cameras, which provide information under the form of point clouds. For example, one could process the point cloud coming from the vision sensor and give them in input to a Convolutional Neural Network (CNN), whose output gives information about the obstacle, like is made in [146]. However, one of the main difficulties is that the large quantity of 3D data captured by sensors could result in computationally expensive algorithms that are ill-suited for real-time collision avoidance.

The aim of this chapter is to introduce a collision avoidance strategy which, relying on data provided by a single RGBD sensor, generates a reference for the robot joints that ensure safety for the human operator. Then, such a reference is followed controlling the robot with an ISM controller in order to deal with possible uncertainties. In this case, the neural network component is not embedded directly in the controller but, as detailed later in the chapter, in the initialization of the obstacle detection scheme.

10.1 Problem Formulation

Consider an open chain robotic manipulator with $n \in \mathbb{N}_{>0}$ joints, whose dynamics and kinematics are expressed as described in Chapter 4, and let $O_b - x_b y_b z_b$ be a reference frame attached to its base.

Then, let $\mathcal{K}^c(t) \subset \mathbb{R}^3$ denotes the three dimensional point cloud generated by a RGBD

camera at a certain instant t, whose elements are expressed in the camera frame $O_c - x_c y_c z_c$. Such a point cloud is composed by the union of three different subsets as

$$\mathcal{K}^c(t) = \mathcal{R}^c(t) \cup \mathcal{M}^c(t) \cup \mathcal{O}^c(t), \tag{10.1}$$

where $\mathcal{R}^c$ includes the elements of the point cloud belonging to the robot, $\mathcal{M}^c$ contains the elements that are associated with the target manipulation volume, while $\mathcal{O}^c$ collects the points that correspond to obstacles that must be avoided.

To ensure collision avoidance, it is important to control the robot so that the value of the distance

$$d(t) = \min_{a \in \mathcal{R}^c, b \in \mathcal{O}^c} \|a(t) - b(t)\| \tag{10.2}$$

is guaranteed to be maintained above a certain design threshold $d^\star \in \mathbb{R} > 0$, for all $t \geq t_0$. Moreover, it is possible to define the points $p_r^c \in \mathcal{R}^c$ and $p_o^c \in \mathcal{O}^c$, both expressed in the camera frame, such that

$$p_r^c, p_o^c = \underset{a \in \mathcal{R}^c, b \in \mathcal{O}^c}{\operatorname{argmin}} \|a(t) - b(t)\|. \tag{10.3}$$

The collision avoidance strategy described in this chapter can be summarized as follows. At each instant, the point cloud coming from the RGBD camera is analyzed so that to identify the set $\mathcal{O}^c$. Then, a reference for the manipulator that moves p_r^c away from p_o^c is generated and followed via an ISM controller. However, in order to identify the points belonging to the obstacle, is fundamental to first identify the set $\mathcal{R}^c$. To this end, the so-called Hand-Eye (HE) calibration, which identifies the static relation between the reference frames $O_b - x_b y_b z_b$ and $O_c - x_c y_c z_c$, must be performed. In the next section, the employed HE calibration scheme is described.

10.2 The HE calibration scheme

The HE calibration scheme adopted in the collision avoidance strategy is inspired by the one proposed in [147] and depicted in Figure 10.1. The objective of such a scheme is to provide an accurate estimate of the constant homogeneous transformation matrix

$$T_b^c = \begin{bmatrix} R_b^c & p_b^c \\ 0_3^\top & 1 \end{bmatrix}, \tag{10.4}$$

which expresses the pose of $O_b - x_b y_b z_b$ with respect to $O_c - x_c y_c z_c$.

10.2.1 Position estimation

In order to provide a first estimate of the position vector $p_b^c \in \mathbb{R}^3$ in (10.4), denoted as

$$\hat{p}_b^c = \begin{bmatrix} \hat{x}_b^c & \hat{y}_b^c & \hat{z}_b^c \end{bmatrix}^\top \tag{10.5}$$

a YOLOv3 detector [148], which relies on a CNN, has been employed. In general, the aim of an object detector such as YOLOv3, is to identify the pixels of an image that contains

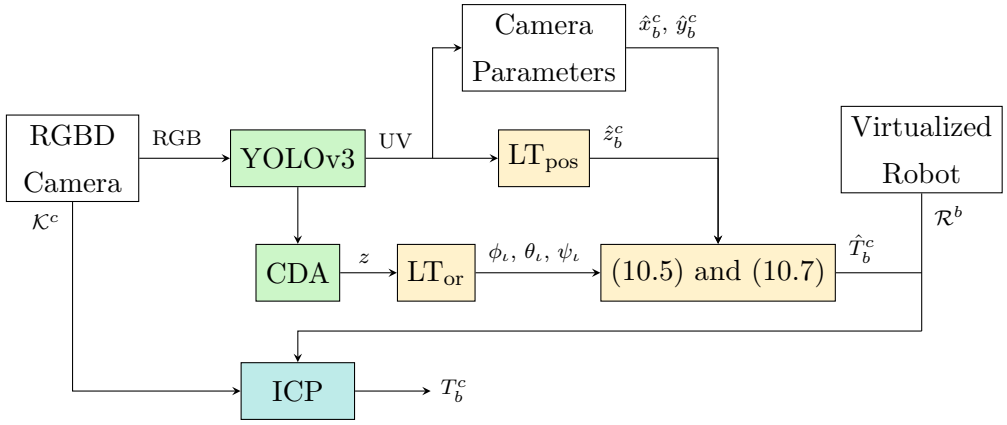


Figure 10.1: Block diagram of the HE calibration scheme. The green blocks are associated with the elements that rely on CNNs, blocks responsible for the computation of the first guess are highlighted in yellow.

an object of interest. In this case, the RGB image captured by the RGBD sensor is given as input to the detector, which returns a copy of the image on which is added a box around the robot.

The UV coordinates of such a box are then used to access a previously defined lookup table, referred as $\mathrm{LT}_{\mathrm{pos}}$ in the block diagram, which maps them to a value for $\hat{z}_b^c \in \mathbb{R}$. The other elements of the vector $\hat{p}_b^c$, i.e., $\hat{x}_b^c$ and $\hat{y}_b^c$, are obtained by transforming the UV coordinates using camera parameters.

To make the proposed scheme capable of working with satisfactorily results without having the manipulator placed in a specific environment, the YOLOv3 detector has been trained on synthetic data subjected to domain randomization [149]. To reduce the gap between the virtual training environment and the real-world workspace, synthetic data has been generated with rendering fidelity, albeit at the cost of computation time during the data generation procedure [150].

To generate the training data, a digital twin of the robot have been created employing physically-based materials and textures extracted directly from photographs of the real manipulator. Then, three light sources, i.e., sun, sphere, and area, have been placed around the simulated environment and activated intermittently in order to gather images with different, realistic diffused lighting. The gathering of the images has been done with a simulated camera set up so that it had the same parameters as the real one. Finally, each image of the digital twin has been overlaid on a random background image selected randomly from a pool of available pictures. Then, each synthetic image is labeled drawing a bounding box around the manipulator, which represent the ground truth for the training. Examples of synthetic images generated with this procedure are shown in Figure 10.2.

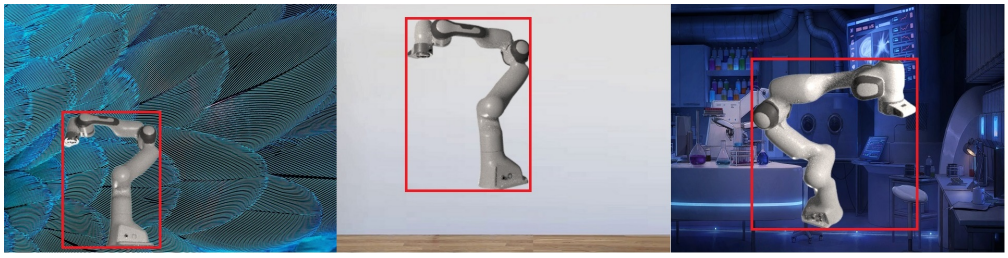


Figure 10.2: Examples of the generated synthetic images with the bounding boxes for training the YOLOv3 detector.

10.2.2 Orientation estimation

The estimation of the robot base frame orientation with respect to the camera frame is carried out by means of a Convolutional Denoising Autoencoder (CDA) [151], depicted in Figure 10.3.

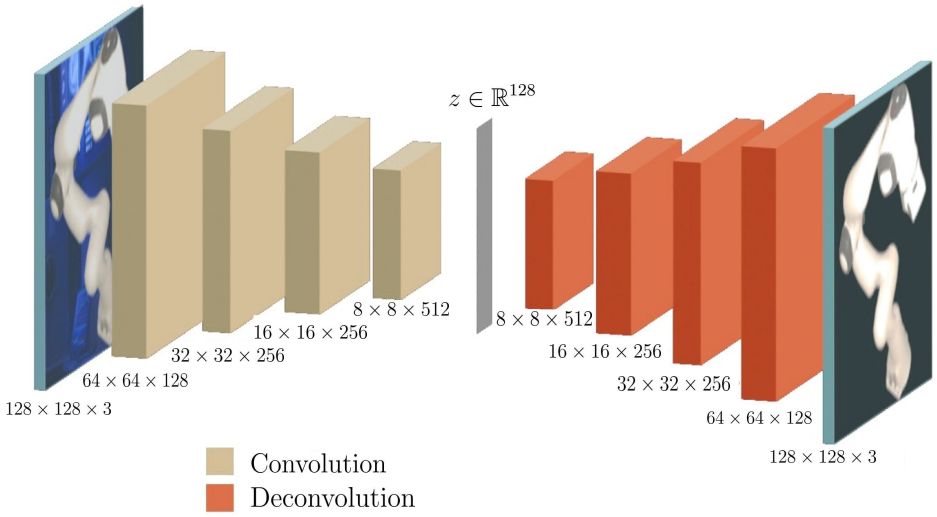


Figure 10.3: Architecture of the CDA employed for orientation estimation.

The CDA is built using two main operations: convolution and deconvolution. In particular, the square input image with size $n_{in} \times n_{in} \times 3$ is convolved until a vector, called *latent space vector* and denoted as $z \in \mathbb{R}^L$, with $L \in \mathbb{N}_{>1}$, is obtained. Then, the deconvolution operation is performed recursively until an output with the same structure of the original input is obtained. In the considered case, the CDA is trained in a supervised way with the aim of obtaining an output image in which everything except from the manipulator is removed, as showed in Fig. 10.3. To do so, a dataset of realistic synthetic images is generated with the methodology presented in the previous section. In this case, the ground truth is the image of the robot without the random background. Examples of two synthetic images

with the corresponding ground truths are presented in Figure 10.4.

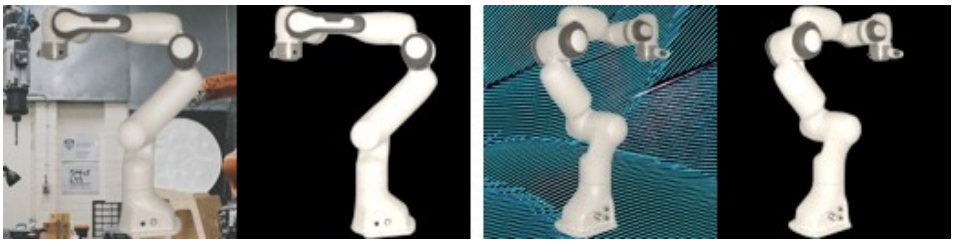


Figure 10.4: Examples of two synthetic images, along with their ground truths, for the training of the CDA.

Once the training procedure is complete, the CDA can be used to provide an estimate of the orientation, expressed by means of Euler angles ϕ, θ, and ψ, of the frame O_b with respect to the camera frame O_c. To do so, only the the first part of the autoencoder is considered, having the latent space vector z as an output. Then, as it is shown in Figure 10.1, such a vector is used to access a lookup table, referred as $\mathrm{LT}_{\mathrm{or}}$ in the block diagram, similar to the one presented in Table 10.1, which associates a set of Euler angles to a specific value of the latent space vector z. In particular, the correct set of Euler angles is selected solving

$$\iota = \underset{j}{\operatorname{argmin}} \left\| z - z_j \right\|. \tag{10.6}$$

Then, the candidate Euler angles are employed to obtain a first estimate of R_b^c, denoted as

$$\hat{R}_b^c = R_{x_c}(\phi_\iota) R_{y_c}(\theta_\iota) R_{z_c}(\psi_\iota), \tag{10.7}$$

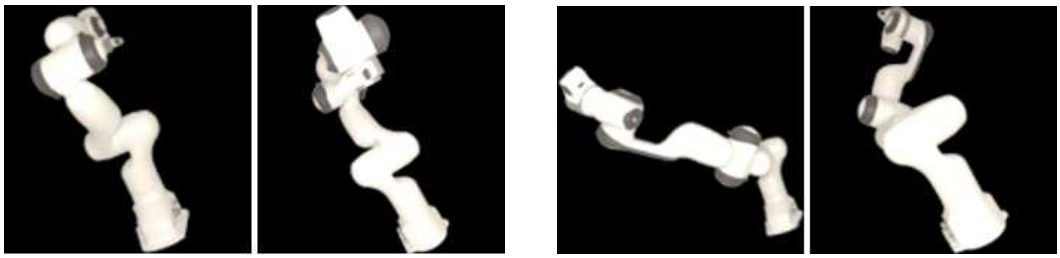
with $R_{x_c}(\cdot)$, $R_{y_c}(\cdot)$, and $R_{z_c}(\cdot)$ denoting the elementary rotations around the axes of the camera frame O_c [24].

The size of the lookup table, denoted by $k \in \mathbb{N}_{>0}$, depends on the granularity $\zeta \in (0, \pi]$ of the angle discretization, chosen by the designer, while the values z_j, with $j \in \{1, 2, \ldots, k\}$, are chosen by providing images of the manipulator placed in a known orientation, given by the Euler angles ϕ_j, θ_j, and ψ_j, as input to the trained CDA.

j	$z \in \mathbb{R}^L$	ϕ [rad]	θ [rad]	ψ [rad]
1	z_0	0	0	0
2	z_1	0.1745	0	0
...	...	...	...	...
$k-1$	z_{k-1}	π	π	3.054
k	z_k	π	π	π

Table 10.1: Example of the lookup table which associates the latent space vector z to a set of Euler angles.

It is worth to notice that, if during the HE calibration the robot joints are in a configuration q_{he} such that the image of the robot given in input to the CDA exhibits complete or partial symmetry, as in the case depicted in Figure 10.5a, the result of the prediction may be incorrect. To overcome such a problem, the orientation estimation is done $n_{\text{he}} \in \mathbb{N}_{>2}$ times, using different configurations $q_{\text{he},i}$, with $i \in \{1, 2, \ldots, n_{\text{he}}\}$ that break axial symmetry in the resulting image, while maintaining the base fixed in space. For each configuration, an image of the robot is captured by the vision sensor and is provided as input to the CDA. The outcome of the operations are n_{he} set of Euler angles. The best candidate is chosen according a voting-based strategy, i.e., at least two predictions have similar outcome. In the case of a tie, the average of the predictions is selected.



(a) Ambiguous orientation **(b)** Non-ambiguous orientation

Figure 10.5: Examples of images in which (a) the robot presents axial symmetry (b) the robot configuration breaks axial symmetry.

10.2.3 Pose estimation adjustment

The previous sections described how a first estimate of the position and orientation, denoted as $\hat{p}_b^c$ and $\hat{R}_b^c$, respectively, are computed. From the developed results, it is possible to obtain a first estimate of the transformation matrix in (10.4), i.e.,

$$\hat{T}_b^c = \begin{bmatrix} \hat{R}_b^c & \hat{p}_b^c \\ 0_3^\top & 1 \end{bmatrix}.$$

However, such an estimate may not be sufficiently accurate due to several aspects, such as the granularity of the lookup tables. For this reason, $\hat{T}_b^c$ is used as initialization of the Iterative Closest Point (ICP) algorithm [152]. In general, the objective of ICP is to find the transformation that allows to align two point clouds.

In this case, the objective is to align the point cloud $\mathcal{K}^c$, retrieved from the RGBD camera and expressed with respect to frame O_c, and the point cloud $\mathcal{R}^b$, obtained from a realistic virtualization of the robotic manipulator, whose points are expressed with respect to frame O_b. The result of the ICP algorithm is the matrix T_b^c in (10.4).

10.3 The collision avoidance scheme

Once the static transformation matrix is computed, the collision avoidance scheme, depicted in Figure 10.6, and presented in the following, can be employed.

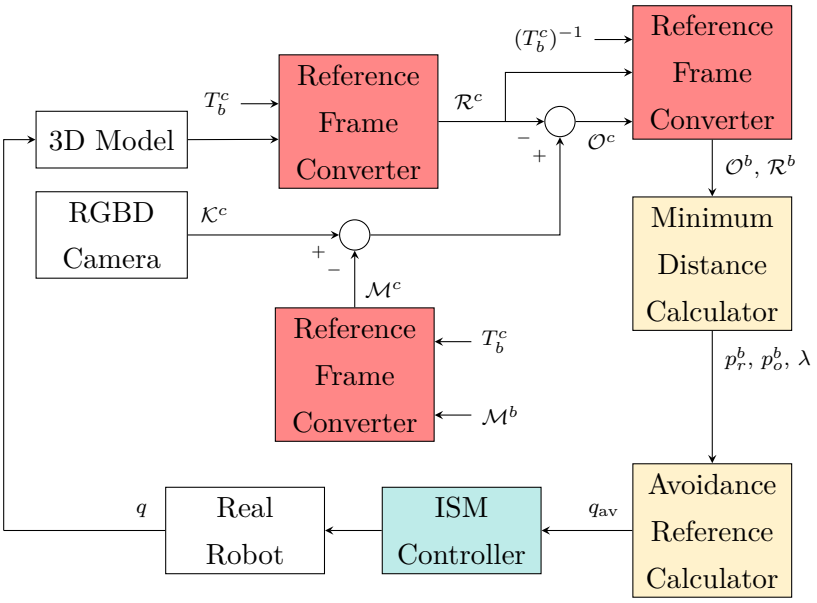


Figure 10.6: Block diagram of the obstacle detection and avoidance scheme. The blocks responsible for the change of reference frame are highlighted in red, the ones that compute the joint position reference in yellow, and the low-level ISM controller in blue.

The objective of the avoidance scheme is to control the robot so that it avoids all the objects captured by the RGBD cameras, with the exception of the manipulation volume. In this case, the manipulation volume is represented by the point cloud $\mathcal{M}^b \subset \mathbb{R}^3$, expressed in the O_b frame, defined by the designer. In order to work properly, the avoidance scheme relies on two main sources of data. The first is the RGBD camera, which generates the point cloud $\mathcal{K}^c$, expressed in the camera frame O_c, that contains all the points belonging to the surface of the objects framed by the camera. The second is the virtualized model of the manipulator, which serves as a digital twin of the physical one. From the model, it is possible to compute the point cloud $\mathcal{R}^b$, expressed in the frame O_b, whose points reconstruct the robot.

Relying on the matrix T_b^c obtained as result of the HE calibration, it is possible to express all the points of $\mathcal{R}^b$ and $\mathcal{M}^b$ into the camera frame, obtaining as result the point clouds $\mathcal{R}^c$ and $\mathcal{M}^c$. Then, the obstacle point cloud can be defined in the camera frame as

$$\mathcal{O}^c = \mathcal{K}^c \setminus (\mathcal{R}^c \cup \mathcal{M}^c). \tag{10.8}$$

Such a point cloud can be expressed in frame O_b by means of T_b^c, obtaining a new set

$\mathcal{O}^b \subset \mathbb{R}^3$ and it is possible to define the vectors $p_r^b \in \mathcal{M}^b$ and $p_o^b \in \mathcal{O}^b$ as

$$p_r^b, p_o^b = \underset{a \in \mathcal{M}^b, b \in \mathcal{O}^b}{\operatorname{argmin}} \|a - b\| . \tag{10.9}$$

Then, to ensure that the robot avoids the points in space belonging to $\mathcal{O}^b$, one could select a non-negative, differentiable function $V_{\mathcal{O}} : \mathcal{R}^b \times \mathcal{O}^b \to \mathbb{R}_{\geq 0}$ which tends to large values as the distance between p_r^b and p_o^b approaches zero, to create an Artificial Potential Field (APF) [153] around the set $\mathcal{O}^b$, such as

$$V_{\mathcal{O}} = \begin{cases} \frac{1}{2}\eta \left(\frac{1}{\lambda - \lambda_0}\right)^2, & \text{if} \lambda \leq \lambda_0 \\ 0, & \text{if} \lambda > \lambda_0 \end{cases} \tag{10.10}$$

where $\lambda \in \mathbb{R}$ is the euclidean distance between p_r^b and p_o^b, i.e., $\lambda = \sqrt{(p_r^b - p_o^b)^2}$, while $\lambda_0 \in \mathbb{R}_{>0}$ and $\eta \in \mathbb{R}_{>0}$ are design parameters which define how conservative is the avoidance action. Once $\lambda \leq \lambda_0$, the point p_r^b is subject to a force which moves it away from the p_o^b with an induced velocity $v_{(\mathcal{O},r)} \in \mathbb{R}^3$, expressed with respect to the frame O_b, whose entity is obtained by computing the gradient of $V_{\mathcal{O}}$, i.e.,

$$v_{(\mathcal{O},r)} = \begin{cases} \eta \left(\frac{1}{\lambda - \lambda_0}\right) \frac{1}{(\lambda - \lambda_0)^2} \frac{\partial \lambda}{\partial p_r^b}, & \text{if} \lambda \leq \lambda_0 \\ 0, & \text{if} \lambda > \lambda_0 \end{cases} \tag{10.11}$$

where $\frac{\partial \lambda}{\partial p_r^b} \in \mathbb{R}^3$ is the partial derivative vector of the distance between p_r^b and p_o^b and, for sake of simplicity, it can be approximated as the unit vector $\frac{\partial \lambda}{\partial p_r^b} = \frac{p_r^b - p_o^b}{\|p_r^b - p_o^b\|}$. In order to compute the set of joint velocities $\dot{q}_{(\mathcal{O},r)}$ which induce on p_r the velocity $v_{(\mathcal{O},r)}$, we use of Jacobian matrices.

In particular, assume that p_r^b is located on a certain link l, mounted on the k-th joint, the Jacobian associated to such a joint, computed in the point p_r^b is denoted as $J_k \in \mathbb{R}^{6 \times n}$ and given by

$$J_k(q, p_r) = \begin{bmatrix} J_k^{(1)}, & J_k^{(2)} & \cdots & J_k^{(k)} & 0_6 & \cdots & 0_6 \end{bmatrix}, \tag{10.12}$$

whose columns are computed as described in Chapter 4. In particular, assuming that modified DH is used, each column of (10.12) is computed as

$$J_k^{(i)} = \begin{bmatrix} z_i \times (p_r^b - p_i) \\ z_i \end{bmatrix}, \tag{10.13}$$

where z_i and p_i are retrieved from the forward kinematics as in Section 4.3.2.

With the objective of generating a position reference for the ISM controller, one could then design the avoidance reference as

$$q_{\text{av}} = q + \nu J_k^+ \begin{bmatrix} v_{(\mathcal{O},r)} \\ 0_3 \end{bmatrix}, \tag{10.14}$$

where $\nu \in \mathbb{R}_{>0}$ is a small design constant.

As depicted in Figure 10.6, the reference is provided by a standard ISM controller, like the one described in Section 2.4, to dominate possible small model mismatches and external disturbances acting on the manipulator.

10.4 Experiment and results

The obstacle avoidance scheme has been tested on the Franka Emika Panda robot, described in Appendix C. Computations were performed on an x86-64 system with an Intel Core i7 10400F CPU an NVidia RTX 3060 Ti GPU and 16 GB of RAM. As for the RGBD sensor, a Microsoft Xbox 360 Kinect, shown in Figure 10.7, has been employed. The sensor provides RGB data with a resolution of 640×480 pixels at a rate of 30 frames per second. It is also equipped with an infrared (IR) emitter and an IR depth sensor for depth measurements. The depth sensor has resolution and frame rate equal to those of the RGB camera. The depth sampling distance is ~ 1.5 mm at 50 cm from the sensor, while it is ~ 5 cm at 5 m.

Figure 10.7: Vision sensor employed for the experiment.

In order to train the YOLOv3 detector, a dataset of 2400 synthetic RGB images with a resolution of 640×480 pixels has been generated using 50 background images. Aiming to reduce the frequency of false positives, the background images are included in the training set. As for the training of the orientation estimator, the orientation space has been discretized in $k = 144$ parts and, for each part, 10 synthetic RGB images with a resolution of $n_{in} \times n_{in}$ pixels were generated, with $n_{in} = 128$. This operation resulted in a dataset of 1440 images, accompanied with their ground truth like depicted in Figure 10.4. Moreover, the size of the latent space vector z has been chosen as $L = 128$. Both the YOLOv3 and the CDA datasets have been divided in 70% training set and 30% validation set.

The efficacy of the HE calibration scheme, instrumental for the correct functioning of the obstacle avoidance scheme, has been evaluated. In particular, an RGB image and and a depth image are given as input to the scheme presented in Figure 10.1. The result of the estimation procedure are provided in Fig. 10.8. Such a figure contains three elements. The first one is the point cloud reconstructed using the depth data retrieved from the depth image, with the points belonging to the robot surface highlighted in green, while the rest have been colored in blue. The second element is the 3D model of the robot, in orange, positioned and oriented according to the value of the first guess $\hat{T}_b^c$. Finally, the red points represent the vertexes of the 3D model placed using the result of the ICP algorithm, i.e., T_b^c. The fact that the red points overlaps almost perfectly the point cloud associated to the real robot, allows us to conclude that the results proposed HE calibration scheme are satisfactory.

In order to assess the validity of the obstacle avoidance scheme, the robot is operated in presence of moving obstacles which perform unplanned movements. In particular, the robot is required to reach and maintain a desired configuration in the joint space, i.e.,

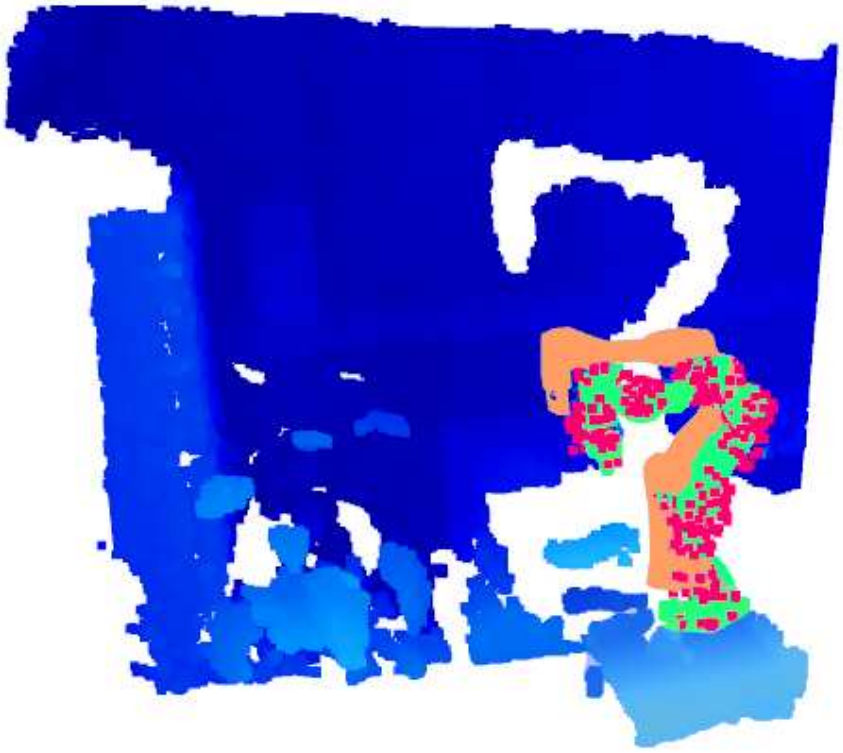


Figure 10.8: Result of the HE calibration process. Depth points from the camera are in blue and green. The orange 3D model represents the first pose estimate, while the red points the result of the ICP algorithm.

$q_d = \begin{bmatrix} 0 & -\frac{\pi}{4} & 0 & -\frac{3}{4}\pi & 0 & \frac{\pi}{2} & \frac{\pi}{4} \end{bmatrix}^\top$ [rad], while a human operator moves freely in its workspace. The avoidance parameters are chosen as $\eta = 0.025$ and $\lambda_0 = 0.3$. Some time instants of the experiments have been captured and presented in Figure 10.9. The manipulation volume is $\mathcal{M}^b = \emptyset$.

Moreover, the time evolution of the distance between the end effector, whose position is denoted as p_e^b, and its desired position $p_{e,d}^b \in \mathbb{R}^3$ (obtained from q_d), is presented in Figure 10.10. The same figure also depicts the time evolution of the minimum distance between the robot and the nearest obstacle point and the evolution of the value of a flag which represent the current state of the robot, i.e., reaching (flag equal to 0) or avoidance (flag equal to 1). From Figure 10.10, it is possible to see how the minimum distance between the end effector and the human operator never reaches a value smaller than 0.12 m, meaning the safety is ensured for all the duration of the experiment.

Figure 10.9: Instants of the experiment. In each frame, the point cloud $\mathcal{O}^c$ are colored in red and presented in the bottom left corner.

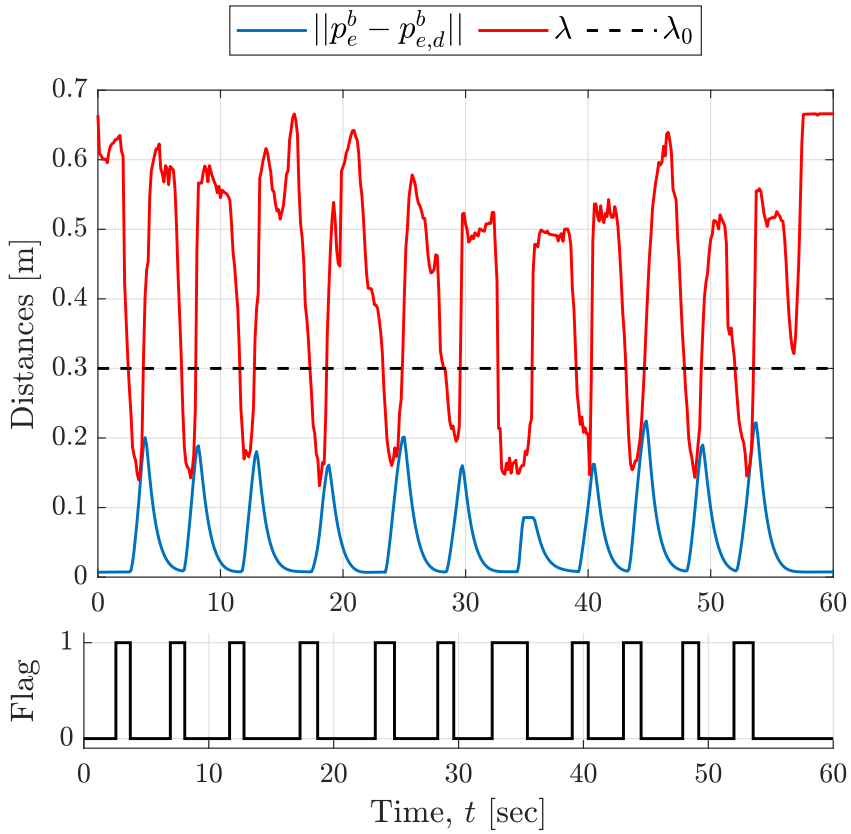


Figure 10.10: Time evolution of the distance between end-effector and target (blue), robot an human operator (red), and the state flag.

Chapter 11

Conclusions and Future Research

In this dissertation, novel control and observer design techniques that exploit the joint use of deep neural networks and sliding modes have been presented, theoretical analyzed and assessed both on simulation and experimentally. The dissertation, divided into four parts, is focused on different aspects of the research done in the last three years. What follows is a brief summary of what has been discussed.

In the first part, the main theoretical concepts about sliding modes, neural networks, and robotics, instrumental to understand of the strategies proposed in the rest of the dissertation, have been introduced.

The focus of the second part was on the design of a novel deep neural network based integral sliding mode control framework, referred to as DNN-ISM, whose aim is to stabilize on a desired trajectory the state of certain classes of perturbed nonlinear systems with fully unknown dynamics. Then, depending on the class of system, different types of constraints have been enforced, extending the DNN-ISM by integration of model predictive control or barrier function components. The control framework has been assessed in simulation on the Duffing Oscillator, the Double Tank and on the model of a complex robotic manipulator with 7 DoF. Moreover, more than satisfactorily experimental results have been obtained controlling a real Franka Emika Panda robot with the DNN-ISM control strategy.

The objective of the third part was to explore how neural networks and sliding modes could be exploited in the domain of fault detection. To this end, two different schemes have been proposed. The former relies on the DNN-ISM framework to design an unknown input observer whose aim is to provide an estimate of a fault acting on the input channel. The second propose a deep reinforcement learning agent that detects, isolates and compensates faults acting on the sensors of a 7 DoF manipulator. Then, a battery of second order sliding mode unknown input observers, which exploit the results of sensor faults compensation, are designed to estimate the fault acting on the actuators of the robot. Both the detection schemes have been assessed in simulation with good results.

In the fourth and final part of the dissertation, the problem of safe and ergonomic physical human-robot interaction has been addressed, proposing two different control strategies. The former relies on te DNN-ISM framework to solve the problem of ergonomic handover in the case in which the model of the robot is partially unknown and the dynamics of the object exchanged during the interaction is unavailable. The latter aims to avoid collisions between the robot and everything is in its surroundings. This has been done by implementing a vision system, whose initialization depends on the output of convolutional neural networks, which generates a reference that allows collision avoidance. Then, such a reference is followed relying on a sliding mode controller which also takes into account possible model mismatches or disturbances. Both architectures have been assessed experimentally on the Franka Emika Panda robot.

To conclude, all the proposed methodologies have been theoretically analyzed. Moreover, the theoretical results are validated by means of simulations and experiments, demonstrating the implementability of the proposed architectures in real-world applications.

11.1 Future Research

Despite the validity of the approaches described in this dissertation, there are some aspects that needs to be addressed in the future.

Enlargement of the class of systems Especially in the constrained case, the DNN-ISM control framework is applied to a very specific class of systems, restricting its range of application. One of the main goals of future research is to extend the class of system, and hence the efficacy of the proposed methodology.

Extension of the class of disturbances At the moment, only matched disturbances are affecting the system. Future directions include the study of the behavior of the system in presence of unmatched disturbances, especially in the constrained scenario.

Enhancement of the DNN based MPC/ISM In its current form, the DNN based MPC/ISM control architecture presents a criticality: the MPC component is not utilized during the adaptation transient. To cope with this aspect, in the future the architecture is going to be extended with robust MPC methodologies.

Introduction of stochasticity At the moment, the behavior of the system does not include stochastic components. In the future, the robustness properties of the DNN-ISM against stochastic processes such as sensor noise or environmental variability will be considered.

Design of DNN based sensor fault diagnosis schemes with guarantees The sensor fault diagnosis scheme presented in this dissertation relies on the use of DRL.

Despite the goodness of the results, the quality of the diagnosis is strictly related to the quality of the training, without any theoretical guarantees. An objective for the future research is to provide some guarantees on the fault approximation error.

Application of DNN based MPC/ISM to collision avoidance An important achievement would be to apply the MPC/ISM architecture with DNN on a real robotic manipulator to perform real-time collision avoidance with theoretical guarantees.

Appendices

Appendix A

Parameter Projection

The *parameter projection* operator is a widely adopted tool in classic adaptive control theory [89]. Specifically, it is particularly useful when some knowledge of the optimal parameters (e.g., the bound of their norm or their sign) is available.

Assume that the optimal parameters of an adaptive controller are collected in a vector $\theta \in \mathbb{R}^p$. Then, one could exploit some available information about θ to describe a suitable domain $\mathcal{B}_\theta \subset \mathbb{R}^p$ in the parameter space, with the objective to keep the estimates of θ, denoted by the vector $\hat{\theta} \in \mathbb{R}^p$, in such a set.

Let $\mathcal{P} : \mathbb{R}^p \to \mathbb{R}$ be a smooth function of the parameter vector. Then, the domain $\mathcal{B}_\theta$ can be defined as the sub-level set of $\mathcal{P}$ as

$$\mathcal{B}_\theta \coloneqq \left\{ \hat{\theta} \in \mathbb{R}^p : \mathcal{P}(\hat{\theta}) \leq 0 \right\}, \tag{A.1}$$

with $\mathcal{B}_\theta^{\circ} \subset \mathcal{B}_\theta$ and $\partial\mathcal{B}_\theta \subset \mathcal{B}_\theta$ denoting its interior and boundary, respectively. Since $\partial\mathcal{B}_\theta$ is the level hyper-surface $\mathcal{P}(\hat{\theta}) = 0$, the gradient $\nabla_{\hat{\theta}}\mathcal{P} = \frac{\partial \mathcal{P}}{\partial \hat{\theta}} \in \mathbb{R}^p$ represents the outward-pointing normal vector to $\partial\mathcal{B}_\theta$. Then, it is possible to define the standard projection operator [89, Apendix E] as

$$\underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau) = \begin{cases} \tau & \text{if } \hat{\theta} \in \mathcal{B}_\theta{}^{\circ} \text{ or } \nabla_{\hat{\theta}}\mathcal{P} \cdot \tau \leq 0, \\ \left(I_p - \Gamma \frac{\nabla_{\hat{\theta}}\mathcal{P} \nabla_{\hat{\theta}}\mathcal{P}^\top}{\nabla_{\hat{\theta}}\mathcal{P}^\top \Gamma \nabla_{\hat{\theta}}\mathcal{P}} \right) \tau & \text{if } \hat{\theta} \in \partial\mathcal{B}_\theta \text{ and } \nabla_{\hat{\theta}}\mathcal{P} \cdot \tau > 0, \end{cases} \tag{A.2}$$

where $\tau \in \mathbb{R}^p$, while $\Gamma \in \mathbb{R}^{p \times p}$ is a positive symmetric matrix. Note that, even though the operator is a function of the arguments $(\tau, \hat{\theta}, \Gamma)$, for sake of readability it is more convenient to explicit just τ.

The meaning of (A.2) is straightforward if one considers $\hat{\theta}$ as the current estimate of the parameters, and τ as its "speed". The $\underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau)$ acts on the vector τ, leaving it unchanged if it $\hat{\theta}$ is inside $\mathcal{B}_\theta$ (i.e., $\hat{\theta} \in \mathcal{B}_\theta{}^{\circ}$) or if τ is pointing outside the set (i.e., $\nabla_{\hat{\theta}}\mathcal{P} \cdot \tau \leq 0$). In fact, in such cases, an ideally continuous-time update τ would not cause $\hat{\theta}$ to go out of $\mathcal{B}_\theta$. On the contrary, if $\hat{\theta} \in \partial\mathcal{B}_\theta$ and τ points outside the set (i.e., $\nabla_{\hat{\theta}}\mathcal{P} \cdot \tau > 0$), the operator $\underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau)$ returns the projection of vector τ on the hyperplane that is tangent to $\partial\mathcal{B}_\theta$ in $\hat{\theta}$.

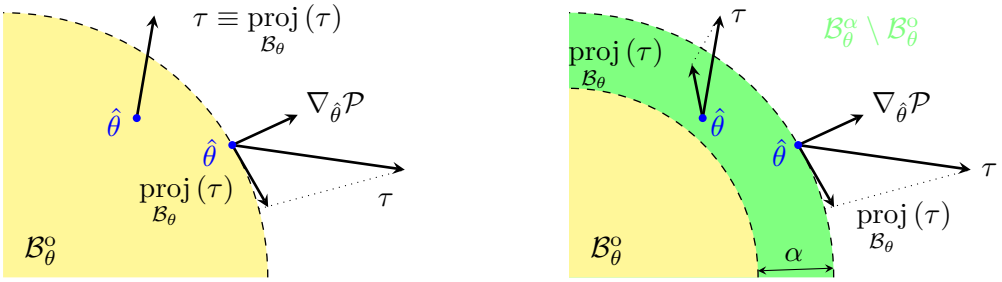
The projection operator defined as in (A.2) usually lead to a discontinuous evolution of τ, violating the the Lipschitz continuity condition and preventing the application of standard existence theorems [89]. For this reason, (A.2) can be modified to make it smoother. In particular, defining a $O(\alpha)$-boundary layer around $\mathcal{B}_\theta$, it is possible to enlarge this last creating a set, denoted as $\mathcal{B}_\theta^\alpha$, which is the result of the union of the boundary layer and $\mathcal{B}_\theta$, i.e.,

$$\mathcal{B}_\theta^\alpha := \{\hat{\theta} \in \mathbb{R}^p : \mathcal{P}(\hat{\theta}) \leq \alpha\}. \tag{A.3}$$

Then, a smooth version of the projection operator can be defined [89, Apendix E] as

$$\underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau) = \begin{cases} \tau & \text{if } \hat{\theta} \in \mathcal{B}_\theta{}^{\text{o}} \text{ or } \nabla_{\hat{\theta}}\mathcal{P} \cdot \tau \leq 0, \\ \left(I_p - c(\hat{\theta})\Gamma \frac{\nabla_{\hat{\theta}}\mathcal{P}\nabla_{\hat{\theta}}\mathcal{P}^\top}{\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P}}\right)\tau & \text{if } \hat{\theta} \in \mathcal{B}_\theta^\alpha \setminus \mathcal{B}_\theta{}^{\text{o}} \text{ and } \nabla_{\hat{\theta}}\mathcal{P} \cdot \tau > 0, \end{cases} \tag{A.4}$$

where $c : \mathbb{R}^p \to [0, 1]$ is any smooth function such that $c(\partial\mathcal{B}_\theta) = 0$ and $c(\partial\mathcal{B}_\theta^\alpha) = 1$. Meaning that the trajectory of $\hat{\theta}$ is diverted progressively on the α-layer, achieving the full projection on the tangent hyperplane on $\partial\mathcal{B}_\theta^\alpha$. As a consequence, the boundedness is guaranteed on $\mathcal{B}_\theta^\alpha$ instead of $\mathcal{B}_\theta$, as it happens when the operator in (A.2) is used. For the reader's convenience, a visualization of the standard and smooth projection operators is provided in Figure A.1.



(a) Standard projection operator (A.2).

(b) Smooth projection operator (A.4).

Figure A.1: Graphical representation of the projection operators defined in (A.2) and (A.4) in the case $\theta \in \mathbb{R}^2$.

Lemma A.1 (Smooth Projection Properties). *Consider the smooth projection operator in* (A.4)*, with* $\mathcal{B}_\theta$ *and* $\mathcal{B}_\theta^\alpha$ *defined as in* (A.1) *and* (A.3)*, respectively. Then, the following properties are true.*

1. *the mapping* $\underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau, \hat{\theta}, \Gamma\right) : \mathbb{R}^p \times \mathcal{B}_\theta^\alpha \times \mathbb{R}^{p\times p} \to \mathbb{R}^p$ *is locally Lipschitz in its arguments;*
2. $\underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau)^\top \Gamma^{-1} \underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau) \leq \tau^\top\Gamma^{-1}\tau$*, for all* $\hat{\theta} \in \mathcal{B}_\theta$*;*
3. *if* $\Gamma(t)$ *and* $\tau(t)$ *are continuously differentiable, and* $\dot{\hat{\theta}} = \underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau)$*, with* $\hat{\theta}(t_0) \in \mathcal{B}_\theta^\alpha$*, then* $\hat{\theta}(t) \in \mathcal{B}_\theta^\alpha$*, for* $t \geq t_0$*;*
4. $-(\theta - \hat{\theta})^\top\Gamma^{-1} \underset{\mathcal{B}_\theta}{\text{proj}}\,(\tau) \leq -(\theta - \hat{\theta})^\top\Gamma^{-1}\tau$*, for all* $\theta \in \mathcal{B}_\theta$ *and* $\hat{\theta} \in \mathcal{B}_\theta^\alpha$*.*

Proof. The proof of each property is developed separately.

1. Even though the proof its straightforward, it is long to develop and thus the reader should refer to [89, Appendix E].

2. By definition of the operator in (A.4), one has that, if it holds $\hat{\theta} \in \mathcal{B}_\theta{}^{\circ}$ or $\nabla_{\hat{\theta}}\mathcal{P}\cdot\tau \leq 0$, the equality trivially holds because the operator corresponds to the identity function. In the second case, i.e., the case in which $\hat{\theta} \in \mathcal{B}_\theta^\alpha \setminus \mathcal{B}_\theta{}^{\circ}$ and $\nabla_{\hat{\theta}}\mathcal{P}\cdot\tau > 0$, it holds that

$$\begin{aligned}\underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau\right)^\top \Gamma^{-1} \underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau\right) &= \tau^\top\Gamma^{-1}\tau - 2c(\hat{\theta})\frac{\left(\nabla_{\hat{\theta}}\mathcal{P}^\top\tau\right)^2}{\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P}} + c(\hat{\theta})\frac{\left\|\nabla_{\hat{\theta}}\mathcal{P}\nabla_{\hat{\theta}}\mathcal{P}^\top\tau\right\|^2}{(\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P})^2} \\ &= \tau^\top\Gamma^{-1}\tau - c(\hat{\theta})\left(2 - c(\hat{\theta})\right)\frac{\left(\nabla_{\hat{\theta}}\mathcal{P}^\top\tau\right)^2}{\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P}}\end{aligned}$$

   Since Γ is a symmetric positive definite matrix and $c(\hat{\theta}) \in [0,1]$, then the second therm is negative. Hence, the following is true

$$\underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau\right)^\top \Gamma^{-1} \underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau\right) \leq \tau^\top\Gamma^{-1}\tau.$$

3. Relying on the definition of (A.4), one has that

$$\begin{aligned}\nabla_{\hat{\theta}}\mathcal{P}^\top \underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau\right) &= \begin{cases}\nabla_{\hat{\theta}}\mathcal{P}^\top\tau \\ \nabla_{\hat{\theta}}\mathcal{P}^\top\left(I_p - c(\hat{\theta})\Gamma\frac{\nabla_{\hat{\theta}}\mathcal{P}\nabla_{\hat{\theta}}\mathcal{P}^\top}{\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P}}\right)\tau\end{cases} \\ &= \begin{cases}\nabla_{\hat{\theta}}\mathcal{P}^\top\tau \\ \left(\nabla_{\hat{\theta}}\mathcal{P}^\top - c(\hat{\theta})\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\frac{\nabla_{\hat{\theta}}\mathcal{P}\nabla_{\hat{\theta}}\mathcal{P}^\top}{\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P}}\right)\tau\end{cases} \\ &= \begin{cases}\nabla_{\hat{\theta}}\mathcal{P}^\top\tau \\ \left(\nabla_{\hat{\theta}}\mathcal{P}^\top - c(\hat{\theta})\frac{\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P}}{\nabla_{\hat{\theta}}\mathcal{P}^\top\Gamma\nabla_{\hat{\theta}}\mathcal{P}}\nabla_{\hat{\theta}}\mathcal{P}^\top\right)\tau\end{cases} \\ &= \begin{cases}\nabla_{\hat{\theta}}\mathcal{P}^\top\tau \\ \left(\nabla_{\hat{\theta}}\mathcal{P}^\top - c(\hat{\theta})\nabla_{\hat{\theta}}\mathcal{P}^\top\right)\tau\end{cases}\end{aligned}$$

   Rearranging the terms, one can rewrite the last equation as

$$\nabla_{\hat{\theta}}\mathcal{P}^\top \underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau\right) = \begin{cases}\nabla_{\hat{\theta}}\mathcal{P}^\top\tau & \text{if } \hat{\theta} \in \mathcal{B}_\theta{}^{\circ} \text{ or } \nabla_{\hat{\theta}}\mathcal{P}\cdot\tau \leq 0, \\ \left(1 - c(\hat{\theta})\right)\nabla_{\hat{\theta}}\mathcal{P}^\top\tau & \text{if } \hat{\theta} \in \mathcal{B}_\theta^\alpha \setminus \mathcal{B}_\theta{}^{\circ} \text{ and } \nabla_{\hat{\theta}}\mathcal{P}\cdot\tau > 0\end{cases}$$

   Let $\dot{\hat{\theta}}(t) = \underset{\mathcal{B}_\theta}{\text{proj}}\left(\tau\right)$, with $\hat{\theta}(t_0) \in \mathcal{B}_\theta^\alpha$. Then, the projection operator dictates the direction of the movement $\hat{\theta}(t)$. To study the boundedness in $\mathcal{B}_\theta^\alpha$ of $\hat{\theta}(t)$, it is sufficient to analyze the second case of the above equation, which is the one in which there is the risk for $\hat{\theta}(t)$ of going outside $\mathcal{B}_\theta^\alpha$. In the worst case, $\hat{\theta}(t) \in \partial\mathcal{B}_\theta^\alpha$ and $c(\hat{\theta}) = 1$, meaning that the vector $\dot{\hat{\theta}}(t)$ is orthogonal to the outgoing normal $\nabla_{\hat{\theta}}\mathcal{P}$, implying the boundedness of $\hat{\theta}(t)$ in $\mathcal{B}_\theta^\alpha$, for $t \geq t_0$.

4. Define the quantity $\tilde{\theta} = \theta - \hat{\theta}$, representing an estimation error. Then, exploiting the definition (A.4), one has that

$$-\tilde{\theta}^\top \Gamma^{-1} \underset{\mathcal{B}_\theta}{\text{proj}}(\tau) = \begin{cases} -\tilde{\theta}^\top \Gamma^{-1} \tau \\ -\tilde{\theta}^\top \Gamma^{-1} \left(I_p - c(\hat{\theta}) \Gamma \frac{\nabla_{\hat{\theta}} \mathcal{P} \nabla_{\hat{\theta}} \mathcal{P}^\top}{\nabla_{\hat{\theta}} \mathcal{P}^\top \Gamma \nabla_{\hat{\theta}} \mathcal{P}} \right) \tau \end{cases}$$
$$= \begin{cases} -\tilde{\theta}^\top \Gamma^{-1} \tau \\ -\left(\tilde{\theta}^\top \Gamma^{-1} - c(\hat{\theta}) \tilde{\theta}^\top \Gamma^{-1} \Gamma \frac{\nabla_{\hat{\theta}} \mathcal{P} \nabla_{\hat{\theta}} \mathcal{P}^\top}{\nabla_{\hat{\theta}} \mathcal{P}^\top \Gamma \nabla_{\hat{\theta}} \mathcal{P}} \right) \tau \end{cases}$$

Then, rearranging the terms one has that

$$-\tilde{\theta}^\top \Gamma^{-1} \underset{\mathcal{B}_\theta}{\text{proj}}(\tau) = -\tilde{\theta} \Gamma^{-1} \tau + \begin{cases} 0 \\ c(\hat{\theta}) \frac{(\tilde{\theta}^\top \nabla_{\hat{\theta}} \mathcal{P}) \nabla_{\hat{\theta}} \mathcal{P}^\top}{\nabla_{\hat{\theta}} \mathcal{P}^\top \Gamma \nabla_{\hat{\theta}} \mathcal{P}} \tau \end{cases}$$

In the first case, the inequality trivially holds. As for the second case, i.e., the one in which it holds $\hat{\theta} \in \mathcal{B}_\theta^\alpha \setminus \mathcal{B}_\theta{}^\circ$ and $\nabla_{\hat{\theta}} \mathcal{P} \cdot \tau > 0$, then it is possible to make the following considerations. First, by virtue of the hypothesis on θ and $\hat{\theta}$, and from the fact that $\mathcal{B}_\theta^\alpha \setminus \mathcal{B}_\theta{}^\circ$ is a α-contour of $\mathcal{B}_\theta$, it means that the vector $\tilde{\theta}$ points toward the interior of $\mathcal{B}_\theta^\alpha$, implying

$$\tilde{\theta}^\top \nabla_{\hat{\theta}} \mathcal{P} \leq 0.$$

Hence, it holds that

$$-\tilde{\theta}^\top \Gamma^{-1} \underset{\mathcal{B}_\theta}{\text{proj}}(\tau) \leq -\tilde{\theta}^\top \Gamma^{-1} \tau.$$

□

Appendix B

Proofs

B.1 Proof of Theorem 5.1

Consider the Lyapunov-like candidate function $v : \mathcal{X} \to \mathbb{R}$ defined as

$$v = \frac{1}{2}\sigma^\top\sigma + \frac{1}{2}\sum_{j=0}^{k_\Phi} \mathrm{vec}\left(\tilde{V}_j\right)^\top \Gamma_{\Phi_j}^{-1}\mathrm{vec}\left(\tilde{V}_j\right) + \frac{1}{2}\sum_{j=0}^{k_\Psi} \mathrm{vec}\left(\tilde{U}_j\right)^\top \Gamma_{\Psi_j}^{-1}\mathrm{vec}\left(\tilde{U}_j\right), \tag{B.1}$$

where σ is the integral sliding variable defined as in (5.39). The above function is characterized by a first time-derivative equal to

$$\dot{v} = \sigma^\top\dot{\sigma} + \sum_{j=0}^{k_\Phi} \mathrm{vec}\left(\tilde{V}_j\right)^\top \Gamma_{\Phi_j}^{-1}\mathrm{vec}\left(\dot{\tilde{V}}_j\right) + \sum_{j=0}^{k_\Psi} \mathrm{vec}\left(\tilde{U}_j\right)^\top \Gamma_{\Psi_j}^{-1}\mathrm{vec}\left(\dot{\tilde{U}}_j\right). \tag{B.2}$$

The expression of $\dot{\sigma}$ can be obtained by computing the first time derivative of σ_0 in (5.5) and substituting the value of $\dot{\hat{z}}$ in (5.43) as

$$\begin{aligned}
\dot{\sigma} &= \dot{\sigma}_0 - \dot{\hat{z}} \\
&= C_1(\dot{x}_1 - \dot{x}_1^\star) + C_2(\dot{x}_2 - \dot{x}_2^\star) - C_1\left(\hat{\Phi}_{k_\Phi}^{[1]} - \dot{x}_1^\star\right) - C_2\left(\hat{\Phi}_{k_\Phi}^{[2]} + \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,i} - \dot{x}_2^\star\right) \\
&= C_1\left(\dot{x}_1 - \hat{\Phi}_{k_\Phi}^{[1]}\right) + C_2\left(\dot{x}_2 - \hat{\Phi}_{k_\Phi}^{[2]} - \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,i}\right) \\
&= C_1\left(f_1 - \hat{\Phi}_{k_\Phi}^{[1]}\right) + C_2\left(f_2 + \bar{B}u + h - \hat{\Phi}_{k_\Phi}^{[2]} - \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,i}\right).
\end{aligned}$$

Then, substituting the control law (5.8) and the ideal approximation of the dynamics in (5.14), it holds that

$$\begin{aligned}
\dot{\sigma} &= C_1\left(f_1 - \hat{\Phi}_{k_\Phi}^{[1]}\right) + C_2\left(f_2 + \bar{B}u + h - \hat{\Phi}_{k_\Phi}^{[2]} - \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,i}\right) \\
&= C_1\left(\Phi_{k_\Phi}^{[1]} + \varepsilon_\Phi^{[1]} - \hat{\Phi}_{k_\Phi}^{[1]}\right) + C_2\Bigg(\Phi_{k_\Phi}^{[2]} + \varepsilon_\Phi^{[2]} + \sum_{i=1}^{m}\Psi_{k_\Psi}^{[i]}u_{n,i} + \varepsilon_\Psi u_n +
\end{aligned}$$

$$+ \bar{B}u_r + h - \hat{\Phi}^{[2]}_{k_\Phi} - \sum_{i=1}^{m} \hat{\Psi}^{[i]}_{k_\Psi} u_{n,i} \Bigg).$$

Recalling that $\tilde{\Phi}^{[p]}_{k_\Phi} = \Phi^{[p]}_{k_\Phi} - \hat{\Phi}^{[p]}_{k_\Phi}$ and that $\tilde{\Psi}^{[i]}_{k_\Psi} = \Phi^{[i]}_{k_\Psi} - \hat{\Psi}^{[i]}_{k_\Psi}$, if one rearranges the terms the first time-derivative of the sliding variable is

$$\dot{\sigma} = C_1 \left(\tilde{\Phi}^{[1]}_{k_\Phi} + \varepsilon^{[1]}_{\Phi} \right) + C_2 \left(\tilde{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \tilde{\Psi}^{[i]}_{k_\Psi} u_{n,i} + \bar{B}u_r + \varepsilon^{[2]}_{\Phi} + \varepsilon_\Psi u_n + h \right). \tag{B.3}$$

Since $\tilde{V}_j = V_j - \hat{V}_j$ and $\tilde{U}_j = U_j - \hat{U}_j$, with V_j and U_j constant, it holds that $\dot{\tilde{V}}_j = -\dot{\hat{V}}_j$ and $\dot{\tilde{U}}_j = -\dot{\hat{U}}_j$. Hence, substituting (B.3) in (B.2) leads to

$$\begin{aligned}
\dot{v} = \sigma^\top & \left[C_1 \left(\tilde{\Phi}^{[1]}_{k_\Phi} + \varepsilon^{[1]}_{\Phi} \right) + C_2 \left(\tilde{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \tilde{\Psi}^{[i]}_{k_\Psi} u_{n,i} + \bar{B}u_r + \varepsilon^{[2]}_{\Phi} + \varepsilon_\Psi u_n + h \right) \right] + \\
& - \sum_{j=0}^{k_\Phi} \mathrm{vec}\left(\tilde{V}_j\right)^\top \Gamma^{-1}_{\Phi_j} \mathrm{vec}\left(\dot{\hat{V}}_j\right) - \sum_{j=0}^{k_\Psi} \mathrm{vec}\left(\tilde{U}_j\right)^\top \Gamma^{-1}_{\Psi_j} \mathrm{vec}\left(\dot{\hat{U}}_j\right).
\end{aligned}$$

Expanding the last two terms to separate the last layer from the inner ones, one has that

$$\begin{aligned}
\dot{v} = \sigma^\top & \left[C_1 \left(\tilde{\Phi}^{[1]}_{k_\Phi} + \varepsilon^{[1]}_{\Phi} \right) + C_2 \left(\tilde{\Phi}^{[2]}_{k_\Phi} + \sum_{i=1}^{m} \tilde{\Psi}^{[i]}_{k_\Psi} u_{n,i} + \bar{B}u_r + \varepsilon^{[2]}_{\Phi} + \varepsilon_\Psi u_n + h \right) \right] + \\
& - \sum_{j=0}^{k_\Phi - 1} \mathrm{vec}\left(\tilde{V}_j\right)^\top \Gamma^{-1}_{\Phi_j} \mathrm{vec}\left(\dot{\hat{V}}_j\right) - \sum_{p=1}^{2} \mathrm{vec}\left(\tilde{V}^{[p]}_{k_\Phi}\right)^\top \left(\Gamma^{[p]}_{\Phi_{k_\Phi}}\right)^{-1} \mathrm{vec}\left(\dot{\hat{V}}^{[p]}_{k_\Phi}\right) + \\
& - \sum_{j=0}^{k_\Psi - 1} \mathrm{vec}\left(\tilde{U}_j\right)^\top \Gamma^{-1}_{\Psi_j} \mathrm{vec}\left(\dot{\hat{U}}_j\right) - \sum_{i=1}^{m} \mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top \Gamma^{-1}_{\Psi_{k_\Psi}} \mathrm{vec}\left(\dot{\hat{U}}^{[i]}_{k_\Psi}\right).
\end{aligned} \tag{B.4}$$

Then, substituting $\tilde{\Phi}^{[1]}_{k_\Phi}$, $\tilde{\Phi}^{[2]}_{k_\Phi}$ and $\tilde{\Psi}^{[i]}_{k_\Psi}$ with (5.29a), (5.29b), and (5.38), respectively, it holds that

$$\begin{aligned}
\dot{v}(x) = \sigma^\top C_1 & \left\{ \Lambda^{[1]}_{\Phi_{k_\Phi}} \mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right) + \Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=0}^{k_\Phi - 1} \Lambda^{[1]}_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi - 1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \right\} + \\
& + \sigma^\top C_2 \left\{ \Lambda^{[2]}_{\Phi_{k_\Phi}} \mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi}\right) + \Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=0}^{k_\Phi - 1} \Lambda^{[2]}_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi - 1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \right. \\
& + \sum_{i=1}^{m} \left(\Lambda^{[i]}_{\Psi_{k_\Psi}} \mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right) + \Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=0}^{k_\Psi - 1} \Lambda^{[i]}_{\Psi_j} \mathrm{vec}\left(\tilde{U}_j\right) + \sum_{j=1}^{k_\Psi - 1} \Xi^{[i]}_{\Psi_j} \Delta_{\Psi_j} \right) u_{n,i} + \\
& \left. + \bar{B}u_r + \varepsilon_\Psi u_n + h \right\} - \sum_{j=0}^{k_\Phi - 1} \mathrm{vec}\left(\tilde{V}_j\right)^\top \Gamma^{-1}_{\Phi_j} \mathrm{vec}\left(\dot{\hat{V}}_j\right) + \\
& - \mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right)^\top \left(\Gamma^{[1]}_{\Phi_{k_\Phi}}\right)^{-1} \mathrm{vec}\left(\dot{\hat{V}}^{[1]}_{k_\Phi}\right) - \mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi}\right)^\top \left(\Gamma^{[2]}_{\Phi_{k_\Phi}}\right)^{-1} \mathrm{vec}\left(\dot{\hat{V}}^{[2]}_{k_\Phi}\right) + \\
& - \sum_{j=0}^{k_\Psi - 1} \mathrm{vec}\left(\tilde{U}_j\right)^\top \Gamma^{-1}_{\Psi_j} \mathrm{vec}\left(\dot{\hat{U}}_j\right) - \sum_{i=1}^{m} \mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top \Gamma^{-1}_{\Psi_{k_\Psi}} \mathrm{vec}\left(\dot{\hat{U}}^{[i]}_{k_\Psi}\right).
\end{aligned} \tag{B.5}$$

For convenience, it is possible separate the terms so that it is easier to identify the ones that must be compensated, obtaining

$$\dot{v} = \sigma^\top C_1 \left\{ \Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \right\} + \sigma^\top C_2 \left\{ \Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \right.$$

$$+\sum_{i=1}^{m}\left(\Delta_{\Psi_{k_\Psi}}^{[i]}+\sum_{j=1}^{k_\Psi-1}\Xi_{\Psi_j}^{[i]}\Delta_{\Psi_j}\right)u_{n,i}+\bar{B}u_r+\varepsilon_\Psi u_n+h\Bigg\}+\sigma^\top C_1\Lambda_{\Phi_{k_\Phi}}^{[1]}\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right)+$$
$$+\sigma^\top C_1\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}^{[1]}\operatorname{vec}\left(\tilde{V}_j\right)+\sigma^\top C_2\Lambda_{\Phi_{k_\Phi}}^{[2]}\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)+\sigma^\top C_2\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}^{[2]}\operatorname{vec}\left(\tilde{V}_j\right)+$$
$$+\sigma^\top C_2\sum_{i=1}^{m}\left(\Lambda_{\Psi_{k_\Psi}}^{[i]}\operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)\right)u_{n,i}+\sigma^\top C_2\sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1}\Lambda_{\Psi_j}^{[i]}\operatorname{vec}\left(\tilde{U}_j\right)\right)u_{n,i}+$$
$$-\sum_{j=0}^{k_\Phi-1}\operatorname{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\operatorname{vec}\left(\dot{\hat{V}}_j\right)-\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right)^\top\left(\Gamma_{\Phi_{k_\Phi}}^{[1]}\right)^{-1}\operatorname{vec}\left(\dot{\hat{V}}_{k_\Phi}^{[1]}\right)+$$
$$-\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)^\top\left(\Gamma_{\Phi_{k_\Phi}}^{[2]}\right)^{-1}\operatorname{vec}\left(\dot{\hat{V}}_{k_\Phi}^{[2]}\right)-\sum_{j=0}^{k_\Psi-1}\operatorname{vec}\left(\tilde{U}_j\right)^\top\Gamma_{\Psi_j}^{-1}\operatorname{vec}\left(\dot{\hat{U}}_j\right)+$$
$$-\sum_{i=1}^{m}\operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)^\top\Gamma_{\Psi_{k_\Psi}}^{-1}\operatorname{vec}\left(\dot{\hat{U}}_{k_\Psi}^{[i]}\right). \tag{B.6}$$

With the aim of compensating the terms that depend on the weight estimation errors $\tilde{V}_j$, $\tilde{V}_{k_\Phi}^{[1]}$, $\tilde{V}_{k_\Phi}^{[2]}$, $\tilde{U}_j$, and $\tilde{U}_{k_\Psi}^{[i]}$, one can select the weight adaptation laws $\dot{\hat{V}}_j$, $\dot{\hat{V}}_{k_\Phi}^{[1]}$, $\dot{\hat{V}}_{k_\Phi}^{[2]}$, $\dot{\hat{U}}_j$, and $\dot{\hat{U}}_{k_\Psi}^{[i]}$ as in (5.45), (5.47a), (5.47b), (5.48), and (5.49), respectively, obtaining

$$\dot{v}=\sigma^\top C_1\Bigg\{\Delta_{\Phi_{k_\Phi}}^{[1]}+\sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}^{[1]}\Delta_{\Phi_j}+\varepsilon_\Phi^{[1]}\Bigg\}+\sigma^\top C_2\Bigg\{\Delta_{\Phi_{k_\Phi}}^{[2]}+\sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}^{[2]}\Delta_{\Phi_j}+\varepsilon_\Phi^{[2]}+$$
$$+\sum_{i=1}^{m}\left(\Delta_{\Psi_{k_\Psi}}^{[i]}+\sum_{j=1}^{k_\Psi-1}\Xi_{\Psi_j}^{[i]}\Delta_{\Psi_j}\right)u_{n,i}+\bar{B}u_r+\varepsilon_\Psi u_n+h\Bigg\}+\sigma^\top C_1\Lambda_{\Phi_{k_\Phi}}^{[1]}\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right)+$$
$$+\sigma^\top C_1\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}^{[1]}\operatorname{vec}\left(\tilde{V}_j\right)+\sigma^\top C_2\Lambda_{\Phi_{k_\Phi}}^{[2]}\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)+\sigma^\top C_2\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}^{[2]}\operatorname{vec}\left(\tilde{V}_j\right)+$$
$$+\sigma^\top C_2\sum_{i=1}^{m}\left(\Lambda_{\Psi_{k_\Psi}}^{[i]}\operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)\right)u_{n,i}+\sigma^\top C_2\sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1}\Lambda_{\Psi_j}^{[i]}\operatorname{vec}\left(\tilde{U}_j\right)\right)u_{n,i}+$$
$$-\sum_{j=0}^{k_\Phi-1}\operatorname{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\operatorname{proj}_{\mathcal{B}_{\Phi_j}}\left(\Gamma_{\Phi_j}\Lambda_{\Phi_j}^\top C^\top\sigma\right)+$$
$$-\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right)^\top\left(\Gamma_{\Phi_{k_\Phi}}^{[1]}\right)^{-1}\operatorname{proj}_{\mathcal{B}_{\Phi_{k_\Phi}}}\left(\Gamma_{\Phi_{k_\Phi}}^{[1]}(\Lambda_{\Phi_{k_\Phi}}^{[1]})^\top C_1^\top\sigma\right)+$$
$$-\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)^\top\left(\Gamma_{\Phi_{k_\Phi}}^{[2]}\right)^{-1}\operatorname{proj}_{\mathcal{B}_{\Phi_{k_\Phi}}}\left(\Gamma_{\Phi_{k_\Phi}}^{[2]}(\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_2^\top\sigma\right)+$$
$$-\sum_{j=0}^{k_\Psi-1}\operatorname{vec}\left(\tilde{U}_j\right)^\top\Gamma_{\Psi_j}^{-1}\operatorname{proj}_{\mathcal{B}_{\Psi_j}}\left(\Gamma_{\Psi_j}\left(\sum_{i=1}^{m}u_{n,i}(\Lambda_{\Psi_j}^{[i]})^\top\right)C_2^\top\sigma\right)+$$
$$-\sum_{i=1}^{m}\operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)^\top\Gamma_{\Psi_{k_\Psi}}^{-1}\operatorname{proj}_{\mathcal{B}_{\Psi_{k_\Psi}}}\left(\Gamma_{\Psi_{k_\Psi}}u_{n,i}(\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top C_2^\top\sigma\right). \tag{B.7}$$

Then, exploiting points 3 and 4 of Lemma 5.2, the above equality is transformed into

$$\begin{aligned}\dot{v} \leq \sigma^\top C_1 \Bigg\{ \Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \Bigg\} + \sigma^\top C_2 \Bigg\{ \Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \\ + \sum_{i=1}^{m} \left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1} \Xi^{[i]}_{\Psi_j} \Delta_{\Psi_j} \right) u_{n,i} + \bar{B} u_r + \varepsilon_\Psi u_n + h \Bigg\} + \sigma^\top C_1 \Lambda^{[1]}_{\Phi_{k_\Phi}} \mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right) + \\ + \sigma^\top C_1 \sum_{j=0}^{k_\Phi-1} \Lambda^{[1]}_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j \right) + \sigma^\top C_2 \Lambda^{[2]}_{\Phi_{k_\Phi}} \mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right) + \sigma^\top C_2 \sum_{j=0}^{k_\Phi-1} \Lambda^{[2]}_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j \right) + \\ + \sigma^\top C_2 \sum_{i=1}^{m} \left(\Lambda^{[i]}_{\Psi_{k_\Psi}} \mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi} \right) \right) u_{n,i} + \sigma^\top C_2 \sum_{i=1}^{m} \left(\sum_{j=0}^{k_\Psi-1} \Lambda^{[i]}_{\Psi_j} \mathrm{vec}\left(\tilde{U}_j \right) \right) u_{n,i} + \\ - \sum_{j=0}^{k_\Phi-1} \mathrm{vec}\left(\tilde{V}_j \right)^\top \Lambda^\top_{\Phi_j} C^\top \sigma - \mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right)^\top (\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top C_1^\top \sigma - \mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right)^\top (\Lambda^{[2]}_{\Phi_{k_\Phi}})^\top C_2^\top \sigma + \\ - \sum_{j=0}^{k_\Psi-1} \mathrm{vec}\left(\tilde{U}_j \right)^\top \left(\sum_{i=1}^{m} u_{n,i} (\Lambda^{[i]}_{\Psi_j})^\top \right) C_2^\top \sigma - \sum_{i=1}^{m} \mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi} \right)^\top u_{n,i} (\Lambda^{[i]}_{\Psi_{k_\Psi}})^\top C_2^\top \sigma. \quad \text{(B.8)}\end{aligned}$$

Recalling that $\dot{v} \in \mathbb{R}$, and exploiting (5.46), it holds that

$$\begin{aligned}\sum_{j=0}^{k_\Phi-1} \mathrm{vec}\left(\tilde{V}_j \right)^\top \Lambda^\top_{\Phi_j} C^\top \sigma &= \left(\sum_{j=0}^{k_\Phi-1} \mathrm{vec}\left(\tilde{V}_j \right)^\top \Lambda^\top_{\Phi_j} C^\top \sigma \right)^\top \\ &= \sigma^\top C \sum_{j=0}^{k_\Phi-1} \Lambda_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j \right) \\ &= \sigma^\top \begin{bmatrix} C_1 & C_2 \end{bmatrix} \sum_{j=0}^{k_\Phi-1} \begin{bmatrix} \Lambda^{[1]}_{\Phi_j} \\ \Lambda^{[2]}_{\Phi_j} \end{bmatrix} \mathrm{vec}\left(\tilde{V}_j \right) \\ &= \sigma^\top C_1 \sum_{j=0}^{k_\Phi-1} \Lambda^{[1]}_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j \right) + \sigma^\top C_2 \sum_{j=0}^{k_\Phi-1} \Lambda^{[2]}_{\Phi_j} \mathrm{vec}\left(\tilde{V}_j \right). \quad \text{(B.9)}\end{aligned}$$

Hence, (B.8) simplifies into

$$\begin{aligned}\dot{v} \leq \sigma^\top C_1 \Bigg\{ \Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \Bigg\} + \sigma^\top C_2 \Bigg\{ \Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \\ + \sum_{i=1}^{m} \left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1} \Xi^{[i]}_{\Psi_j} \Delta_{\Psi_j} \right) u_{n,i} + \bar{B} u_r + \varepsilon_\Psi u_n + h \Bigg\} + \sigma^\top C_1 \Lambda^{[1]}_{\Phi_{k_\Phi}} \mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right) + \\ + \sigma^\top C_2 \Lambda^{[2]}_{\Phi_{k_\Phi}} \mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right) + \sigma^\top C_2 \sum_{i=1}^{m} \left(\Lambda^{[i]}_{\Psi_{k_\Psi}} \mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi} \right) \right) u_{n,i} + \\ + \sigma^\top C_2 \sum_{i=1}^{m} \left(\sum_{j=0}^{k_\Psi-1} \Lambda^{[i]}_{\Psi_j} \mathrm{vec}\left(\tilde{U}_j \right) \right) u_{n,i} - \mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right)^\top (\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top C_1^\top \sigma + \\ - \mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right)^\top (\Lambda^{[2]}_{\Phi_{k_\Phi}})^\top C_2^\top \sigma - \sum_{j=0}^{k_\Psi-1} \mathrm{vec}\left(\tilde{U}_j \right)^\top \left(\sum_{i=1}^{m} u_{n,i} (\Lambda^{[i]}_{\Psi_j})^\top \right) C_2^\top \sigma + \\ - \sum_{i=1}^{m} \mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi} \right)^\top u_{n,i} (\Lambda^{[i]}_{\Psi_{k_\Psi}})^\top C_2^\top \sigma. \quad \text{(B.10)}\end{aligned}$$

Moreover, the fact that $\dot{v} \in \mathbb{R}$, allows to write the following identities

$$\begin{aligned}
\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right)^\top (\Lambda_{\Phi_{k_\Phi}}^{[1]})^\top C_1^\top \sigma &= \left(\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right)^\top (\Lambda_{\Phi_{k_\Phi}}^{[1]})^\top C_1^\top \sigma\right)^\top \\
&= \sigma^\top C_1 \Lambda_{\Phi_{k_\Phi}}^{[1]} \operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right), \qquad \text{(B.11)} \\
\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)^\top (\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_2^\top \sigma &= \left(\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)^\top (\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_2^\top \sigma\right)^\top \\
&= \sigma^\top C_2 \Lambda_{\Phi_{k_\Phi}}^{[2]} \operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right), \qquad \text{(B.12)} \\
\sum_{j=0}^{k_\Psi-1} \operatorname{vec}\left(\tilde{U}_j\right)^\top \left(\sum_{i=1}^{m} u_{n,i} (\Lambda_{\Psi_j}^{[i]})^\top\right) C_2^\top \sigma &= \left(\sum_{j=0}^{k_\Psi-1} \operatorname{vec}\left(\tilde{U}_j\right)^\top \left(\sum_{i=1}^{m} u_{n,i} (\Lambda_{\Psi_j}^{[i]})^\top\right) C_2^\top \sigma\right)^\top \\
&= \sigma^\top C_2 \sum_{j=0}^{k_\Psi-1} \sum_{i=1}^{m} u_{n,i} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right) \\
&= \sigma^\top C_2 \sum_{i=1}^{m} \left(\sum_{j=0}^{k_\Psi-1} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right)\right) u_{n,i}, \qquad \text{(B.13)} \\
\sum_{i=1}^{m} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)^\top u_{n,i} (\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top C_2^\top \sigma &= \left(\sum_{i=1}^{m} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)^\top u_{n,i} (\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top C_2^\top \sigma\right)^\top \\
&= \sigma^\top C_2 \sum_{i=1}^{m} \left(\Lambda_{\Psi_{k_\Psi}}^{[i]} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)\right) u_{n,i}, \qquad \text{(B.14)}
\end{aligned}$$

which, if substituted in (B.10), this last one further simplifies into

$$\begin{aligned}
\dot{v}(x) \leq \sigma^\top C_1 \Bigg\{ \Delta_{\Phi_{k_\Phi}}^{[1]} + \sum_{j=1}^{k_\Phi-1} \Xi_{\Phi_j}^{[1]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[1]} \Bigg\} + \sigma^\top C_2 \Bigg\{ \Delta_{\Phi_{k_\Phi}}^{[2]} + \sum_{j=1}^{k_\Phi-1} \Xi_{\Phi_j}^{[2]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[2]} + \\
+ \sum_{i=1}^{m} \left(\Delta_{\Psi_{k_\Psi}}^{[i]} + \sum_{j=1}^{k_\Psi-1} \Xi_{\Psi_j}^{[i]} \Delta_{\Psi_j} \right) u_{n,i} + \bar{B} u_r + \varepsilon_\Psi u_n + h \Bigg\} \qquad \text{(B.15)}
\end{aligned}$$

Then, substituting the expression of the switching control law u_r in (5.44), one has that

$$\begin{aligned}
\dot{v}(x) \leq \sigma^\top C_1 \Bigg\{ \Delta_{\Phi_{k_\Phi}}^{[1]} + \sum_{j=1}^{k_\Phi-1} \Xi_{\Phi_j}^{[1]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[1]} \Bigg\} + \sigma^\top C_2 \Bigg\{ \Delta_{\Phi_{k_\Phi}}^{[2]} + \sum_{j=1}^{k_\Phi-1} \Xi_{\Phi_j}^{[2]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[2]} + \\
+ \sum_{i=1}^{m} \left(\Delta_{\Psi_{k_\Psi}}^{[i]} + \sum_{j=1}^{k_\Psi-1} \Xi_{\Psi_j}^{[i]} \Delta_{\Psi_j} \right) u_{n,i} + \varepsilon_\Psi u_n + h \Bigg\} - \rho \sigma^\top C_2 \bar{B} \frac{\sigma}{\|\sigma\|}. \qquad \text{(B.16)}
\end{aligned}$$

By means of Assumption 5.1, 5.3 ,and 5.6, and Proposition 5.2, it holds that

$$\dot{v} \leq \|\sigma\| \|C_1\| \left\{ \bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1} \right\} + \|\sigma\| \|C_2\| \left\{ \bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \|u_n\| + \bar{h} \right\} - \rho \sigma^\top C_2 \bar{B} \frac{\sigma}{\|\sigma\|}.$$

Then, since $\bar{B}$ is assumed to be symmetric and positive definite (Assumption 5.2) and C_2 is chosen according to Assumption 5.4, it holds that

$$\sigma^\top C_2 \bar{B} \sigma \geq \|\sigma\|^2 \underline{\lambda}(C_2) \underline{\lambda}(\bar{B}) \geq \|\sigma\|^2 \underline{\lambda}(C_2) \underline{\gamma}. \qquad \text{(B.17)}$$

Hence, the above inequality can be bounded as

$$\begin{aligned}\dot{v} &\leq \|\sigma\| \|C_1\| \{\bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1}\} + \|\sigma\| \|C_2\| \left\{\bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \|u_n\| + \bar{h}\right\} - \rho \|\sigma\| \underline{\lambda}(C_2)\underline{\gamma} \\ &\leq \|\sigma\| \left\{\|C_1\| (\bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1}) + \|C_2\| \left[\bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \|u_n\| + \bar{h}\right] - \rho\underline{\lambda}(C_2)\underline{\gamma}\right\}.\end{aligned}$$

If the control gain ρ is chosen accordingly to Theorem 5.1, i.e.,

$$\rho > \frac{\|C_1\| \{\bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1}\} + \|C_2\| \{\bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \|u_n\| + \bar{h}\} + \bar{\eta}}{\underline{\lambda}(C_2)\underline{\gamma}},$$

with $\bar{\eta}\mathbb{R}_{>0}$, then the first time-derivative of the Lyapunov function is bounded as

$$\dot{v} \leq -\bar{\eta} \|\sigma\| \tag{B.18}$$

and, as a consequence it is, guarantee that $\sigma(x(t)) \to 0_m$ for $t \to \infty$, which concludes the proof.

B.2 Proof of Theorem 5.2

The development of proof is analogous to the one in [53, Theorem 3]. Consider the Lyapunov candidate function $v : \mathcal{X} \to \mathbb{R}$ defined as

$$v = \frac{1}{2}\sigma^\top\sigma + \frac{1}{2}\sum_{i=1}^{2}\tilde{\rho}_i^2, \tag{B.19}$$

where $\tilde{\rho}_i = \rho_i^\star - \hat{\rho}_i$ is the error between the components of the discontinuous control gain and the ones of an unknown constant ideal gain

$$\rho^\star = \rho_1^\star + \rho_2^\star \|u_n\| + \frac{\eta^\star}{\underline{\lambda}(C_2)\underline{\gamma}},$$

with $\eta^\star \in \mathbb{R}_{>0}$, defined as the one capable of enforcing a sliding mode $\sigma(x(t)) = 0_m$ for $t \geq t_1$. From such a definition, it is immediate that the components of $\rho^\star$ are

$$\rho_1^\star = \frac{\|C_1\| \left(\sup_{x\in\mathcal{X}} \left\|f_1 - \hat{\Phi}^{[1]}_{k_\Phi,t_1}\right\|\right) + \|C_2\| \left(\sup_{x\in\mathcal{X}} \left\|f_2 - \hat{\Phi}^{[2]}_{k_\Phi,t_1}\right\| + \bar{h}\right)}{\underline{\lambda}(C_2)\underline{\gamma}}, \tag{B.20}$$

$$\rho_2{}^\star = \frac{\|C_2\| \sup_{x\in\mathcal{X}} \left\|\sum_{i=1}^{m} \bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi,t_k}\right\|}{\underline{\lambda}(C_2)\underline{\gamma}}, \tag{B.21}$$

with $\hat{\Phi}^{[1]}_{k_\Phi,t_1}, \hat{\Phi}^{[2]}_{k_\Phi,t_1}$, and $\hat{\Psi}^{[i]}_{k_\Psi,t_1}$ denoting the output of the DNNs characterized by fixed weights $\hat{V}_j(t_1)$, for $j \in \{0, 1, \ldots, k_\Phi\}$ and $\hat{U}_j(t_1)$, for $j \in \{0, 1, \ldots, k_\Psi\}$.

The first time-derivative of (B.19) is computed as

$$\begin{aligned}\dot{v} &= \sigma^\top\dot{\sigma} - \sum_{i=1}^{2}\tilde{\rho}_i\dot{\hat{\rho}}_i, \\ &= \sigma^\top\left[C_1\left(\tilde{\Phi}^{[1]}_{k_\Phi,t_1} + \varepsilon^{[1]}_\Phi\right) + C_2\left(\tilde{\Phi}^{[2]}_{k_\Phi,t_1} + \sum_{i=1}^{m}\tilde{\Psi}^{[i]}_{k_\Psi,t_1}u_{n,i} + \varepsilon^{[2]}_\Phi + \varepsilon_\Psi u_n + h\right)\right] +\end{aligned}$$

$$- \sigma^\top C_2 \bar{B}(\hat{\rho}_1 + \hat{\rho}_2 \|u_n\|)\frac{\sigma}{\|\sigma\|} - \tilde{\rho}_1\dot{\hat{\rho}}_1 - \tilde{\rho}_2\dot{\hat{\rho}}_2,$$

where $\tilde{\Phi}^{[1]}_{k_\Phi,t_1} = \Phi^{[1]}_{k_\Phi} - \hat{\Phi}^{[1]}_{k_\Phi,t_1}$, $\tilde{\Phi}^{[2]}_{k_\Phi,t_1} = \Phi^{[2]}_{k_\Phi} - \hat{\Phi}^{[2]}_{k_\Phi,t_1}$, and $\tilde{\Psi}^{[i]}_{k_\Psi,t_1} = \Psi^{[i]}_{k_\Psi} - \hat{\Psi}^{[1]}_{k_\Psi,t_1}$. Recalling that $\hat{\rho} = \rho^\star - \tilde{\rho} = \rho_1^\star + \rho_2^\star \|u_n\| + \frac{\eta^\star}{\underline{\lambda}(C_2)\underline{\gamma}} - \tilde{\rho}_1 - \tilde{\rho}_2\|u_n\|$, $\dot{v}(x)$ can be expanded as

$$\begin{aligned}\dot{v} = \sigma^\top\left[C_1\left(\tilde{\Phi}^{[1]}_{k_\Phi,t_1} + \varepsilon^{[1]}_\Phi\right) + C_2\left(\tilde{\Phi}^{[2]}_{k_\Phi,t_1} + \sum_{i=1}^m \tilde{\Psi}^{[i]}_{k_\Psi,t_1}u_{n,i} + \varepsilon^{[2]}_\Phi + \varepsilon_\Psi u_n + h\right)\right] + \\ - \sigma^\top C_2\bar{B}\left(\rho_1^\star + \rho_2^\star\|u_n\| + \frac{\eta^\star}{\underline{\lambda}(C_2)\underline{\gamma}}\right)\frac{\sigma}{\|\sigma\|} + \sigma^\top C_2\bar{B}(\tilde{\rho}_1 + \tilde{\rho}_2\|u_n\|)\frac{\sigma}{\|\sigma\|} + \\ - \tilde{\rho}_1\dot{\hat{\rho}}_1 - \tilde{\rho}_2\dot{\hat{\rho}}_2.\end{aligned}$$

Exploiting Proposition 5.3 to bound the approximation errors and bounding the other terms relying on Assumptions 5.2 and 5.3, it holds that

$$\begin{aligned}\dot{v} \le \|\sigma\|\|C_1\|\sup_{x\in\mathcal{X}}\left\|f_1 - \hat{\Phi}^{[1]}_{k_\Phi,t_1}\right\| + \|\sigma\|\|C_2\|\sup_{x\in\mathcal{X}}\left\|f_2 - \hat{\Phi}^{[2]}_{k_\Phi,t_1}\right\| + \|\sigma\|\|C_2\|\bar{h} + \\ + \|\sigma\|\|C_2\|\sup_{x\in\mathcal{X}}\left\|\sum_{i=1}^m \bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi,t_k}\right\|\|u_n\| - \|\sigma\|\underline{\lambda}(C_2)\underline{\gamma}\rho_1^\star + \\ - \|\sigma\|\underline{\lambda}(C_2)\underline{\gamma}\rho_2^\star\|u_n\| - \|\sigma\|\eta^\star + \|\sigma\|\|C_2\|\bar{\gamma}\|\tilde{\rho}_1\| + \|\sigma\|\|C_2\|\bar{\gamma}\|\tilde{\rho}_2\|\|u_n\| + \\ - \tilde{\rho}_1\dot{\hat{\rho}}_1 - \tilde{\rho}_2\dot{\hat{\rho}}_2.\end{aligned}$$

Substituting the values $\rho_1^\star$ and $\rho_2^\star$ as in (B.20) and (B.21), respectively, the above inequality can be simplified as

$$\dot{v} \le \|\sigma\|\|C_2\|\bar{\gamma}\|\tilde{\rho}_1\| + \|\sigma\|\|C_2\|\bar{\gamma}\|\tilde{\rho}_2\|\|u_n\| - \tilde{\rho}_1\dot{\hat{\rho}}_1 - \tilde{\rho}_2\dot{\hat{\rho}}_2 - \|\sigma\|\eta^\star.$$

Substituting the adaptation laws (5.60), it holds that

$$\begin{aligned}\dot{v} \le \|\sigma\|\|C_2\|\bar{\gamma}\|\tilde{\rho}_1\| + \|\sigma\|\|C_2\|\bar{\gamma}\|\tilde{\rho}_2\|\|u_n\| - \tilde{\rho}_1\underset{\varrho_1}{\text{proj}}\left(\left(\|C_2\|\bar{\gamma} + \alpha_1\right)\|\sigma\|\right) + \\ - \tilde{\rho}_2\underset{\varrho_2}{\text{proj}}\left(\left(\|C_2\|\bar{\gamma}\|u_n\| + \alpha_2\right)\|\sigma\|\right) - \|\sigma\|\eta^\star.\end{aligned}$$

Moreover, if one let $\rho_i^\star > \bar{\rho}_i$, since $\hat{\rho}_i \le \bar{\rho}_i$ by virtue of the projection operator, then $\|\tilde{\rho}_i\| \equiv \tilde{\rho}_i$, for $i \in \{1,2\}$. Hence, it holds that

$$\begin{aligned}\dot{v} \le \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_1 + \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_2\|u_n\| - \tilde{\rho}_1\underset{\varrho_1}{\text{proj}}\left(\left(\|C_2\|\bar{\gamma} + \alpha_1\right)\|\sigma\|\right) + \\ - \tilde{\rho}_2\underset{\varrho_2}{\text{proj}}\left(\left(\|C_2\|\bar{\gamma}\|u_n\| + \alpha_2\right)\|\sigma\|\right) - \|\sigma\|\eta^\star.\end{aligned}$$

Exploiting the property of the project operator in Lemma A.1, one has

$$\begin{aligned}\dot{v} &\le \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_1 + \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_2\|u_n\| - \tilde{\rho}_1\Big(\|C_2\|\bar{\gamma} + \alpha_1\Big)\|\sigma\| + \\ &\quad - \tilde{\rho}_2\Big(\|C_2\|\bar{\gamma}\|u_n\| + \alpha_2\Big)\|\sigma\| - \|\sigma\|\eta^\star \\ &\le \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_1 + \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_2\|u_n\| - \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_1 - \|\sigma\|\|C_2\|\alpha_1\tilde{\rho}_1 + \\ &\quad - \|\sigma\|\|C_2\|\bar{\gamma}\tilde{\rho}_2\|u_n\| - \|\sigma\|\|C_2\|\alpha_2\tilde{\rho}_2 - \|\sigma\|\eta^\star\end{aligned}$$

$$
\begin{aligned}
&\leq -\|\sigma\|\,\|C_2\|\,\alpha_1\tilde{\rho}_1 - \|\sigma\|\,\|C_2\|\,\alpha_2\tilde{\rho}_2 - \|\sigma\|\,\eta^\star \\
&\leq -\|\sigma\|\,\|C_2\|\,\alpha_1\,\|\tilde{\rho}_1\| - \|\sigma\|\,\|C_2\|\,\alpha_2\,\|\tilde{\rho}_2\| - \|\sigma\|\,\eta^\star \\
&\leq -\eta_1\,\|\tilde{\rho}_1\| - \eta_2\,\|\tilde{\rho}_2\| - \eta^\star\,\|\sigma\| \\
&\leq -\sqrt{2}\min\{\eta^\star,\eta_1,\eta_2\}\left(\frac{\|\sigma\|}{\sqrt{2}} + \frac{\|\tilde{\rho}_1\|}{\sqrt{2}} + \frac{\|\tilde{\rho}_2\|}{\sqrt{2}}\right) \\
&\leq -\sqrt{2}\min\{\eta^\star,\eta_1,\eta_2\}\sqrt{v(x)},
\end{aligned}
$$

proving that there exist $t_2 \geq t_1$ such that $\sigma(x(t)) = 0_m$ for $t \geq t_2$ [53], and concluding the proof.

B.3 Proof of Theorem 6.1

The proof follows the exact same reasoning of the one of Theorem 5.1, but it is reported for sake of completeness.

Consider the Lyapunov-like candidate function $v : \mathcal{X} \to \mathbb{R}$ given by

$$
v = \frac{1}{2}\sigma^\top\sigma + \frac{1}{2}\sum_{j=0}^{k_\Phi}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\mathrm{vec}\left(\tilde{V}_j\right) + \frac{1}{2}\sum_{j=0}^{k_\Psi}\mathrm{vec}\left(\tilde{U}_j\right)^\top\Gamma_{\Psi_j}^{-1}\mathrm{vec}\left(\tilde{U}_j\right), \tag{B.22}
$$

where σ is the integral sliding variable defined as in (6.10). The first time-derivative of v is equal to

$$
\dot{v} = \sigma^\top\dot{\sigma} + \sum_{j=0}^{k_\Phi}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\mathrm{vec}\left(\dot{\tilde{V}}_j\right) + \sum_{j=0}^{k_\Psi}\mathrm{vec}\left(\tilde{U}_j\right)^\top\Gamma_{\Psi_j}^{-1}\mathrm{vec}\left(\dot{\tilde{U}}_j\right). \tag{B.23}
$$

The first time-derivative of σ is obtained computing the first time derivative of σ_0 in (6.5) and substituting the value of $\dot{\hat{z}}$ in (6.11). In particular

$$
\begin{aligned}
\dot{\sigma} &= \dot{\sigma}_0 - \dot{\hat{z}} \\
&= \delta_a C_{1,a}(\dot{x}_1 - \dot{x}_{1,s}) + (1-\delta_a)C_{1,r}(\dot{x}_1 - \dot{x}_1^\star) + \delta_a C_{2,a}(\dot{x}_2 - \dot{x}_{2,s}) + \\
&\quad + (1-\delta_a)C_{2,r}(\dot{x}_2 - \dot{x}_2^\star) - \delta_a C_{1,a}(\hat{\Phi}_{k_\Phi}^{[1]} - \dot{x}_{1,s}) - (1-\delta_a)C_{1,r}(\hat{\Phi}_{k_\Phi}^{[1]} - \dot{x}_1^\star) + \\
&\quad - \delta_a C_{2,a}\Big(\hat{\Phi}_{k_\Phi}^{[2]} + \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,a,i} - \dot{x}_{2,s}\Big) - (1-\delta_a)C_{2,r}\Big(\hat{\Phi}_{k_\Phi}^{[2]} + \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,r,i} - \dot{x}_2^\star\Big) \\
&= \delta_a C_{1,a}\Big(\dot{x}_1 - \hat{\Phi}_{k_\Phi}^{[1]}\Big) + \delta_a C_{2,a}\Big(\dot{x}_2 - \hat{\Phi}_{k_\Phi}^{[2]} - \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,a,i}\Big) + \\
&\quad + (1-\delta_a)C_{1,r}\Big(\dot{x}_1 - \hat{\Phi}_{k_\Phi}^{[1]}\Big) + (1-\delta_a)C_{2,r}\Big(\dot{x}_2 - \hat{\Phi}_{k_\Phi}^{[2]} - \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,r,i}\Big) \\
&= C_{1,\delta}\Big(f_1 - \hat{\Phi}_{k_\Phi}^{[1]}\Big) + \delta_a C_{2,a}\Big(f_2 + \bar{B}u_n + \bar{B}u_r + h - \hat{\Phi}_{k_\Phi}^{[2]} - \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,a,i}\Big) + \\
&\quad + (1-\delta_a)C_{2,r}\Big(f_2 + \bar{B}u_n + \bar{B}u_r + h - \hat{\Phi}_{k_\Phi}^{[2]} - \sum_{i=1}^{m}\hat{\Psi}_{k_\Psi}^{[i]}u_{n,r,i}\Big).
\end{aligned}
$$

Recalling that u_n is defined as in (6.9a), and collecting common terms, the last equation can be rewritten as

$$\dot{\sigma} = C_{1,\delta}\Big(f_1 - \hat{\Phi}^{[1]}_{k_\Phi}\Big) + C_{2,\delta}\Big(f_2 - \hat{\Phi}^{[2]}_{k_\Phi} + \bar{B}u_r + h\Big) + \\ + \delta_a C_{2,a}\left(\sum_{i=1}^{m}\Big(\bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi}\Big)u_{n,a,i}\right) + (1-\delta_a)C_{2,r}\left(\sum_{i=1}^{m}\Big(\bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi}\Big)u_{n,r,i}\right).$$

Substituting the nominal dynamics with its estimation in (5.14), it holds that

$$\dot{\sigma} = C_{1,\delta}\Big(\Phi^{[1]}_{k_\Phi} + \varepsilon^{[1]}_\Phi - \hat{\Phi}^{[2]}_{k_\Phi}\Big) + C_{2,\delta}\Big(\Phi^{[2]}_{k_\Phi} + \varepsilon^{[1]}_\Phi - \hat{\Phi}^{[2]}_{k_\Phi} + \bar{B}u_r + h\Big) + \\ + \delta_a C_{2,a}\left(\sum_{i=1}^{m}\Big(\Psi^{[i]}_{k_\Psi} - \hat{\Psi}^{[i]}_{k_\Psi}\Big)u_{n,a,i} + \varepsilon_\Psi u_{n,a}\right) + \\ + (1-\delta_a)C_{2,r}\left(\sum_{i=1}^{m}\Big(\Psi^{[i]}_{k_\Psi} - \hat{\Psi}^{[i]}_{k_\Psi}\Big)u_{n,r,i} + \varepsilon_\Psi u_{n,r}\right).$$

Finally, since $\tilde{\Phi}^{[p]}_{k_\Phi} = \Phi^{[p]}_{k_\Phi} - \hat{\Phi}^{[p]}_{k_\Phi}$, and $\tilde{\Psi}^{[i]}_{k_\Psi} = \Phi^{[i]}_{k_\Psi} - \hat{\Psi}^{[i]}_{k_\Psi}$, one can express $\dot{\sigma}$ as

$$\dot{\sigma} = C_{1,\delta}\Big(\tilde{\Phi}^{[1]}_{k_\Phi} + \varepsilon^{[1]}_\Phi\Big) + C_{2,\delta}\Big(\tilde{\Phi}^{[2]}_{k_\Phi} + \varepsilon^{[2]}_\Phi + \bar{B}u_r + h\Big) + \\ + \delta_a C_{2,a}\left(\sum_{i=1}^{m}\tilde{\Psi}^{[i]}_{k_\Psi}u_{n,a,i} + \varepsilon_\Psi u_{n,a}\right) + (1-\delta_a)C_{2,r}\left(\sum_{i=1}^{m}\tilde{\Psi}^{[i]}_{k_\Psi}u_{n,r,i} + \varepsilon_\Psi u_{n,r}\right). \quad \text{(B.24)}$$

Substituting (B.24) into (B.23) leads to

$$\dot{v} = \sigma^\top C_{1,\delta}\Big(\tilde{\Phi}^{[1]}_{k_\Phi} + \varepsilon^{[1]}_\Phi\Big) + \sigma^\top C_{2,\delta}\Big(\tilde{\Phi}^{[2]}_{k_\Phi} + \varepsilon^{[2]}_\Phi + \bar{B}u_r + h\Big) + \\ + \delta_a \sigma^\top C_{2,a}\left(\sum_{i=1}^{m}\tilde{\Psi}^{[i]}_{k_\Psi}u_{n,a,i} + \varepsilon_\Psi u_{n,a}\right) + (1-\delta_a)\sigma^\top C_{2,r}\left(\sum_{i=1}^{m}\tilde{\Psi}^{[i]}_{k_\Psi}u_{n,r,i} + \varepsilon_\Psi u_{n,r}\right) + \\ - \sum_{j=0}^{k_\Phi}\mathrm{vec}\left(\tilde{V}_j\right)^\top \Gamma^{-1}_{\Phi_j}\mathrm{vec}\left(\dot{\hat{V}}_j\right) + \sum_{j=0}^{k_\Psi}\mathrm{vec}\left(\tilde{U}_j\right)^\top \Gamma^{-1}_{\Psi_j}\mathrm{vec}\left(\dot{\hat{U}}_j\right),$$

which, expanding the last two terms can be written as

$$\dot{v} = \sigma^\top C_{1,\delta}\Big(\tilde{\Phi}^{[1]}_{k_\Phi} + \varepsilon^{[1]}_\Phi\Big) + \sigma^\top C_{2,\delta}\Big(\tilde{\Phi}^{[2]}_{k_\Phi} + \varepsilon^{[2]}_\Phi + \bar{B}u_r + h\Big) + \\ + \delta_a \sigma^\top C_{2,a}\left(\sum_{i=1}^{m}\tilde{\Psi}^{[i]}_{k_\Psi}u_{n,a,i} + \varepsilon_\Psi u_{n,a}\right) + (1-\delta_a)\sigma^\top C_{2,r}\left(\sum_{i=1}^{m}\tilde{\Psi}^{[i]}_{k_\Psi}u_{n,r,i} + \varepsilon_\Psi u_{n,r}\right) + \\ - \sum_{j=0}^{k_\Phi-1}\mathrm{vec}\left(\tilde{V}_j\right)^\top \Gamma^{-1}_{\Phi_j}\mathrm{vec}\left(\dot{\hat{V}}_j\right) - \sum_{p=1}^{2}\mathrm{vec}\left(\tilde{V}^{[p]}_{k_\Phi}\right)^\top \left(\Gamma^{[p]}_{\Phi_{k_\Phi}}\right)^{-1}\mathrm{vec}\left(\dot{\hat{V}}^{[p]}_{k_\Phi}\right) + \\ - \sum_{j=0}^{k_\Psi-1}\mathrm{vec}\left(\tilde{U}_j\right)^\top \Gamma^{-1}_{\Psi_j}\mathrm{vec}\left(\dot{\hat{U}}_j\right) - \sum_{i=1}^{m}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top \Gamma^{-1}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\dot{\hat{U}}^{[i]}_{k_\Psi}\right).$$

Then, substituting $\tilde{\Phi}^{[1]}_{k_\Phi}$, $\tilde{\Phi}^{[2]}_{k_\Phi}$, and $\tilde{\Psi}^{[i]}_{k_\Psi}$ with the expressions (5.29a), (5.29b), and (5.38), respectively, one obtains

$$\dot{v} = \sigma^\top C_{1,\delta}\left\{\Lambda^{[1]}_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right) + \Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=0}^{k_\Phi-1}\Lambda^{[1]}_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi-1}\Xi^{[1]}_{\Phi_j}\Delta_{\Phi_j} + \varepsilon^{[1]}_\Phi\right\} +$$

$$
\begin{aligned}
&+\sigma^\top C_{2,\delta}\Bigg\{\Lambda^{[2]}_{\Phi_{k_\Phi}}\operatorname{vec}\left(\tilde{V}^{[2]}_{k_\Phi}\right)+\Delta^{[2]}_{\Phi_{k_\Phi}}+\sum_{j=0}^{k_\Phi-1}\Lambda^{[2]}_{\Phi_j}\operatorname{vec}\left(\tilde{V}_j\right)+\sum_{j=1}^{k_\Phi-1}\Xi^{[2]}_{\Phi_j}\Delta_{\Phi_j}+\varepsilon^{[2]}_{\Phi}+\\
&+\bar{B}u_r+h\Bigg\}+\delta_a\sigma^\top C_{2,a}\Bigg\{\sum_{i=1}^{m}\Bigg(\Lambda^{[i]}_{\Psi_{k_\Psi}}\operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)+\Delta^{[i]}_{\Psi_{k_\Psi}}+\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\operatorname{vec}\left(\tilde{U}_j\right)+\\
&+\sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\Bigg)u_{n,a,i}+\varepsilon_\Psi u_{n,a}\Bigg\}+(1-\delta_a)\sigma^\top C_{2,r}\Bigg\{\sum_{i=1}^{m}\Bigg(\Lambda^{[i]}_{\Psi_{k_\Psi}}\operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)+\\
&+\Delta^{[i]}_{\Psi_{k_\Psi}}\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\operatorname{vec}\left(\tilde{U}_j\right)++\sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\Bigg)u_{n,r,i}+\varepsilon_\Psi u_{n,r}\Bigg\}+\\
&-\sum_{j=0}^{k_\Phi-1}\operatorname{vec}\left(\tilde{V}_j\right)^\top\Gamma^{-1}_{\Phi_j}\operatorname{vec}\left(\dot{\hat{V}}_j\right)-\sum_{p=1}^{2}\operatorname{vec}\left(\tilde{V}^{[p]}_{k_\Phi}\right)^\top\left(\Gamma^{[p]}_{\Phi_{k_\Phi}}\right)^{-1}\operatorname{vec}\left(\dot{\hat{V}}^{[p]}_{k_\Phi}\right)+\\
&-\sum_{j=0}^{k_\Psi-1}\operatorname{vec}\left(\tilde{U}_j\right)^\top\Gamma^{-1}_{\Psi_j}\operatorname{vec}\left(\dot{\hat{U}}_j\right)-\sum_{i=1}^{m}\operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top\Gamma^{-1}_{\Psi_{k_\Psi}}\operatorname{vec}\left(\dot{\hat{U}}^{[i]}_{k_\Psi}\right).
\end{aligned}
$$

Then, the terms can be rearranged to isolate the ones that must be compensated via the adaptation laws

$$
\begin{aligned}
\dot{v}=&\sigma^\top C_{1,\delta}\Bigg\{\Delta^{[1]}_{\Phi_{k_\Phi}}+\sum_{j=1}^{k_\Phi-1}\Xi^{[1]}_{\Phi_j}\Delta_{\Phi_j}+\varepsilon^{[1]}_{\Phi}\Bigg\}+\sigma^\top C_{2,\delta}\Bigg\{\Delta^{[2]}_{\Phi_{k_\Phi}}+\sum_{j=1}^{k_\Phi-1}\Xi^{[2]}_{\Phi_j}\Delta_{\Phi_j}+\varepsilon^{[2]}_{\Phi}+\\
&+\bar{B}u_r+h\Bigg\}+\delta_a\sigma^\top C_{2,a}\Bigg\{\sum_{i=1}^{m}\Bigg(\Delta^{[i]}_{\Psi_{k_\Psi}}+\sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\Bigg)u_{n,a,i}+\varepsilon_\Psi u_{n,a}\Bigg\}+\\
&+(1-\delta_a)\sigma^\top C_{2,r}\Bigg\{\sum_{i=1}^{m}\Bigg(\Delta^{[i]}_{\Psi_{k_\Psi}}+\sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\Bigg)u_{n,r,i}+\varepsilon_\Psi u_{n,r}\Bigg\}+\\
&+\sigma^\top C_{1,\delta}\Lambda^{[1]}_{\Phi_{k_\Phi}}\operatorname{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right)+\sigma^\top C_{1,\delta}\sum_{j=0}^{k_\Phi-1}\Lambda^{[1]}_{\Phi_j}\operatorname{vec}\left(\tilde{V}_j\right)+\\
&+\sigma^\top C_{2,\delta}\Lambda^{[2]}_{\Phi_{k_\Phi}}\operatorname{vec}\left(\tilde{V}^{[2]}_{k_\Phi}\right)+\sigma^\top C_{2,\delta}\sum_{j=0}^{k_\Phi-1}\Lambda^{[2]}_{\Phi_j}\operatorname{vec}\left(\tilde{V}_j\right)+\\
&+\delta_a\sigma^\top C_{2,a}\sum_{i=1}^{m}\Lambda^{[i]}_{\Psi_{k_\Psi}}\operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)u_{n,a,i}+\delta_a\sigma^\top C_{2,a}\sum_{i=1}^{m}\Bigg(\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\operatorname{vec}\left(\tilde{U}_j\right)\Bigg)u_{n,a,i}+\\
&+(1-\delta_a)\sigma^\top C_{2,r}\sum_{i=1}^{m}\Lambda^{[i]}_{\Psi_{k_\Psi}}\operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)u_{n,r,i}+\\
&+(1-\delta_a)\sigma^\top C_{2,r}\sum_{i=1}^{m}\Bigg(\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\operatorname{vec}\left(\tilde{U}_j\right)\Bigg)u_{n,r,i}+\\
&-\sum_{j=0}^{k_\Phi-1}\operatorname{vec}\left(\tilde{V}_j\right)^\top\Gamma^{-1}_{\Phi_j}\operatorname{vec}\left(\dot{\hat{V}}_j\right)-\sum_{p=1}^{2}\operatorname{vec}\left(\tilde{V}^{[p]}_{k_\Phi}\right)^\top\left(\Gamma^{[p]}_{\Phi_{k_\Phi}}\right)^{-1}\operatorname{vec}\left(\dot{\hat{V}}^{[p]}_{k_\Phi}\right)+\\
&-\sum_{j=0}^{k_\Psi-1}\operatorname{vec}\left(\tilde{U}_j\right)^\top\Gamma^{-1}_{\Psi_j}\operatorname{vec}\left(\dot{\hat{U}}_j\right)-\sum_{i=1}^{m}\operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top\Gamma^{-1}_{\Psi_{k_\Psi}}\operatorname{vec}\left(\dot{\hat{U}}^{[i]}_{k_\Psi}\right).
\end{aligned}
$$

Then, substituting the weight adaptation laws (6.12), (6.14), (6.15), and (6.16), and ex-

ploiting points 3 and 4 of Lemma 5.2, one has that

$$\begin{aligned}
\dot{v} \leq\ & \sigma^\top C_{1,\delta}\left\{\Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1}\Xi^{[1]}_{\Phi_j}\Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi}\right\} + \sigma^\top C_{2,\delta}\left\{\Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1}\Xi^{[2]}_{\Phi_j}\Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \right.\\
&\left. + \bar{B}u_r + h\right\} + \delta_a\sigma^\top C_{2,a}\left\{\sum_{i=1}^{m}\left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\right)u_{n,a,i} + \varepsilon_\Psi u_{n,a}\right\} +\\
&+ (1-\delta_a)\sigma^\top C_{2,r}\left\{\sum_{i=1}^{m}\left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\right)u_{n,r,i} + \varepsilon_\Psi u_{n,r}\right\} +\\
&+ \sigma^\top C_{1,\delta}\Lambda^{[1]}_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right) + \sigma^\top C_{1,\delta}\sum_{j=0}^{k_\Phi-1}\Lambda^{[1]}_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) +\\
&+ \sigma^\top C_{2,\delta}\Lambda^{[2]}_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi}\right) + \sigma^\top C_{2,\delta}\sum_{j=0}^{k_\Phi-1}\Lambda^{[2]}_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) +\\
&+ \delta_a\sigma^\top C_{2,a}\sum_{i=1}^{m}\Lambda^{[i]}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)u_{n,a,i} + \delta_a\sigma^\top C_{2,a}\sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\mathrm{vec}\left(\tilde{U}_j\right)\right)u_{n,a,i} +\\
&+ (1-\delta_a)\sigma^\top C_{2,r}\sum_{i=1}^{m}\Lambda^{[i]}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)u_{n,r,i} +\\
&+ (1-\delta_a)\sigma^\top C_{2,r}\sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\mathrm{vec}\left(\tilde{U}_j\right)\right)u_{n,r,i} +\\
&- \sum_{j=0}^{k_\Phi-1}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Lambda^\top_{\Phi_j}C^\top_\delta\sigma - \mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right)^\top(\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top(C_{1,\delta})^\top\sigma +\\
&- \mathrm{vec}\left(\tilde{V}^{[2]}_{k_\Phi}\right)^\top(\Lambda^{[2]}_{\Phi_{k_\Phi}})^\top(C_{2,\delta})^\top\sigma - \sum_{j=0}^{k_\Psi-1}\mathrm{vec}\left(\tilde{U}_j\right)^\top\left(\sum_{i=1}^{m}u_{n,i}(\Lambda^{[i]}_{\Psi_j})^\top\right)(C_{2,\delta})^\top\sigma +\\
&- \sum_{i=1}^{m}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top u_{n,i}(\Lambda^{[i]}_{\Psi_{k_\Psi}})^\top(C_{2,\delta})^\top\sigma. \qquad \text{(B.25)}
\end{aligned}$$

Since $\dot{v} \in \mathbb{R}$, if one exploits (5.46), it holds that

$$\begin{aligned}
\sum_{j=0}^{k_\Phi-1}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Lambda^\top_{\Phi_j}C^\top_\delta\sigma &= \left(\sum_{j=0}^{k_\Phi-1}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Lambda^\top_{\Phi_j}C^\top_\delta\sigma\right)^\top\\
&= \sigma^\top C_\delta\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right)\\
&= \sigma^\top\begin{bmatrix}C_{1,\delta} & C_{2,\delta}\end{bmatrix}\sum_{j=0}^{k_\Phi-1}\begin{bmatrix}\Lambda^{[1]}_{\Phi_j}\\ \Lambda^{[2]}_{\Phi_j}\end{bmatrix}\mathrm{vec}\left(\tilde{V}_j\right)\\
&= \sigma^\top C_{1,\delta}\sum_{j=0}^{k_\Phi-1}\Lambda^{[1]}_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) + \sigma^\top C_{2,\delta}\sum_{j=0}^{k_\Phi-1}\Lambda^{[2]}_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right), \qquad \text{(B.26)}\\
\mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right)^\top(\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top C^\top_{1,\delta}\sigma &= \left(\mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right)^\top(\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top C^\top_{1,\delta}\sigma\right)^\top\\
&= \sigma^\top C_{1,\delta}\Lambda^{[1]}_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}^{[1]}_{k_\Phi}\right), \qquad \text{(B.27)}
\end{aligned}$$

$$\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)^\top (\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_{2,\delta}^\top \sigma = \left(\operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right)^\top (\Lambda_{\Phi_{k_\Phi}}^{[2]})^\top C_{2,\delta}^\top \sigma\right)^\top$$
$$= \sigma^\top C_{2,\delta} \Lambda_{\Phi_{k_\Phi}}^{[2]} \operatorname{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right), \tag{B.28}$$

and (B.25) can be simplified as

$$\begin{aligned}
\dot{v} \leq\ & \sigma^\top C_{1,\delta}\left\{\Delta_{\Phi_{k_\Phi}}^{[1]} + \sum_{j=1}^{k_\Phi-1} \Xi_{\Phi_j}^{[1]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[1]}\right\} + \sigma^\top C_{2,\delta}\left\{\Delta_{\Phi_{k_\Phi}}^{[2]} + \sum_{j=1}^{k_\Phi-1} \Xi_{\Phi_j}^{[2]} \Delta_{\Phi_j} + \varepsilon_\Phi^{[2]} + \right. \\
& \left. + \bar{B} u_r + h\right\} + \delta_a \sigma^\top C_{2,a}\left\{\sum_{i=1}^{m}\left(\Delta_{\Psi_{k_\Psi}}^{[i]} + \sum_{j=1}^{k_\Psi-1} \Xi_{\Psi_j}^{[i]} \Delta_{\Psi_j}\right) u_{n,a,i} + \varepsilon_\Psi u_{n,a}\right\} + \\
& + (1-\delta_a)\sigma^\top C_{2,r}\left\{\sum_{i=1}^{m}\left(\Delta_{\Psi_{k_\Psi}}^{[i]} + \sum_{j=1}^{k_\Psi-1} \Xi_{\Psi_j}^{[i]} \Delta_{\Psi_j}\right) u_{n,r,i} + \varepsilon_\Psi u_{n,r}\right\} + \\
& + \delta_a \sigma^\top C_{2,a} \sum_{i=1}^{m} \Lambda_{\Psi_{k_\Psi}}^{[i]} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right) u_{n,a,i} + \delta_a \sigma^\top C_{2,a} \sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right)\right) u_{n,a,i} + \\
& + (1-\delta_a)\sigma^\top C_{2,r} \sum_{i=1}^{m} \Lambda_{\Psi_{k_\Psi}}^{[i]} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right) u_{n,r,i} + \\
& + (1-\delta_a)\sigma^\top C_{2,r} \sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right)\right) u_{n,r,i} + \\
& - \sum_{j=0}^{k_\Psi-1} \operatorname{vec}\left(\tilde{U}_j\right)^\top \left(\sum_{i=1}^{m} u_{n,i} (\Lambda_{\Psi_j}^{[i]})^\top\right) (C_{2,\delta})^\top \sigma + \\
& - \sum_{i=1}^{m} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)^\top u_{n,i} (\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top (C_{2,\delta})^\top \sigma.
\end{aligned} \tag{B.29}$$

Moreover, it holds that

$$\begin{aligned}
\sum_{j=0}^{k_\Psi-1} \operatorname{vec}\left(\tilde{U}_j\right)^\top \left(\sum_{i=1}^{m} u_{n,i} (\Lambda_{\Psi_j}^{[i]})^\top\right) C_{2,\delta}^\top \sigma &= \left(\sum_{j=0}^{k_\Psi-1} \operatorname{vec}\left(\tilde{U}_j\right)^\top \left(\sum_{i=1}^{m} u_{n,i} (\Lambda_{\Psi_j}^{[i]})^\top\right) C_{2,\delta}^\top \sigma\right)^\top \\
&= \sigma^\top C_{2,\delta} \sum_{j=0}^{k_\Psi-1} \sum_{i=1}^{m} u_{n,i} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right) \\
&= \sigma^\top C_{2,\delta} \sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right)\right) u_{n,i}, \\
&= \delta_a \sigma^\top C_{2,a} \sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right)\right) u_{n,a,i} + \\
&\quad + (1-\delta_a)\sigma^\top C_{2,r} \sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1} \Lambda_{\Psi_j}^{[i]} \operatorname{vec}\left(\tilde{U}_j\right)\right) u_{n,r,i},
\end{aligned} \tag{B.30}$$

$$\sum_{i=1}^{m} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)^\top u_{n,i} (\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top C_{2,\delta}^\top \sigma = \left(\sum_{i=1}^{m} \operatorname{vec}\left(\tilde{U}_{k_\Psi}^{[i]}\right)^\top u_{n,i} (\Lambda_{\Psi_{k_\Psi}}^{[i]})^\top C_{2,\delta}^\top \sigma\right)^\top$$

$$
\begin{aligned}
&= \sigma^\top C_{2,\delta} \sum_{i=1}^{m} \Lambda^{[i]}_{\Psi_{k_\Psi}} \operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right) u_{n,i} \\
&= \delta_a \sigma^\top C_{2,a} \sum_{i=1}^{m} \Lambda^{[i]}_{\Psi_{k_\Psi}} \operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right) u_{n,a,i} + \\
&\quad + (1-\delta_a) \sigma^\top C_{2,r} \sum_{i=1}^{m} \Lambda^{[i]}_{\Psi_{k_\Psi}} \operatorname{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right) u_{n,r,i}.
\end{aligned} \tag{B.31}
$$

Exploiting the above identities and substituting u_r as in (6.9b), expression (B.29) can be rewritten as

$$
\begin{aligned}
\dot{v} \leq\ & \sigma^\top C_{1,\delta} \left\{ \Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \right\} + \sigma^\top C_{2,\delta} \left\{ \Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + h \right\} + \\
& + \delta_a \sigma^\top C_{2,a} \left\{ \sum_{i=1}^{m} \left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi - 1} \Xi^{[i]}_{\Psi_j} \Delta_{\Psi_j} \right) u_{n,a,i} + \varepsilon_\Psi u_{n,a} \right\} + \\
& + (1-\delta_a) \sigma^\top C_{2,r} \left\{ \sum_{i=1}^{m} \left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi - 1} \Xi^{[i]}_{\Psi_j} \Delta_{\Psi_j} \right) u_{n,r,i} + \varepsilon_\Psi u_{n,r} \right\} - \rho \sigma^\top C_{2,\delta} \bar{B} \frac{\sigma}{\|\sigma\|}.
\end{aligned}
$$

Then, by means of Assumption 5.1, 5.3, 5.6, and Proposition 5.2, it holds that

$$
\begin{aligned}
\dot{v} \leq\ & \|\sigma\| \, \|C_{1,\delta}\| \left\{ \bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1} \right\} + \|\sigma\| \, \|C_{2,\delta}\| \left\{ \bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + \bar{h} \right\} + \\
& + \delta_a \, \|\sigma\| \, \|C_{2,a}\| \left\{ m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \, \|u_{n,a}\| \right\} + (1-\delta_a) \, \|\sigma\| \, \|C_{2,r}\| \left\{ m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \, \|u_{n,r}\| \right\} + \\
& - \rho \sigma^\top C_{2,\delta} \bar{B} \frac{\sigma}{\|\sigma\|}.
\end{aligned}
$$

Since matrix $\bar{B}$ is assumed to be symmetric and positive definite (Assumption 5.2) and the components of $C_{2,\delta}$ are chosen according to Assumption 6.1, it holds that

$$
\sigma^\top C_{2,\delta} \bar{B} \sigma \geq \|\sigma\|^2 \, \underline{\lambda}(C_{2,\delta}) \underline{\lambda}(\bar{B}) \geq \|\sigma\|^2 \, \underline{\lambda}(C_{2,\delta}) \underline{\gamma},
$$

where $\underline{\lambda}(C_{2,\delta}) \coloneqq \delta_a \underline{\lambda}(C_{2,a}) + (1-\delta_a) \underline{\lambda}(C_{2,r})$. Hence, $\dot{v}$ can be bounded as

$$
\begin{aligned}
\dot{v} \leq\ & \|\sigma\| \, \|C_{1,\delta}\| \left\{ \bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1} \right\} + \|\sigma\| \, \|C_{2,\delta}\| \left\{ \bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + \bar{h} \right\} + \\
& + \delta_a \, \|\sigma\| \, \|C_{2,a}\| \left\{ m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \, \|u_{n,a}\| \right\} + (1-\delta_a) \, \|\sigma\| \, \|C_{2,r}\| \left\{ m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi) \, \|u_{n,r}\| \right\} + \\
& - \rho \, \|\sigma\| \, \underline{\lambda}(C_{2,\delta}) \underline{\gamma}.
\end{aligned}
$$

If one defines the quantities $\bar{r}_{\Phi_1} \coloneqq \bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1}$, $\bar{r}_{\Phi_2} \coloneqq \bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2}$ and $\bar{r}_\Psi \coloneqq m(\bar{c}_\Psi + \bar{\varepsilon}_\Phi)$, the above inequality can be written in a more compact way as

$$
\begin{aligned}
\dot{v} \leq\ & \|\sigma\| \left\{ \|C_{1,\delta}\| \, \bar{r}_{\Phi_1} + \|C_{2,\delta}\| \left(\bar{r}_{\Phi_2} + \bar{h} \right) + \delta_a \, \|C_{2,a}\| \, \bar{r}_\Psi \, \|u_{n,a}\| + \right. \\
& \left. + (1-\delta) \, \|C_{2,r}\| \, \bar{r}_\Psi \, \|u_{n,r}\| - \rho \underline{\lambda}(C_{2,\delta}) \underline{\gamma} \right\}.
\end{aligned}
$$

If the control gain ρ is chosen as in Theorem 6.1, i.e.,

$$
\rho > \frac{\|C_{1,\delta}\| \, \bar{r}_{\Phi_1} + \|C_{2,\delta}\| \left(\bar{r}_{\Phi_2} + \bar{h} \right) + \delta_a \, \|C_{2,a}\| \, \bar{r}_\Psi \, \|u_{n,a}\| + (1-\delta_a) \, \|C_{2,r}\| \, \bar{r}_\Psi \, \|u_{n,r}\| + \bar{\eta}}{\underline{\lambda}(C_{2,\delta}) \underline{\gamma}},
$$

with $\bar{\eta} \in \mathbb{R}_{>0}$, then it holds that

$$\dot{v} \leq -\bar{\eta} \left\| \sigma \right\|,$$

implying that $\sigma(x(t)) \to 0_m$ for $t \to \infty$. This concludes the proof.

B.4 Proof of Theorem 6.2

The development of proof is similar to the proof of Theorem 5.2 and follows the one in [53, Theorem 3]. Consider the Lyapunov candidate function $v : \mathcal{X} \to \mathbb{R}$ given by

$$v = \frac{1}{2}\sigma^\top \sigma + \frac{1}{2}\sum_{i=1}^{2} \tilde{\rho}_i^2, \tag{B.32}$$

where $\tilde{\rho}_i = \rho_i^\star - \hat{\rho}_i$ represent the components the error between the estimate of the discontinuous control gain and its ideal value

$$\rho^\star = \rho_1^\star + \rho_2^\star\Big(\delta_a \left\| u_{n,a} \right\| + (1-\delta_a)\left\| u_{n,r} \right\|\Big) + \frac{\eta^\star}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}},$$

with $\eta^\star \in \mathbb{R}_{>0}$. In particular, the elements $\rho_i^\star$ are defined so that $\rho^\star$ is able to enforce sliding mode $\sigma(x(t)) = 0_m$ for $t \leq t_1$. Hence,

$$\rho_1^\star = \frac{\left\| C_{1,\delta} \right\| \left(\sup_{x\in\mathcal{X}} \left\| f_1 - \hat{\Phi}^{[1]}_{k_\Phi,t_1} \right\| \right) + \left\| C_{2,\delta} \right\| \left(\sup_{x\in\mathcal{X}} \left\| f_2 - \hat{\Phi}^{[2]}_{k_\Phi,t_1} \right\| + \bar{h} \right)}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}}, \tag{B.33}$$

$$\rho_2{}^\star = \frac{\left\| C_{2,\delta} \right\| \sup_{x\in\mathcal{X}} \left\| \sum_{i=1}^{m} \bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi,t_1} \right\|}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}}, \tag{B.34}$$

with $\hat{\Phi}^{[1]}_{k_\Phi,t_1}, \hat{\Phi}^{[2]}_{k_\Phi,t_1}$, and $\hat{\Psi}^{[i]}_{k_\Psi,t_1}$ denoting the output of the DNNs characterized by fixed weights $\hat{V}_j(t_1)$, for $j \in \{0, 1, \ldots, k_\Phi\}$ and $\hat{U}_j(t_1)$, for $j \in \{0, 1, \ldots, k_\Psi\}$.

Exploiting (B.24), the first time-derivative of (B.32) is computed as

$$\begin{aligned}
\dot{v}(x) &= \sigma^\top \dot{\sigma} - \sum_{i=1}^{2} \tilde{\rho}_i \dot{\hat{\rho}}_i \\
&= \sigma^\top \Bigg[C_{1,\delta}\Big(\tilde{\Phi}^{[1]}_{k_{\Phi,t_1}} + \varepsilon^{[1]}_\Phi\Big) + C_{2,\delta}\Big(\tilde{\Phi}^{[2]}_{k_{\Phi,t_1}} + \varepsilon^{[2]}_\Phi + h\Big) + \\
&\quad + \delta_a C_{2,a}\left(\sum_{i=1}^{m} \tilde{\Psi}^{[i]}_{k_{\Psi,t_1}} u_{n,a,i} + \varepsilon_\Psi u_{n,a} \right) + (1-\delta_a) C_{2,r}\left(\sum_{i=1}^{m} \tilde{\Psi}^{[i]}_{k_{\Psi,t_1}} u_{n,r,i} + \varepsilon_\Psi u_{n,r} \right) \Bigg] + \\
&\quad - \sigma^\top C_{2,\delta} \bar{B} \left(\hat{\rho}_1 + \hat{\rho}_2\Big(\delta_a \left\| u_{n,a} \right\| + (1-\delta_a)\left\| u_{n,r} \right\|\Big) \right) \frac{\sigma}{\left\| \sigma \right\|} - \tilde{\rho}_1 \dot{\hat{\rho}}_1 - \tilde{\rho}_2 \dot{\hat{\rho}}_2,
\end{aligned}$$

where $\tilde{\Phi}^{[1]}_{k_\Phi,t_1} = \Phi^{[1]}_{k_\Phi} - \hat{\Phi}^{[1]}_{k_\Phi,t_1}$, $\tilde{\Phi}^{[2]}_{k_\Phi,t_1} = \Phi^{[2]}_{k_\Phi} - \hat{\Phi}^{[2]}_{k_\Phi,t_1}$, and $\tilde{\Psi}^{[i]}_{k_\Psi,t_1} = \Psi^{[i]}_{k_\Psi} - \hat{\Psi}^{[1]}_{k_\Psi,t_1}$. Recalling that $\hat{\rho} = \rho^\star - \tilde{\rho}$, $\dot{v}(x)$ can be expanded as

$$\dot{v}(x) = \sigma^\top \Bigg[C_{1,\delta}\Big(\tilde{\Phi}^{[1]}_{k_{\Phi,t_1}} + \varepsilon^{[1]}_\Phi\Big) + C_{2,\delta}\Big(\tilde{\Phi}^{[2]}_{k_{\Phi,t_1}} + \varepsilon^{[2]}_\Phi + h\Big) +$$

$$+ \delta_a C_{2,a}\left(\sum_{i=1}^{m} \tilde{\Psi}^{[i]}_{k_\Psi,t_1} u_{n,a,i} + \varepsilon_\Psi u_{n,a}\right) + (1-\delta_a) C_{2,r}\left(\sum_{i=1}^{m} \tilde{\Psi}^{[i]}_{k_\Psi,t_1} u_{n,r,i} + \varepsilon_\Psi u_{n,r}\right)\Bigg] +$$
$$- \sigma^\top C_{2,\delta} \bar{B}\left(\rho_1^\star + \rho_2^\star\Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\| + \frac{\eta^\star}{\underline{\lambda}(C_{2,\delta})\underline{\gamma}}\Big)\right)\frac{\sigma}{\|\sigma\|} +$$
$$+ \sigma^\top C_{2,\delta} \bar{B}\left(\tilde{\rho}_1 + \tilde{\rho}_2\Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big)\right)\frac{\sigma}{\|\sigma\|} +$$
$$- \tilde{\rho}_1\dot{\tilde{\rho}}_1 - \tilde{\rho}_2\dot{\tilde{\rho}}_2,$$

Exploiting Proposition 5.3 to bound the approximation errors and bounding the other terms relying on Assumptions 5.2 and 5.3, it holds that

$$\dot{v}(x) \leq \|\sigma\| \|C_{1,\delta}\| \sup_{x\in\mathcal{X}} \left\| f_1 - \hat{\Phi}^{[1]}_{k_\Phi,t_1}\right\| + \|\sigma\| \|C_{2,\delta}\| \sup_{x\in\mathcal{X}} \left\| f_2 - \hat{\Phi}^{[2]}_{k_\Phi,t_1}\right\| + \|\sigma\| \|C_{2,\delta}\| \bar{h} +$$
$$+ \delta_a \|\sigma\| \|C_{2,a}\| \sup_{x\in\mathcal{X}} \left\| \sum_{i=1}^{m} \bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi,t_1}\right\| \|u_{n,a}\| +$$
$$+ (1-\delta_a)\|\sigma\| \|C_{2,r}\| \sup_{x\in\mathcal{X}} \left\| \sum_{i=1}^{m} \bar{B}^{(i)} - \hat{\Psi}^{[i]}_{k_\Psi,t_1}\right\| \|u_{n,r}\| +$$
$$- \|\sigma\| \underline{\lambda}(C_{2,\delta})\underline{\gamma}\rho_1^\star - \|\sigma\| \underline{\lambda}(C_{2,\delta})\underline{\gamma}\rho_2^\star\Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big) - \|\sigma\|\eta^\star +$$
$$+ \|\sigma\| \|C_{2,\delta}\| \bar{\gamma} \|\tilde{\rho}_1\| + \|\sigma\| \|C_{2,\delta}\| \bar{\gamma} \|\tilde{\rho}_2\| \Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big) +$$
$$- \tilde{\rho}_1\dot{\tilde{\rho}}_1 - \tilde{\rho}_2\dot{\tilde{\rho}}_2.$$

Substituting the values $\rho_1^\star$ and $\rho_2^\star$ as in (B.33) and (B.34), respectively, the above inequality can be simplified as

$$\dot{v}(x) \leq \|\sigma\| \|C_{2,\delta}\| \bar{\gamma} \|\tilde{\rho}_1\| + \|\sigma\| \|C_{2,\delta}\| \bar{\gamma} \|\tilde{\rho}_2\| \Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big) +$$
$$- \tilde{\rho}_1\dot{\tilde{\rho}}_1 - \tilde{\rho}_2\dot{\tilde{\rho}}_2 - \|\sigma\|\eta^\star.$$

Substituting the adaptation laws (6.19), it holds that

$$\dot{v}(x) \leq \|\sigma\| \|C_{2,\delta}\| \bar{\gamma} \|\tilde{\rho}_1\| + \|\sigma\| \|C_{2,\delta}\| \bar{\gamma} \|\tilde{\rho}_2\| \Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big) +$$
$$- \tilde{\rho}_1 \underset{\varrho_1}{\text{proj}}\left(\Big(\|C_{2,\delta}\| \bar{\gamma} + \alpha_1\Big)\|\sigma\|\right) +$$
$$- \tilde{\rho}_2 \underset{\varrho_2}{\text{proj}}\left(\left(\|C_{2,\delta}\| \bar{\gamma}\Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big) + \alpha_2\right)\|\sigma\|\right) +$$
$$- \|\sigma\|\eta^\star.$$

Letting $\rho_i^\star > \bar{\rho}_i$, since $\hat{\rho}_i \leq \bar{\rho}_i$ by virtue of the projection operator, then $\tilde{\rho}_i \equiv \|\tilde{\rho}_i\|$, for $i \in \{1,2\}$. Hence, if one exploits the property of the project operator in Lemma A.1, it is possible to write

$$\dot{v}(x) \leq \|\sigma\| \|C_{2,\delta}\| \bar{\gamma}\tilde{\rho}_1 + \|\sigma\| \|C_{2,\delta}\| \bar{\gamma}\tilde{\rho}_2\Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big) +$$
$$- \tilde{\rho}_1\Big(\|C_{2,\delta}\| \bar{\gamma} + \alpha_1\Big)\|\sigma\| +$$

$$
\begin{aligned}
&- \tilde{\rho}_2 \left(\|C_{2,\delta}\| \, \bar{\gamma}\Big(\delta_a \|u_{n,a}\| + (1-\delta_a)\|u_{n,r}\|\Big) + \alpha_2 \right) \|\sigma\| - \|\sigma\| \, \eta^\star \\
&\leq - \|\sigma\| \, \alpha_1 \tilde{\rho}_1 - \|\sigma\| \, \alpha_2 \tilde{\rho}_2 - \|\sigma\| \, \eta^\star \\
&\leq -\eta_1 \|\tilde{\rho}_1\| - \eta_2 \|\tilde{\rho}_2\| - \eta^\star \|\sigma\| \\
&\leq -\sqrt{2} \min\{\eta^\star, \eta_1, \eta_2\} \left(\frac{\|\sigma\|}{\sqrt{2}} + \frac{\|\tilde{\rho}_1\|}{\sqrt{2}} + \frac{\|\tilde{\rho}_2\|}{\sqrt{2}} \right) \\
&\leq -\sqrt{2} \min\{\eta^\star, \eta_1, \eta_2\} \sqrt{v(x)},
\end{aligned}
$$

implying that there exist $t_2 \geq t_1$ such that $\sigma(x(t)) = 0_m$ for $t \geq t_2$ [53]. This concludes the proof.

B.5 Proof of Theorem 6.3

Consider the Lyapunov candidate function $v : \mathcal{X} \to \mathbb{R}$ defined as

$$
v(x) = \frac{1}{2}\beta(\sigma) + \frac{1}{2}\sum_{j=0}^{k_\Phi} \text{vec}\left(\tilde{V}_j\right)^\top \Gamma_{\Phi_j}^{-1} \text{vec}\left(\tilde{V}_j\right) + \frac{1}{2}\sum_{j=0}^{k_\Psi} \text{vec}\left(\tilde{U}_j\right)^\top \Gamma_{\Psi_j}^{-1} \text{vec}\left(\tilde{U}_j\right). \tag{B.35}
$$

Then, its first time-derivative can be computed observing that, according to the chain rule, it holds that

$$
\dot{\beta} = \frac{d\beta}{dt} = \frac{d\beta}{d\|\sigma\|^2} \frac{d\|\sigma\|^2}{dt} = \frac{d\beta}{d\|\sigma\|^2} \frac{d(\sigma^\top \sigma)}{dt} = 2\frac{d\beta}{d\|\sigma\|^2} \sigma^\top \dot{\sigma}.
$$

Hence, $\dot{v}$ is equal to

$$
\dot{v} = \frac{d\beta}{d\|\sigma\|^2} \sigma^\top \dot{\sigma} + \sum_{j=0}^{k_\Phi} \text{vec}\left(\tilde{V}_j\right)^\top \Gamma_{\Phi_j}^{-1} \text{vec}\left(\dot{\tilde{V}}_j\right) + \sum_{j=0}^{k_\Psi} \text{vec}\left(\tilde{U}_j\right)^\top \Gamma_{\Psi_j}^{-1} \text{vec}\left(\dot{\tilde{U}}_j\right). \tag{B.36}
$$

From now on, the proof articulates following a reasoning which is similar to the one of the proof of Theorem 5.1 (Appendix B.1). Hence only the fundamental step are reported.

Since σ is defined as in (6.51), it holds that

$$
\begin{aligned}
\dot{\sigma} &= \dot{x} - \hat{\Phi}_{k_\Phi} - \sum_{i=1}^{m} \hat{\Psi}_{k_\Psi}^{[i]} u_{\text{qp},i} \\
&= f + \bar{B} u_{\text{qp}} - \rho \bar{B} \frac{\sigma}{\|\sigma\|} + h - \hat{\Phi}_{k_\Phi} - \sum_{i=1}^{m} \hat{\Psi}_{k_\Psi}^{[i]} u_{\text{qp},i} \\
&= \tilde{\Phi}_{k_\Phi} + \sum_{i=1}^{m} \tilde{\Psi}_{k_\Psi}^{[i]} u_{\text{qp},i} + \varepsilon_\Phi + \varepsilon_\Psi u_{\text{qp}} + h - \rho \bar{B} \frac{\sigma}{\|\sigma\|}.
\end{aligned} \tag{B.37}
$$

Substituting (B.37) in (B.36), it holds that

$$
\begin{aligned}
\dot{v} = \frac{d\beta}{d\|\sigma\|^2} \sigma^\top \left\{ \tilde{\Phi}_{k_\Phi} + \sum_{i=1}^{m} \tilde{\Psi}_{k_\Psi}^{[i]} u_{\text{qp},i} + \varepsilon_\Phi + \varepsilon_\Psi u_{\text{qp}} + h \right\} - \rho \frac{d\beta}{d\|\sigma\|^2} \sigma^\top \bar{B} \frac{\sigma}{\|\sigma\|} + \\
- \sum_{j=0}^{k_\Phi} \text{vec}\left(\tilde{V}_j\right)^\top \Gamma_{\Phi_j}^{-1} \text{vec}\left(\dot{\tilde{V}}_j\right) - \sum_{j=0}^{k_\Psi} \text{vec}\left(\tilde{U}_j\right)^\top \Gamma_{\Psi_j}^{-1} \text{vec}\left(\dot{\tilde{U}}_j\right).
\end{aligned}
$$

Writing the expression of the DNNs approximation errors, the above equation becomes

$$\begin{aligned}\dot{v} = \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Bigg\{&\Lambda_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \Delta_{\Phi_{k_\Phi}} + \sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}\Delta_{\Phi_j} + \\ &+ \sum_{i=1}^{m}\left(\Lambda^{[i]}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right) + \Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\mathrm{vec}\left(\tilde{U}_j\right) + \sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\right)u_{\mathrm{qp},i} + \\ &+ \varepsilon_\Phi + \varepsilon_\Psi u_{\mathrm{qp}} + h\Bigg\} - \rho\frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\bar{B}\frac{\sigma}{\left\|\sigma\right\|} - \sum_{j=0}^{k_\Phi}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Gamma^{-1}_{\Phi_j}\mathrm{vec}\left(\dot{\hat{V}}_j\right) + \\ &- \sum_{j=0}^{k_\Psi}\mathrm{vec}\left(\tilde{U}_j\right)^\top\Gamma^{-1}_{\Psi_j}\mathrm{vec}\left(\dot{\hat{U}}_j\right).\end{aligned}$$

Rearranging the terms, one obtains

$$\begin{aligned}\dot{v} = \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Bigg\{&\Delta_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}\Delta_{\Phi_j} + \sum_{i=1}^{m}\left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\right)u_{\mathrm{qp},i} + \\ &+ \varepsilon_\Phi + \varepsilon_\Psi u_{\mathrm{qp}} + h\Bigg\} - \rho\frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\bar{B}\frac{\sigma}{\left\|\sigma\right\|} + \\ &+ \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Lambda_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) + \\ &+ \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{i=1}^{m}\Lambda^{[i]}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)u_{\mathrm{qp},i} + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\mathrm{vec}\left(\tilde{U}_j\right)\right)u_{\mathrm{qp},i} + \\ &- \sum_{j=0}^{k_\Phi}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Gamma^{-1}_{\Phi_j}\mathrm{vec}\left(\dot{\hat{V}}_j\right) - \sum_{j=0}^{k_\Psi}\mathrm{vec}\left(\tilde{U}_j\right)^\top\Gamma^{-1}_{\Psi_j}\mathrm{vec}\left(\dot{\hat{U}}_j\right).\end{aligned} \tag{B.38}$$

Then, if one expands the last two term, the above equation becomes

$$\begin{aligned}\dot{v} = \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Bigg\{&\Delta_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}\Delta_{\Phi_j} + \sum_{i=1}^{m}\left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\right)u_{\mathrm{qp},i} + \\ &+ \varepsilon_\Phi + \varepsilon_\Psi u_{\mathrm{qp}} + h\Bigg\} - \rho\frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\bar{B}\frac{\sigma}{\left\|\sigma\right\|} + \\ &+ \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Lambda_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) + \\ &+ \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{i=1}^{m}\Lambda^{[i]}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)u_{\mathrm{qp},i} + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\mathrm{vec}\left(\tilde{U}_j\right)\right)u_{\mathrm{qp},i} + \\ &- \sum_{j=0}^{k_\Phi-1}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Gamma^{-1}_{\Phi_j}\mathrm{vec}\left(\dot{\hat{V}}_j\right) - \mathrm{vec}\left(\tilde{V}_{k_\Phi}\right)^\top\left(\Gamma_{\Phi_{k_\Phi}}\right)^{-1}\mathrm{vec}\left(\dot{\hat{V}}_{k_\Phi}\right) + \\ &- \sum_{j=0}^{k_\Psi-1}\mathrm{vec}\left(\tilde{U}_j\right)^\top\Gamma^{-1}_{\Psi_j}\mathrm{vec}\left(\dot{\hat{U}}_j\right) - \sum_{i=1}^{m}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top\Gamma^{-1}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\dot{\hat{U}}^{[i]}_{k_\Psi}\right).\end{aligned} \tag{B.39}$$

Substituting the adaptation laws (6.53), (6.54), and (6.55), and exploiting the property of

the projection operator Lemma 5.2, it holds that

$$\begin{aligned}\dot{v} \leq \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Bigg\{\Delta_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}\Delta_{\Phi_j} + \sum_{i=1}^{m}\left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\right)u_{\mathrm{qp},i} + \\ + \varepsilon_\Phi + \varepsilon_\Psi u_{\mathrm{qp}} + h\Bigg\} - \rho\frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\bar{B}\frac{\sigma}{\left\|\sigma\right\|} + \\ + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Lambda_{\Phi_{k_\Phi}}\mathrm{vec}\left(\tilde{V}_{k_\Phi}\right) + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) + \\ + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{i=1}^{m}\Lambda^{[i]}_{\Psi_{k_\Psi}}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)u_{\mathrm{qp},i} + \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\sum_{i=1}^{m}\left(\sum_{j=0}^{k_\Psi-1}\Lambda^{[i]}_{\Psi_j}\mathrm{vec}\left(\tilde{U}_j\right)\right)u_{\mathrm{qp},i} + \\ - \sum_{j=0}^{k_\Phi-1}\mathrm{vec}\left(\tilde{V}_j\right)^\top\left(\Lambda_{\Phi_j}\right)^\top\frac{\partial\beta}{\partial\left\|\sigma\right\|^2}\sigma - \mathrm{vec}\left(\tilde{V}_{k_\Phi}\right)^\top\left(\Lambda_{\Phi_{k_\Phi}}\right)^\top\frac{\partial\beta}{\partial\left\|\sigma\right\|^2}\sigma + \\ - \sum_{j=0}^{k_\Psi-1}\mathrm{vec}\left(\tilde{U}_j\right)^\top\left(\sum_{i=1}^{m}u_{\mathrm{qp},i}(\Lambda^{[i]}_{\Psi_j})^\top\right)\frac{\partial\beta}{\partial\left\|\sigma\right\|^2}\sigma + \\ - \sum_{i=1}^{m}\mathrm{vec}\left(\tilde{U}^{[i]}_{k_\Psi}\right)^\top u_{\mathrm{qp},i}(\Lambda^{[i]}_{\Psi_j})^\top\frac{\partial\beta}{\partial\left\|\sigma\right\|^2}\sigma. \end{aligned} \tag{B.40}$$

Then, since $\dot{v} \in \mathbb{R}$, it is possible to cancel out the terms that depend on the weight error, having

$$\begin{aligned}\dot{v} \leq \frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\Bigg\{\Delta_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}\Delta_{\Phi_j} + \sum_{i=1}^{m}\left(\Delta^{[i]}_{\Psi_{k_\Psi}} + \sum_{j=1}^{k_\Psi-1}\Xi^{[i]}_{\Psi_j}\Delta_{\Psi_j}\right)u_{\mathrm{qp},i} + \\ + \varepsilon_\Phi + \varepsilon_\Psi u_{\mathrm{qp}} + h\Bigg\} - \rho\frac{d\beta}{d\left\|\sigma\right\|^2}\sigma^\top\bar{B}\frac{\sigma}{\left\|\sigma\right\|}. \end{aligned} \tag{B.41}$$

Since $\frac{\partial\beta}{\partial\|\sigma\|^2} \in \mathbb{R}_{>0}$, exploiting Assumption 5.1, 5.3, and 5.6, and Proposition 5.2, the above quantity can be bounded as

$$\dot{v} \leq \left|\frac{d\beta}{d\left\|\sigma\right\|^2}\right|\left\|\sigma\right\|\left\{\bar{c}_\Phi + \bar{\varepsilon}_\Phi + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi)\left\|u_{\mathrm{qp}}\right\| + \bar{h}\right\} - \rho\left|\frac{d\beta}{d\left\|\sigma\right\|^2}\right|\underline{\gamma}\left\|\sigma\right\|. \tag{B.42}$$

Hence, choosing

$$\rho = \frac{\bar{c}_\Phi + \bar{\varepsilon}_\Phi + m(\bar{c}_\Psi + \bar{\varepsilon}_\Psi)\left\|u_{\mathrm{qp}}\right\| + \bar{h} + \bar{\eta}}{\underline{\gamma}}, \tag{B.43}$$

with $\bar{\eta} \in \mathbb{R}_{>0}$ being a design parameter, then it holds that

$$\dot{v} \leq -\bar{\eta}\left|\frac{d\beta}{d\left\|\sigma\right\|^2}\right|\left\|\sigma\right\|, \tag{B.44}$$

implying that $\sigma(x(t)) \to 0_m$ for $t \to \infty$. Moreover, since $\sigma(x(t_0)) = 0_m$, it is guaranteed that $\|\sigma(x(t))\| \leq \varepsilon_\sigma$, for $t \geq t_0$ by virtue of the BLF. This concludes the proof.

B.6 Proof of Theorem 6.4

By Assumption 6.2 and from Nagumo's theorem [154], it holds that the QP problem (6.58) ensures that $\hat{\mathcal{X}}_a$ is forward invariant with respect to the state of the system whose dynamics

are defined by the DNNs, which is

$$\dot{\hat{x}}(t) = \hat{\Phi}_{k_\Phi} + \mathrm{vec}^{-1}\left(\hat{\Psi}_{k_\Psi}\right) u_{\mathrm{qp}}.$$

Hence, it holds that $\hat{x}(t) \in \hat{\mathcal{X}}_a$, for $t \geq t_0$. From (6.57), the boundary $\partial\hat{\mathcal{X}}_a$ of $\hat{\mathcal{X}}_a$ is given by $\partial\hat{\mathcal{X}}_a = \{x \in \mathcal{X}_a : \mathrm{dist}(x, \partial\mathcal{X}_a) = \varepsilon_\sigma\}$, and thus $\mathrm{dist}(\partial\hat{\mathcal{X}}_a, \partial\mathcal{X}_a) = \varepsilon_\sigma$. Then, the distance between the state trajectory x and $\partial\mathcal{X}_a$, can be lower bounded by exploiting the triangular inequality as

$$\begin{aligned}\mathrm{dist}(x(t), \partial\mathcal{C}) &\geq \mathrm{dist}(\hat{x}(t), \partial\mathcal{X}_a) - \mathrm{dist}(x(t), \hat{x}(t)) \\ &> \mathrm{dist}(\partial\hat{\mathcal{X}}_a, \partial\mathcal{X}_a) - \varepsilon_\sigma = 0\end{aligned}$$

No further step is necessary, as $\hat{x}$ can only approach $\partial\hat{\mathcal{X}}_a$ from the interior of $\hat{\mathcal{X}}_a$, concluding the proof.

B.7 Proof of Theorem 7.1

Let the Lyapunov candidate function $v : \mathcal{X} \to \mathbb{R}$ be defined as

$$v = \frac{1}{2}\sigma^\top\sigma. \tag{B.45}$$

Considering the choice of σ_0 in (7.5), and $\dot{z}$ in (7.6), one has that

$$\begin{aligned}\dot{v} &= \sigma^\top\dot{\sigma} \\ &= \sigma^\top\{\dot{\sigma}_0 - \dot{z}\} \\ &= \sigma^\top\Bigg\{C_1\Big[f_1(x) - f_1(\hat{x})\Big] + C_2\Big[f_2(x) - f_2(\hat{x})\Big] + C_2\Big[\Big(\bar{B}(x) - \bar{B}(\hat{x})\Big)u + \\ &\quad + \bar{B}(x)\Delta u - \bar{B}(\hat{x})v_n - \bar{B}(\hat{x})v_r\Big] - C_1\Big[f_1(x) - f_1(\hat{x})\Big] - C_2\Big[f_2(x) - f_2(\hat{x})\Big] + \\ &\quad C_2\Big[\Big(\bar{B}(x) - \bar{B}(\hat{x})\Big)u - \bar{B}(\hat{x})v_n\Big]\Bigg\} \\ &= \sigma^\top C_2\Big[\bar{B}(x)\Delta u - \bar{B}(\hat{x})v_r\Big]\end{aligned}$$

Substituting the control law v_r in (7.4) and rearranging the terms, it holds that

$$\dot{v} = \sigma^\top C_2\bar{B}(x)\Delta u - \rho\sigma^\top C_2\bar{B}(\hat{x})\frac{\sigma}{\|\sigma\|}.$$

Then, exploiting Assumptions 5.2 and 7.1, $\dot{v}$ can be bounded as

$$\dot{v} = \|\sigma\|\,\|C_2\|\,\bar{\gamma}\bar{\delta} - \rho\,\|\sigma\|\,\underline{\lambda}(C_2)\underline{\gamma}.$$

Hence, choosing the discontinuous control gain such that

$$\rho > \frac{\|C_2\|\,\bar{\gamma}}{\underline{\lambda}(C_2)\underline{\gamma}}\bar{\delta}, \tag{B.46}$$

it holds that $\dot{v} < 0$, meaning that the condition $\sigma(x(t)) = 0_m$ is enforced in finite time. Moreover, since $\sigma(x(t_0)) = 0_m$ by design, a sliding mode $\sigma(x(t)) = 0_m$ is enforced for $t \geq t_0$.

When in sliding mode, it is possible to compute the equivalent controller v_{eq} by imposing $\dot{\sigma} = 0_m$ and solving for v_r. In this case,

$$\dot{\sigma} = C_2\Big[\bar{B}(x)\Delta u - \bar{B}(\hat{x})\tilde{v}_r\Big] = 0_m,$$

which leads to

$$v_{\text{eq}} = (\bar{B}(\hat{x}))^{+}\bar{B}(x)\Delta u,$$

concluding the proof.

B.8 Proof of Theorem 7.2

Consider the Lyapunov-like candidate function $v : \mathcal{X} \to \mathbb{R}$ defined as

$$v = \frac{1}{2}\sigma^\top\sigma + \frac{1}{2}\sum_{j=0}^{k_\Phi}\text{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\text{vec}\left(\tilde{V}_j\right), \tag{B.47}$$

characterized by first time-derivative

$$\begin{aligned}\dot{v} &= \sigma^\top\dot{\sigma} + \sum_{j=0}^{k_\Phi}\text{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\text{vec}\left(\dot{\tilde{V}}_j\right)\\ &= \sigma^\top\{\dot{\sigma}_0 - \dot{\hat{z}}\} - \sum_{j=0}^{k_\Phi}\text{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\text{vec}\left(\dot{\tilde{V}}_j\right)\end{aligned}$$

Considering the sliding variable σ in (7.9), σ_0 in (7.5), and $\dot{\hat{z}}$ in (7.10), and defining $f_1(x) \equiv f_1$, $f_2(x) \equiv f_2$, $\bar{B}(x) \equiv \bar{B}$, and $\bar{B}(\hat{x}) \equiv \hat{\bar{B}}$ for sake of readability, one has that

$$\begin{aligned}\dot{v} &= \sigma^\top\Bigg\{C_1 f_1 + C_2\Big[f_2 - \Big(\bar{B} - \hat{\bar{B}}\Big)u + \bar{B}\Delta u - \hat{\bar{B}}v_n - \hat{\bar{B}}v_r\Big] - C_1\hat{\Phi}_{k_\Phi}^{[1]} - C_2\Big[\hat{\Phi}_{k_\Phi}^{[2]} +\\ &\quad + \Big(\bar{B} - \hat{\bar{B}}\Big)u - \hat{\bar{B}}v_n\Big]\Bigg\} - \sum_{j=0}^{k_\Phi}\text{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\text{vec}\left(\dot{\tilde{V}}_j\right)\\ &= \sigma^\top\Bigg\{C_1\Big[f_1 - \hat{\Phi}_{k_\Phi}^{[1]}\Big] + C_2\Big[f_2 - \hat{\Phi}_{k_\Phi}^{[2]} + \bar{B}\Delta u - \hat{\bar{B}}v_r\Big]\Bigg\} - \sum_{j=0}^{k_\Phi}\text{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\text{vec}\left(\dot{\tilde{V}}_j\right)\\ &= \sigma^\top\Bigg\{C_1\Big[\tilde{\Phi}_{k_\Phi}^{[1]} + \varepsilon_\Phi^{[1]}\Big] + C_2\Big[\tilde{\Phi}_{k_\Phi}^{[2]} + \varepsilon_\Phi^{[2]} + \bar{B}\Delta u - \hat{\bar{B}}v_r\Big]\Bigg\} - \sum_{j=0}^{k_\Phi}\text{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\text{vec}\left(\dot{\tilde{V}}_j\right).\end{aligned}$$

Substituting $\tilde{\Phi}_{k_\Phi}^{[1]}$ and $\tilde{\Phi}_{k_\Phi}^{[2]}$ as in (5.29), one has that

$$\begin{aligned}\dot{v} = \sigma^\top\Bigg\{&C_1\Big[\Lambda_{\Phi_{k_\Phi}}^{[1]}\text{vec}\left(\tilde{V}_{k_\Phi}^{[1]}\right) + \Delta_{\Phi_{k_\Phi}}^{[1]} + \sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}^{[1]}\text{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}^{[1]}\Delta_{\Phi_j} + \varepsilon_\Phi^{[1]}\Big]\\ &+ C_2\Big[\Lambda_{\Phi_{k_\Phi}}^{[2]}\text{vec}\left(\tilde{V}_{k_\Phi}^{[2]}\right) + \Delta_{\Phi_{k_\Phi}}^{[2]} + \sum_{j=0}^{k_\Phi-1}\Lambda_{\Phi_j}^{[2]}\text{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi-1}\Xi_{\Phi_j}^{[2]}\Delta_{\Phi_j} + \varepsilon_\Phi^{[2]}\\ &+ \bar{B}\Delta u - \hat{\bar{B}}v_r\Big]\Bigg\} - \sum_{j=0}^{k_\Phi}\text{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\text{vec}\left(\dot{\tilde{V}}_j\right).\end{aligned}$$

Then, rearranging the terms and expanding the last summation, it is possible to obtain

$$\begin{aligned}\dot{v} = \sigma^\top \Bigg\{ C_1 \Big[\Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \Big] + C_2 \Big[\Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \bar{B} \Delta u \Big] \Bigg\} + \\ - \sigma^\top C_2 \hat{\bar{B}} v_2 + \sigma^\top C_1 \Lambda^{[1]}_{\Phi_{k_\Phi}} \operatorname{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right) + \sigma^\top C_1 \sum_{j=0}^{k_\Phi - 1} \Lambda^{[1]}_{\Phi_j} \operatorname{vec}\left(\tilde{V}_j \right) + \\ + \sigma^\top C_2 \Lambda^{[2]}_{\Phi_{k_\Phi}} \operatorname{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right) + \sigma^\top C_2 \sum_{j=0}^{k_\Phi - 1} \Lambda^{[2]}_{\Phi_j} \operatorname{vec}\left(\tilde{V}_j \right) - \sum_{j=0}^{k_\Phi - 1} \operatorname{vec}\left(\tilde{V}_j \right)^\top \Gamma^{-1}_{\Phi_j} \operatorname{vec}\left(\dot{\tilde{V}}_j \right) + \\ - \operatorname{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right)^\top \left(\Gamma^{[1]}_{\Phi_{k_\Phi}} \right)^{-1} \operatorname{vec}\left(\dot{\tilde{V}}^{[1]}_{k_\Phi} \right) - \operatorname{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right)^\top \left(\Gamma^{[2]}_{\Phi_{k_\Phi}} \right)^{-1} \operatorname{vec}\left(\dot{\tilde{V}}^{[2]}_{k_\Phi} \right).\end{aligned}$$

Adjusting the the weights according to (7.11), (7.12a), and (7.12b), and exploiting property 3 of Lemma 5.2, the above equation becomes

$$\begin{aligned}\dot{v} \leq \sigma^\top \Bigg\{ C_1 \Big[\Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \Big] + C_2 \Big[\Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \bar{B} \Delta u \Big] \Bigg\} + \\ - \sigma^\top C_2 \hat{\bar{B}} v_r + \sigma^\top C_1 \Lambda^{[1]}_{\Phi_{k_\Phi}} \operatorname{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right) + \sigma^\top C_1 \sum_{j=0}^{k_\Phi - 1} \Lambda^{[1]}_{\Phi_j} \operatorname{vec}\left(\tilde{V}_j \right) + \\ + \sigma^\top C_2 \Lambda^{[2]}_{\Phi_{k_\Phi}} \operatorname{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right) + \sigma^\top C_2 \sum_{j=0}^{k_\Phi - 1} \Lambda^{[2]}_{\Phi_j} \operatorname{vec}\left(\tilde{V}_j \right) - \sum_{j=0}^{k_\Phi - 1} \operatorname{vec}\left(\tilde{V}_j \right)^\top \Lambda^{\top}_{\Phi_j} C^\top \sigma + \\ - \operatorname{vec}\left(\tilde{V}^{[1]}_{k_\Phi} \right)^\top (\Lambda^{[1]}_{\Phi_{k_\Phi}})^\top C_1^\top \sigma - \operatorname{vec}\left(\tilde{V}^{[2]}_{k_\Phi} \right)^\top (\Lambda^{[2]}_{\Phi_{k_\Phi}})^\top C_2^\top \sigma.\end{aligned}$$

Exploiting the fact that $\dot{v} \in \mathbb{R}$ and since (B.9) holds, one can simplify all the terms that depend on the weight approximation errors, obtaining

$$\begin{aligned}\dot{v} \leq \sigma^\top \Bigg\{ C_1 \Big[\Delta^{[1]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[1]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[1]}_{\Phi} \Big] + C_2 \Big[\Delta^{[2]}_{\Phi_{k_\Phi}} + \sum_{j=1}^{k_\Phi - 1} \Xi^{[2]}_{\Phi_j} \Delta_{\Phi_j} + \varepsilon^{[2]}_{\Phi} + \\ + \bar{B} \Delta u \Big] \Bigg\} - \sigma^\top C_2 \hat{\bar{B}} v_r.\end{aligned}$$

Then, substituting v_r as in (7.4), and by means of Assumptions 5.1, 5.2, 7.1, and 5.6, and Proposition 5.2, it holds that

$$\dot{v} \leq \|\sigma\| \left\{ \|C_1\| \left[\bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1} \right] + \|C_2\| \left[\bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + \bar{\gamma} \bar{\delta} \right] \right\} - \rho \underline{\lambda}(C_2) \underline{\gamma} \|\sigma\|.$$

Finally, if one selects the discontinuous control gain as

$$\rho > \frac{\|C_1\| \left\{ \bar{c}_{\Phi_1} + \bar{\varepsilon}_{\Phi_1} \right\} + \|C_2\| \left\{ \bar{c}_{\Phi_2} + \bar{\varepsilon}_{\Phi_2} + \bar{\delta} \right\} + \bar{\eta}}{\underline{\lambda}(C_2) \underline{\gamma}},$$

it holds that $\dot{v} \leq -\eta \|\sigma\|$, implying $\sigma(x(t)) \to 0_m$ for $t \to \infty$ and concluding the proof.

B.9 Proof of Theorem 7.3

The proof is similar to the one of Theorem 5.2 and it follows the one in [53, Theorem 3]. Nevertheless, the main steps are reported for completeness. Consider the Lyapunov

candidate function $v : \mathcal{X} \to \mathbb{R}$ defined as

$$v(x) = \frac{1}{2}\sigma^\top\sigma + \frac{1}{2}\tilde{\rho}^2, \tag{B.48}$$

where $\tilde{\rho} = \rho^\star - \hat{\rho}$ is the error between the discontinuous control gain and the unknown ideal gain, capable of of enforcing a sliding mode $\sigma(x(t)) = 0_m$ for $t \leq t_1$. In particular, such a gain is defined as

$$\rho^\star = \frac{\|C_1\| \left(\sup_{x\in\mathcal{X}} \left\| f_1 - \hat{\Phi}^{[1]}_{k_\Phi,t_1} \right\|\right) + \|C_2\| \left(\sup_{x\in\mathcal{X}} \left\| f_2 - \hat{\Phi}^{[2]}_{k_\Phi,t_1} \right\| + \bar{\gamma}\bar{\delta}\right)}{\underline{\lambda}(C_2)\underline{\gamma}} + \frac{\eta^\star}{\underline{\lambda}(C_2)\underline{\gamma}}, \tag{B.49}$$

where $\eta^\star \in \mathbb{R}_{>0}$, while $\hat{\Phi}^{[1]}_{k_\Phi,t_1}$ and $\hat{\Phi}^{[2]}_{k_\Phi,t_1}$ represent the output of the DNN characterized by fixed weights $V_j(t_1)$, for $j \in \{0, 1, \ldots, k_\Phi\}$.

Recalling that the integral sliding variable σ is chosen as in (7.9), where σ_0 is the one in (7.5) and $\hat{z}$ is characterized by the dynamics in (7.10), keeping the weights of the DNN fixed to the value they had at time t_1, then the first time-derivative of the Lyapunov candidate function is computed as

$$\begin{aligned}
\dot{v}(x) &= \sigma^\top\dot{\sigma} - \tilde{\rho}\dot{\hat{\rho}},\\
&= \sigma^\top\left\{C_1\left[\tilde{\Phi}^{[1]}_{k_\Phi,t_1} + \varepsilon^{[1]}_\Phi\right] + C_2\left[\tilde{\Phi}^{[2]}_{k_\Phi,t_1} + \varepsilon^{[2]}_\Phi + \bar{B}\Delta u\right]\right\} - \hat{\rho}\sigma^\top C_2\hat{\bar{B}}\frac{\sigma}{\|\sigma\|} - \tilde{\rho}\dot{\hat{\rho}},\\
&= \sigma^\top\left\{C_1\left[\tilde{\Phi}^{[1]}_{k_\Phi,t_1} + \varepsilon^{[1]}_\Phi\right] + C_2\left[\tilde{\Phi}^{[2]}_{k_\Phi,t_1} + \varepsilon^{[2]}_\Phi + \bar{B}\Delta u\right]\right\} - \rho^\star\sigma^\top C_2\hat{\bar{B}}\frac{\sigma}{\|\sigma\|} +\\
&\quad - \sigma^\top C_2\hat{\bar{B}}\frac{\eta^\star}{\underline{\lambda}(C_2)\underline{\gamma}}\frac{\sigma}{\|\sigma\|} + \tilde{\rho}\sigma^\top C_2\hat{\bar{B}}\frac{\sigma}{\|\sigma\|} - \tilde{\rho}\dot{\hat{\rho}}
\end{aligned}$$

where $\tilde{\Phi}^{[1]}_{k_\Phi,t_1} = \Phi^{[1]}_{k_\Phi} - \hat{\Phi}^{[1]}_{k_\Phi,t_1}$ and $\tilde{\Phi}^{[2]}_{k_\Phi,t_1} = \Phi^{[2]}_{k_\Phi} - \hat{\Phi}^{[2]}_{k_\Phi,t_1}$.

Exploiting Proposition 5.3 to bound the approximation errors and bounding the other terms relying on Assumptions 5.2 and 5.3, it holds that

$$\begin{aligned}
\dot{v}(x) \leq{}& \|\sigma\|\,\|C_1\| \sup_{x\in\mathcal{X}} \left\| f_1 - \hat{\Phi}^{[1]}_{k_\Phi,t_1} \right\| + \|\sigma\|\,\|C_2\| \sup_{x\in\mathcal{X}} \left\| f_2 - \hat{\Phi}^{[2]}_{k_\Phi,t_1} \right\| + \|\sigma\|\,\|C_2\|\,\bar{\gamma}\bar{\delta} +\\
&- \|\sigma\|\,\underline{\lambda}(C_2)\underline{\gamma}\rho^\star - \|\sigma\|\,\eta^\star + \|\sigma\|\,\bar{\lambda}(C_2)\bar{\gamma}\,\|\tilde{\rho}\| - \tilde{\rho}\dot{\hat{\rho}}.
\end{aligned}$$

Substituting the values $\rho^\star$ a (B.49) the above inequality can be simplified as

$$\dot{v}(x) = -\|\sigma\|\,\eta^\star + \|\sigma\|\,\bar{\lambda}(C_2)\bar{\gamma}\,\|\tilde{\rho}\| - \tilde{\rho}\dot{\hat{\rho}}.$$

Substituting the adaptation law in (7.14) and exploiting property 3 of Lemma 5.2, one has that

$$\dot{v}(x) \leq -\|\sigma\|\,\eta^\star + \|\sigma\|\,\bar{\lambda}(C_2)\bar{\gamma}\,\|\tilde{\rho}\| - \tilde{\rho}\bar{\lambda}(C_2)\bar{\gamma}\,\|\sigma\| - \tilde{\rho}\alpha\,\|\sigma\|\,.$$

Moreover, letting $\rho^\star > \bar{\rho}$, since $\hat{\rho} \leq \bar{\rho}$ by virtue of the projection operator, then it holds $\|\tilde{\rho}\| \equiv \tilde{\rho}$ and the following holds

$$\dot{v}(x) \leq -\|\sigma\|\,\eta^\star - \tilde{\rho}\alpha\,\|\sigma\|$$

$$
\begin{aligned}
&\leq -\alpha \left\|\sigma\right\| \left\|\tilde{\rho}\right\| - \eta^{\star} \left\|\sigma\right\| \\
&\leq -\eta_1 \left\|\tilde{\rho}\right\| - \eta^{\star} \left\|\sigma\right\| \\
&\leq -\sqrt{2} \min\{\eta^{\star}, \eta_1\} \left(\frac{\left\|\sigma\right\|}{\sqrt{2}} + \frac{\left\|\tilde{\rho}\right\|}{\sqrt{2}} \right) \\
&\leq -\sqrt{2} \min\{\eta^{\star}, \eta_1\} \sqrt{v(x)}
\end{aligned}
$$

proving that there exist $t_2 \geq t_1$ such that $\sigma(x(t)) = 0_m$ for $t \geq t_2$ [53].

Moreover, imposing $\dot{\sigma} = 0_m$ and solving for v_r allows to compute the equivalent control v_{eq}. In particular

$$
\dot{\sigma} = C_1 \left[\tilde{\Phi}^{[1]}_{k_\Phi, t_1} + \varepsilon^{[1]}_{\Phi} \right] + C_2 \left[\tilde{\Phi}^{[2]}_{k_\Phi, t_1} + \varepsilon^{[2]}_{\Phi} \right] + C_2 \bar{B} \Delta u + C_2 \hat{\bar{B}} v_r = 0_m
$$

leads to

$$
v_{\text{eq}} = (C_2 \hat{\bar{B}})^{-1} \Big\{ C_1 \left[\tilde{\Phi}^{[1]}_{k_\Phi, t_1} + \varepsilon^{[1]}_{\Phi} \right] + C_2 \left[\tilde{\Phi}^{[2]}_{k_\Phi, t_1} + \varepsilon^{[2]}_{\Phi} \right] \Big\} + (C_2 \hat{\bar{B}})^{-1} C_2 \bar{B} \Delta u.
$$

The proof is concluded.

B.10 Proof of Theorem 8.1

The proof follows directly from [44, Theorem 2] and [155, Theorem 2]. Let $\sigma_{1,i}(t_0)$ and $\sigma_{2,i}(t_0)$, with $i \in \{1, 2, \ldots, n\}$, be the initial condition of the sliding variable components. Then, for sake or simplicity, let such conditions be defined so that $\sigma_{1,i}(0) > -\frac{\sigma_{2,i}(t_0)|\sigma_{2,i}(t_0)|}{2\alpha_{r,i}}$ and $\sigma_{2,i}(t_0) > 0$ (all the other symmetric conditions are analogous). Since $\sigma_{1,i} = -\frac{\sigma_{2,i}|\sigma_{2,i}|}{2\alpha_{r,i}}$ corresponds to the minimum time curve in the nominal case, one has to prove that, in the case of the worst realization of the uncertain terms, the auxiliary state trajectory under (8.25) follows this curve, in an equivalent sense. In fact, computing the vector field, one has $[\sigma_{2,i}, -F_i + \alpha_i] = [\sigma_{2,i}, -\alpha_{r,i}]$, that is the trajectory moves towards the curve, while pointing downward. When the curve $\sigma_{1,i} = -\frac{\sigma_{2,i}|\sigma_{2,i}|}{2\alpha_{r,i}}$ is reached, the control sign changes and the vector field becomes $[\sigma_{2,i}, -F_i + \alpha_i] = [\sigma_{2,i}, \alpha_{\text{r},i}]$, with $\sigma_{2,i} < 0$. This means that the trajectory is always tangent to the curve, with the states of (8.24) moving towards the origin in minimum time.

B.11 Proof of Theorem 9.1

The above theorem can be proven by performing Lyapunov analysis on the candidate function

$$
v = \frac{1}{2} \sigma^{\top} \sigma + \frac{1}{2} \sum_{j=0}^{k_\Phi} \text{vec}\left(\tilde{V}_j\right)^{\top} \Gamma_{\Phi_j}^{-1} \text{vec}\left(\tilde{V}_j\right) + \frac{\tilde{\rho}^2}{2\mu}, \tag{B.50}
$$

where $\tilde{\rho} = \rho^{\star} - \hat{\rho} \in \mathbb{R}_{>0}$, characterized by first time-derivative of (B.50) is given by

$$
\dot{v} = \sigma^{\top} \dot{\sigma} - \sum_{j=0}^{k_\Phi} \text{vec}\left(\tilde{V}_j\right)^{\top} \Gamma_{\Phi_j}^{-1} \text{vec}\left(\dot{\hat{V}}_j\right) - \frac{\tilde{\rho} \dot{\hat{\rho}}}{\mu}. \tag{B.51}
$$

Then, it is possible to compute the time-derivative of the integral sliding variable in 9.16, i.e.,

$$\begin{aligned}
\dot{\sigma} &= \dot{\sigma}_0 - \dot{\hat{z}} \\
&= C_1\Big\{\dot{q}^\star - \ddot{q}^\star\Big\} + C_2\Big\{M^{-1}(\tau_h - \nu) + M^{-1}\tau_n + M^{-1}\tau_r - \ddot{q}^\star\Big\} - C_1\Big\{\dot{q}^\star - \ddot{q}^\star\Big\} + \\
&\quad - C_2\Big\{M^{-1}\hat{\Phi}_{k_\Phi} + M^{-1}\tau_n + M^{-1}\tau_r - \ddot{q}^\star\Big\} \\
&= C_2M^{-1}\Big\{\tau_h - \nu - \hat{\Phi}_{k_\Phi}\Big\} + C_2M^{-1}\tau_r \\
&= C_2M^{-1}\Big\{\Phi_{k_\Phi} + \varepsilon_\Phi - \hat{\Phi}_{k_\Phi}\Big\} - C_2M^{-1}\hat{\rho}\frac{\sigma}{\|\sigma\|} \\
&= C_2M^{-1}\Big\{\tilde{\Phi}_{k_\Phi} + \varepsilon_\Phi\Big\} - C_2M^{-1}(\rho^\star - \tilde{\rho})\frac{\sigma}{\|\sigma\|}
\end{aligned}$$

If one exploits the form (5.29) to rewrite $\tilde{\Phi}_{k_\Phi}$, it is possible to rewrite $\dot{\sigma}$ as

$$\dot{\sigma} = C_2M^{-1}\left\{\sum_{j=0}^{k_\Phi}\Lambda_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) + \sum_{j=1}^{k_\Phi}\Xi_{\Phi_j}\Delta_{\Phi_j} + \varepsilon_\Phi\right\} - C_2M^{-1}(\rho^\star - \tilde{\rho})\frac{\sigma}{\|\sigma\|}.$$

Hence, substituting it into (B.51) yields

$$\begin{aligned}
\dot{v} = \sigma^\top C_2M^{-1}&\left\{\sum_{j=1}^{k_\Phi}\Xi_{\Phi_j}\Delta_{\Phi_j} + \varepsilon_\Phi\right\} - \sigma^\top C_2M^{-1}\rho^\star\frac{\sigma}{\|\sigma\|} + \\
&+ \sigma^\top C_2M^{-1}\sum_{j=0}^{k_\Phi}\Lambda_{\Phi_j}\mathrm{vec}\left(\tilde{V}_j\right) - \sum_{j=0}^{k_\Phi}\mathrm{vec}\left(\tilde{V}_j\right)^\top\Gamma_{\Phi_j}^{-1}\mathrm{vec}\left(\dot{\tilde{V}}_j\right) + \\
&+ \sigma^\top C_2M^{-1}\tilde{\rho}\frac{\sigma}{\|\sigma\|} - \frac{\tilde{\rho}\dot{\tilde{\rho}}}{\mu}
\end{aligned}$$

Then, applying the DNN adaptation law in (9.19) and exploiting point 3 of Lemma 5.2, the above equation can be simplified as

$$\dot{v} \leq \sigma^\top C_2M^{-1}\left\{\sum_{j=1}^{k_\Phi}\Xi_{\Phi_j}\Delta_{\Phi_j} + \varepsilon_\Phi\right\} - \sigma^\top C_2M^{-1}\rho^\star\frac{\sigma}{\|\sigma\|} + \sigma^\top C_2M^{-1}\tilde{\rho}\frac{\sigma}{\|\sigma\|} - \frac{\tilde{\rho}\dot{\tilde{\rho}}}{\mu}.$$

Using the constants in Assumption 5.6 and Proposition 5.2 the above inequality can be bounded above as

$$\dot{v} \leq \|\sigma\|\left\|C_2M^{-1}\right\|\{\bar{c}_\Phi + \bar{\varepsilon}_\Phi\} - \underline{\lambda}(C_2M^{-1})\rho^\star\|\sigma\| + \tilde{\rho}\left\|C_2M^{-1}\right\|\|\sigma\| - \frac{\tilde{\rho}\dot{\tilde{\rho}}}{\mu}.$$

Then, since the ideal gain $\rho^\star$ is defined as in (9.20), the first term is dominated and it holds that

$$\dot{v} \leq -\eta^\star\|\sigma\| + \tilde{\rho}\left\|C_2M^{-1}\right\|\|\sigma\| - \frac{\tilde{\rho}\dot{\tilde{\rho}}}{\mu}.$$

Substituting (9.22) in the above inequality yields

$$\dot{v} \leq -\eta^\star\|\sigma\| + \tilde{\rho}\left\|C_2M^{-1}\right\|\|\sigma\| - \tilde{\rho}\|\sigma\|\left\|C_2M^{-1}\right\|\mathrm{sign}\left(\|\sigma\| - \varepsilon_\sigma\right).$$

Two cases can be distinguished. First, if $\|\sigma\| > \varepsilon_\sigma$, it holds that $\mathrm{sign}\left(\|\sigma\| - \varepsilon_\sigma\right) = 1$, and, as a consequence

$$\dot{v} \leq -\eta^\star\|\sigma\| + \tilde{\rho}\left\|C_2M^{-1}\right\|\|\sigma\| - \tilde{\rho}\|\sigma\|\left\|C_2M^{-1}\right\|$$

$$\leq -\eta^{\star} \|\sigma\| .$$

While, if $\|\sigma\| < \varepsilon_{\sigma}$, then $\operatorname{sign}(\|\sigma\| - \varepsilon_{\sigma}) = -1$ and it is true that

$$\dot{v} \leq -\eta^{\star} \|\sigma\| + 2\tilde{\rho} \|\sigma\| \left\| C_2 M^{-1} \right\| ,$$

meaning that nothing can be said about the behavior of the integral sliding variable when σ when $\|\sigma\| < \varepsilon_{\sigma}$. The two conditions lead to the conclusion that σ is ultimately bounded into the set $\Omega_{\sigma} \coloneqq \{\sigma \in \mathbb{R}^n : \|\sigma\| \leq \varepsilon_{\sigma}\}$, implying the enforcement of a practical sliding mode.

Appendix C

Franka Emika Panda Robot

The Franka Emika Panda robot is a collaborative robot characterized by 7 DoF, present in the Intelligent Robotics Lab of the University of Pavia, and depicted in Figure C.1. In this appendix, a description about the technical specifications, kinematics and dynamics are provided. Moreover, it is described how the robot is simulated and controlled.

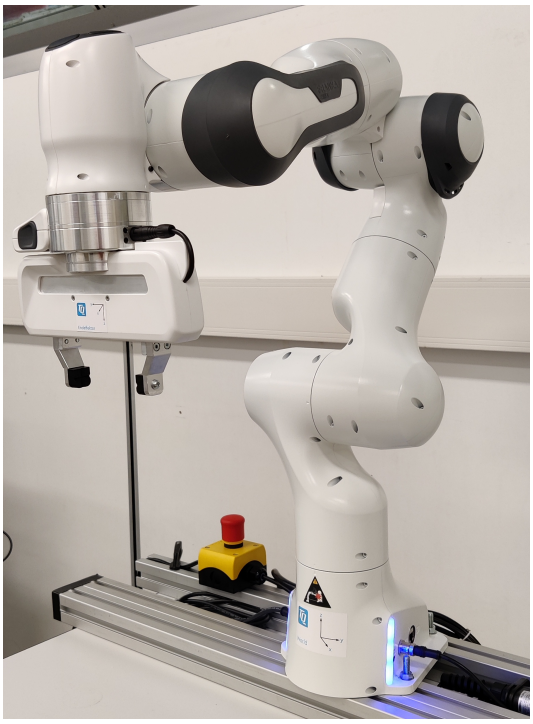


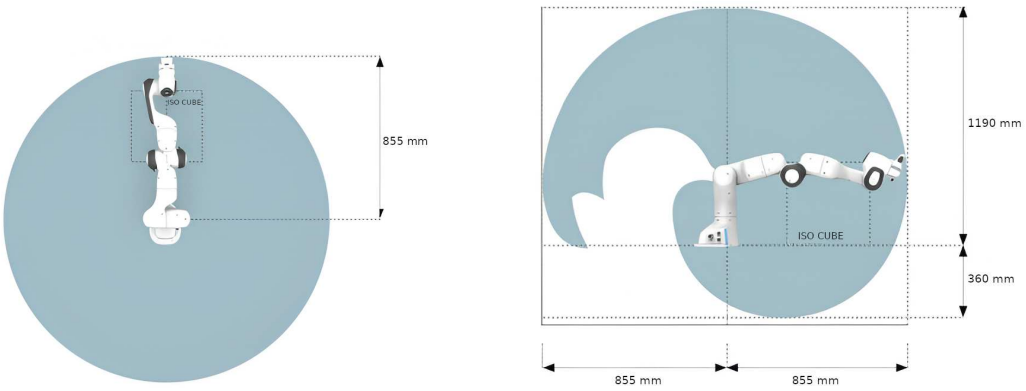
Figure C.1: Franka Emika Panda robot present in the University of Pavia.

C.1 Technical Specifications

The robot is characterized by 7 DoF (one for each joint), all equipped with torque sensors, which allow the manipulator to be controlled by setting torque references. The limits of the joints, for what concerns position, velocity, acceleration, and torque, are presented in Table C.1, while top and side views of the workspace of the manipulator are depicted in Figure C.2.

Name	$\bar{q}$ [rad]	$\underline{q}$ [rad]	$\bar{\dot{q}}$ [rad/s]	$\bar{\ddot{q}}$ [rad/s^2]	$\bar{\tau}$ [Nm]
Joint 1	2.8973	-2.8973	2.175	15	87
Joint 2	1.7628	-1.7628	2.175	7.5	87
Joint 3	2.8973	-2.8973	2.175	10	87
Joint 4	-0.0698	-3.0718	2.175	12.5	87
Joint 5	2.8973	-2.8973	2.61	15	12
Joint 6	3.7525	-0.0175	2.61	20	12
Joint 7	2.8973	-2.8973	2.61	20	12

Table C.1: Limits of the robot joints.



(a) Top view of the robot workspace.

(b) Side view of the robot workspace.

Figure C.2: Workspace of the Franka Emika Panda robot.

In order to compute the kinematics of the manipulator, instrumental for the development of certain control strategies, the Panda robot makes use of modified (proximate) DH parameters [82], presented in Figure C.3 and Table C.2.

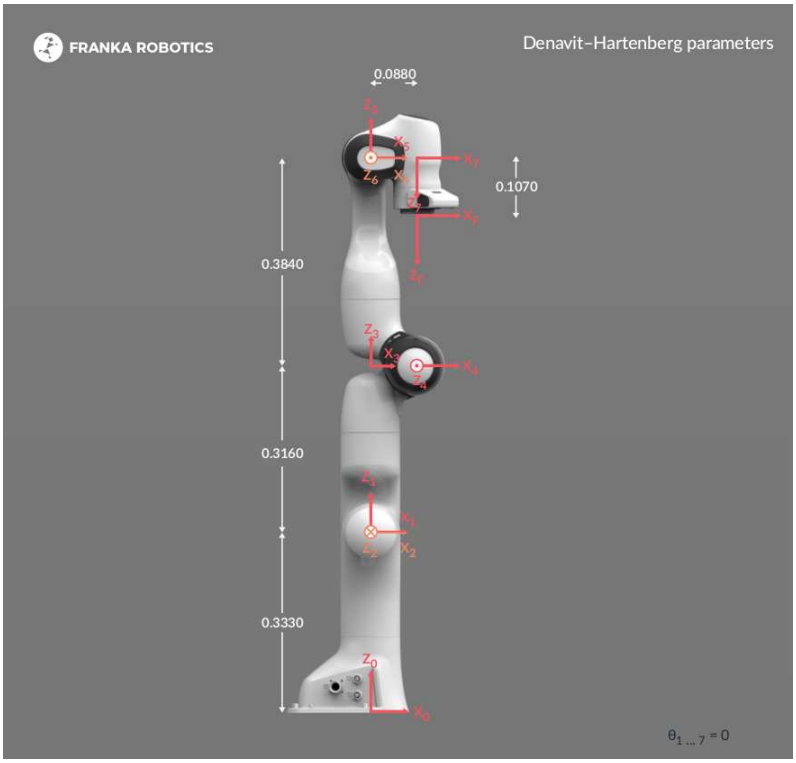


Figure C.3: Graphical representation of the modified DH parameters of the Franka Emika Panda robot (picture taken from the documentation).

C.2 Dynamical Modeling

Since it is an open-chain manipulator, the dynamics of the Franka Emika Panda robot can be described by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s\text{sign}\,(\dot{q}) + g(q) = \tau$$

where $q \in \mathbb{R}^7$ is the vector of the joint positions, $M : \mathbb{R}^7 \to \mathbb{R}^{7\times7}$ is the inertia matrix, $C : \mathbb{R}^7 \times \mathbb{R}^7 \to \mathbb{R}^{7\times7}$ is the Coriolis matrix, $F_v \in \mathbb{R}^{7\times7}$ is the viscous friction coefficient matrix, $F_v \in \mathbb{R}^{7\times7}$ is the static friction coefficient matrix, $g : \mathbb{R}^7 \to \mathbb{R}^7$ is the gravitational torque, while $\tau \in \mathbb{R}^7$ is the input torque.

As detailed in the Panda robot documentation [156], the effects caused by gravity and frictions are internally compensated, meaning that it is possible to consider $g(q) = 0_7$, $F_v\dot{q} = 0_7$, and $F_s\text{sign}\,(\dot{q}) = 0_7$, for all $q \in \mathbb{R}^7$ and $\dot{q} \in \mathbb{R}^7$. As a consequence, it is possible to write a the dynamics of the robot as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau \tag{C.1}$$

The values of the inertia and Coriolis matrices is strictly dependent on the dynamics parameters of the robot. Some basic information about this last one is provided by Franka Emika in the documentation [156]. For a complete identification of the robot dynamics, the reader is invited to refer to the work of Gaz et al. [157].

Name	a [m]	d [m]	α [rad]	θ [rad]
Joint 1	0	0.33	0	q_1
Joint 2	0	0	$-\frac{\pi}{2}$	q_2
Joint 3	0	0.316	$\frac{\pi}{2}$	q_3
Joint 4	0.0825	0	$\frac{\pi}{2}$	q_4
Joint 5	-0.0825	0.384	$-\frac{\pi}{2}$	q_5
Joint 6	0	0	$\frac{\pi}{2}$	q_6
Joint 7	0.08	0	$\frac{\pi}{2}$	q_7
Flange	0	0.107	0	0

Table C.2: Modified DH parameters of the Franka Emika Panda robot.

Note that, even though the masses and inertia of the robot components are not directly available, Franka Emika made the matrices M and C retrievable directly from the robot via the proprietary programming library (more details will be provided in Section C.4).

C.3 PyBullet Simulation

In order to accurately simulate the behavior of the Franka Emika Panda robot, it has been chosen to rely on PyBullet [158], a Python module for robotics simulation and machine learning, with a focus on sim-to-real transfer. Such a module can be easily integrated with any control strategy thanks to the easy-to-use API and provides robotic functionalities such as forward dynamic simulation, inverse dynamic computation, forward and inverse dynamics.

In a PyBullet simulation it is possible to load any robot described in the Unified Robot Description Format (URDF) [159], which contains information about its kinematic structure and dynamics. Then, the forward dynamics of the robot is simulated via the Articulated Body Algorithm (ABA) [86], while the inverse dynamics, useful for the computation of the mass matrix and Coriolis matrix, is performed using the Recursive Newton-Euler Algorithm (RNEA), described in [87].

In the work reported in this dissertation, the Panda robot has been simulated using the URDF description provided by Franka Emika [160]. A virtualized version of the Panda robot in PyBullet is presented in Figure C.4. To have a simulation that is coherent to the dynamics of the real robot (C.1), the effect of gravity is disabled in the simulation settings, and frictions coefficients are removed from the URDF configuration file of the robot.

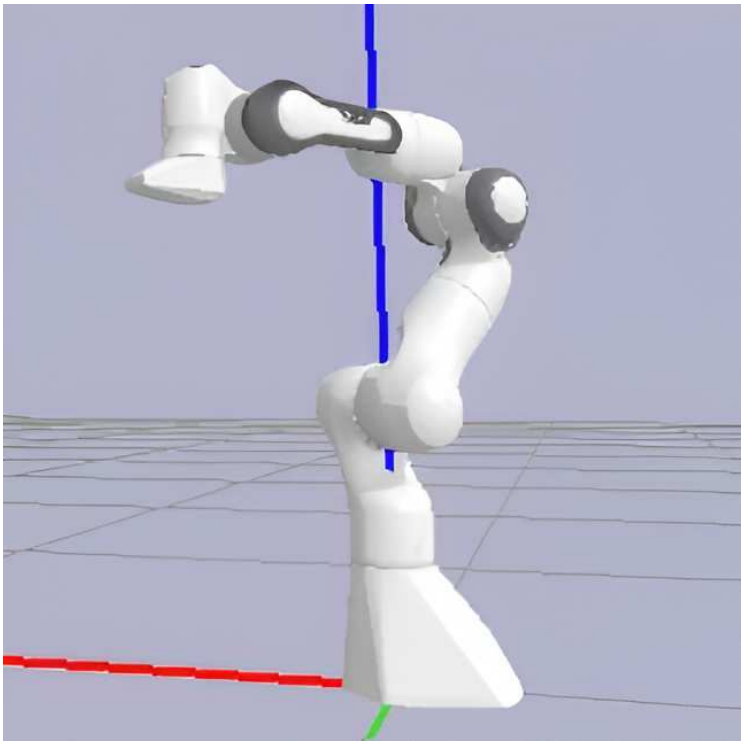


Figure C.4: Virtualization of the Franka Emika Panda robot in PyBullet.

C.4 Controlling the Panda robot

In order to exchange information with the Panda robot, it is required to have a computer on which is installed *Libfranka* [161], a C++ library provided by Franka Emika that allows to retrieve data and send the references for the control signals to the manipulator. Such a computer is connected to the low-level controller of the robot via Ethernet connection, as depicted in Figure C.5. Moreover, to ensure low latency communication, Ubuntu operative system with custom real-time kernel must be installed.

Even though Libfranka allows an easy interfacing with the robot, for the experiments reported in this thesis, the software for controlling the robot and gathering data from sensors, has been integrated in the Robot Operating System (ROS) framework. This integration allows to have an easier communication infrastructure for the software components and gives the possibility to exploit the benefits of different programming languages.

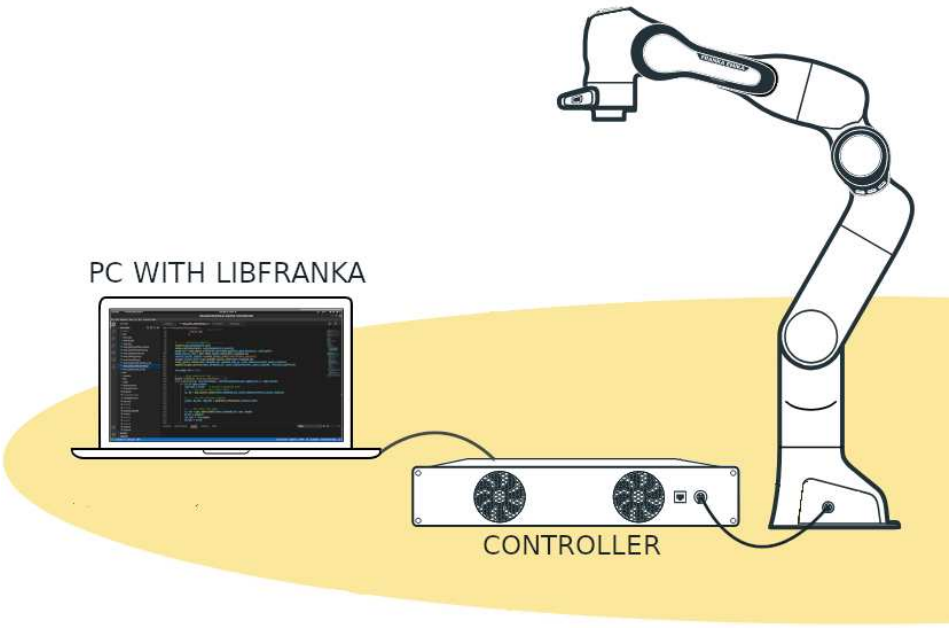


Figure C.5: Connection diagram of the Franka Emika Panda robot, the low-level controller, and the computer with Libfranka library.

Bibliography

[1] K. Zhou and J. C. Doyle, *Essentials of robust control*, vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

[2] A. Weinmann, *Uncertain models and robust control*. Springer Science & Business Media, 2012.

[3] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization," *Psychological Review*, vol. 65, no. 8, p. 386–408, 1958.

[4] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[5] J. Hastad, "Almost optimal lower bounds for small depth circuits," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 6–20, 1986.

[6] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] S. N. Kumpati, P. Kannan, *et al.*, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990.

[8] E. Terzi, F. Bonassi, M. Farina, and R. Scattolini, "Learning model predictive control with long short-term memory networks," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8877–8896, 2021.

[9] D. Psaltis, A. Sideris, and A. A. Yamamura, "A multilayered neural network controller," *IEEE control systems magazine*, vol. 8, no. 2, pp. 17–21, 1988.

[10] V. I. Utkin, *Sliding Modes in Control and Optimization*. Springer Berlin, Heidelberg, 1992.

[11] M. Zhihong, X. Yu, K. Eshraghian, and M. Palaniswami, "A robust adaptive sliding mode tracking control using an rbf neural network for robotic manipulators," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 5, pp. 2403–2408, IEEE, 1995.

[12] J. Fei and H. Ding, "Adaptive sliding mode control of dynamic system using rbf neural network," *Nonlinear Dynamics*, vol. 70, pp. 1563–1573, 2012.

[13] X. Chen, W. Shen, M. Dai, Z. Cao, J. Jin, and A. Kapoor, "Robust adaptive sliding-mode observer using rbf neural network for lithium-ion battery state of charge estimation in electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 1936–1947, 2015.

[14] F.-J. Lin and P.-H. Shen, "Robust fuzzy neural network sliding-mode control for two-axis motion control system," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 4, pp. 1209–1225, 2006.

[15] N. Al-Holou, T. Lahdhiri, D. S. Joo, J. Weaver, and F. Al-Abbas, "Sliding mode neural network inference fuzzy logic control for active suspension systems," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 234–246, 2002.

[16] R.-J. Wai and F.-J. Lin, "Fuzzy neural network sliding-mode position controller for induction servo motor drive," *IEE Proceedings-Electric Power Applications*, vol. 146, no. 3, pp. 297–308, 1999.

[17] H. Morioka, K. Wada, A. Sabanovic, and K. Jezernik, "Neural network based chattering free sliding mode control," in *SICE'95. Proceedings of the 34th SICE Annual Conference. International Session Papers*, pp. 1303–1308, IEEE, 1995.

[18] M. A. Hussain and P. Y. Ho, "Adaptive sliding mode control with neural network based hybrid models," *Journal of Process Control*, vol. 14, no. 2, pp. 157–176, 2004.

[19] X. Lu, X. Zhang, G. Zhang, J. Fan, and S. Jia, "Neural network adaptive sliding mode control for omnidirectional vehicle with uncertainties," *ISA transactions*, vol. 86, pp. 201–214, 2019.

[20] Z. Han, S. Li, and H. Liu, "Composite learning sliding mode synchronization of chaotic fractional-order neural networks," *Journal of Advanced Research*, vol. 25, pp. 87–96, 2020.

[21] S. Emelyanov, *Variable structure control systems.* Nauka, Moscow, 1967.

[22] A. Isidori, *Nonlinear control systems: an introduction.* Springer, 1985.

[23] P. Wach, *Dynamics and Control of Electrical Drives.* Springer-Verlag Berlin Heidelberg, 2011.

[24] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control.* Springer London, 2008.

[25] H. K. Khalil, *Nonlinear Systems.* Prentice Hall, 2002.

[26] A. Ferrara, G. P. Incremona, and M. Cucuzzella, *Advanced and Optimization Based Sliding Mode Control: Theory and Applications.* Philadelphia, PA: Society for Industrial and Applied Mathematics, 2019.

[27] C. Edwards and S. Spurgeon, *Sliding Mode Control: Theory And Applications*. CRC Press, 1998.

[28] V. Utkin, J. Guldner, and X. Shi, *Sliding Modes in Elctromechanical Systems*. Taylor & Francis, 1999.

[29] A. F. Filippov, "Differential equations with discontinuous righthand sides," in *Mathematics and Its Applications*, 1988.

[30] B. Drazenovic, "The invariance conditions in variable structure systems," *Automatica*, vol. 5, no. 3, pp. 287–295, 1969.

[31] R. A. DeCarlo, S. H. Zak, and G. P. Matthews, "Variable structure control of nonlinear multivariable systems: a tutorial," *Proceedings of the IEEE*, vol. 76, no. 3, pp. 212–232, 1988.

[32] J. L. Chang, S. L. Lin, K. C. Chu, and M. S. Chen, "Lyapunov stability analysis of second-order sliding-mode control and its application to chattering reduction design," *International Journal of Control, Automation and Systems*, vol. 14, no. 3, pp. 691–697, 2016.

[33] G. Bartolini, A. Ferrara, and E. Usai, "Chattering avoidance by second-order sliding mode control," *IEEE Transactions on Automatic Control*, vol. 43, no. 2, pp. 241–246, 1998.

[34] G. Bartolini and T. Zolezzi, "Behavior of variable-structure control systems near the sliding manifold," *Systems & control letters*, vol. 21, no. 1, pp. 43–48, 1993.

[35] G. Bartolini, E. Punta, and T. Zolezzi, "Approximability properties for second-order sliding mode control systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 10, pp. 1813–1825, 2007.

[36] V. Utkin and J. Shi, "Integral sliding mode in systems operating under uncertainty conditions," in *35th IEEE Conference on Decision and Control*, vol. 4, (Kobe, Japan), pp. 4591–4596, Dec. 1996.

[37] F. Castanos and L. Fridman, "Analysis and design of integral sliding manifolds for systems with unmatched perturbations," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 853–858, 2006.

[38] M. Rubagotti, A. Estrada, F. Castaños, A. Ferrara, and L. Fridman, "Integral sliding mode control for nonlinear systems with matched and unmatched perturbations," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2699–2704, 2011.

[39] J.-J. E. Slotine, W. Li, *et al.*, *Applied nonlinear control*, vol. 199. Prentice hall Englewood Cliffs, NJ, 1991.

[40] A. Levant, "Higher-order sliding modes, differentiation and output-feedback control," *International journal of Control*, vol. 76, no. 9-10, pp. 924–941, 2003.

[41] G. Bartolini, A. Ferrara, A. Levant, and E. Usai, "On second order sliding mode controllers," *Variable structure systems, sliding mode and nonlinear control*, pp. 329–350, 1999.

[42] G. Bartolini, A. Ferrara, E. Usai, and V. I. Utkin, "On multi-input chattering-free second-order sliding mode control," *IEEE transactions on automatic control*, vol. 45, no. 9, pp. 1711–1717, 2000.

[43] T. Floquet, J.-P. Barbot, and W. Perruquetti, "Higher-order sliding mode stabilization for a class of nonholonomic perturbed systems," *Automatica*, vol. 39, no. 6, pp. 1077–1083, 2003.

[44] F. Dinuzzo and A. Ferrara, "Higher order sliding mode controllers with optimal reaching," *IEEE Transactions on Automatic Control*, vol. 54, no. 9, pp. 2126–2136, 2009.

[45] A. Levant, "Quasi-continuous high-order sliding-mode controllers," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 5, pp. 4605–4610, IEEE, 2003.

[46] J. P. Aubin and A. Cellina, *Differential Inclusions: Set-Valued Maps and Viability Theory*. Berlin, Heidelberg: Springer-Verlag, 1984.

[47] A. Ferrara and M. Rubagotti, "A sub-optimal second order sliding mode controller for systems with saturating actuators," *IEEE Transactions on Automatic Control*, vol. 54, no. 5, pp. 1082–1087, 2009.

[48] H. Kaufman, I. Barkana, and K. Sobel, *Direct adaptive control algorithms: theory and applications*. Springer Science & Business Media, 2012.

[49] G. Bartolini, A. Ferrara, and V. I. Utkin, "Adaptive sliding mode control in discrete-time systems," *Automatica*, vol. 31, no. 5, pp. 769–773, 1995.

[50] C. Edwards and Y. B. Shtessel, "Adaptive continuous higher order sliding mode control," *Automatica*, vol. 65, pp. 183–190, 2016.

[51] S. Roy, S. Baldi, and L. M. Fridman, "On adaptive sliding mode control without a priori bounded uncertainty," *Automatica*, vol. 111, p. 108650, 2020.

[52] Y.-J. Huang, T.-C. Kuo, and S.-H. Chang, "Adaptive sliding-mode control for nonlinearsystems with uncertain parameters," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 534–539, 2008.

[53] F. Plestan, Y. Shtessel, V. Bregeault, and A. Poznyak, "New methodologies for adaptive sliding mode control," *International Journal of Control*, vol. 83, no. 9, pp. 1907–1919, 2010.

[54] T. R. Oliveira, J. P. V. Cunha, and L. Hsu, "Adaptive sliding mode control for disturbances with unknown bounds," in *2016 14th International Workshop on Variable Structure Systems (VSS)*, pp. 59–64, IEEE, 2016.

[55] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[56] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, pp. 143–195, 1999.

[57] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Conference on learning theory*, pp. 2306–2327, PMLR, 2020.

[58] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[59] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[60] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[62] H. Bozdogan, "Model selection and akaike's information criterion (aic): The general theory and its analytical extensions," *Psychometrika*, vol. 52, pp. 345–370, 1987.

[63] P. Grünwald, "Model selection based on minimum description length," *Journal of mathematical psychology*, vol. 44, no. 1, pp. 133–152, 2000.

[64] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia: case studies on organization and retrieval*, pp. 21–49, Springer, 2008.

[65] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[66] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[67] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[68] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[69] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proceedings of the 25th international conference on Machine learning*, pp. 664–671, 2008.

[70] V. Gullapalli, "A stochastic reinforcement learning algorithm for learning real-valued functions," *Neural networks*, vol. 3, no. 6, pp. 671–692, 1990.

[71] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[72] D. Bertsekas, *Dynamic programming and optimal control: Volume I*, vol. 4. Athena scientific, 2012.

[73] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[74] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[75] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, pp. 9–44, 1988.

[76] J. A. Boyan, "Technical update: Least-squares temporal difference learning," *Machine learning*, vol. 49, pp. 233–246, 2002.

[77] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[78] L.-J. Lin, *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.

[79] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[80] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*, vol. 200. Springer, 2008.

[81] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[82] J. Craig, *Introduction To Robotics: Mechanics And Control, 3/E*. Pearson Education, 2009.

[83] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 16, pp. 1–19, 2004.

[84] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 14–20, 1985.

[85] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1, pp. 298–303, IEEE, 2001.

[86] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[87] J. Y. Luh, M. W. Walker, and R. P. Paul, "On-line computational scheme for mechanical manipulators," 1980.

[88] R. Freeman and P. V. Kokotovic, *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science & Business Media, 2008.

[89] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.

[90] A. Astolfi and R. Ortega, "Immersion and invariance: A new tool for stabilization and adaptive control of nonlinear systems," *IEEE Transactions on Automatic control*, vol. 48, no. 4, pp. 590–606, 2003.

[91] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems—a survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[92] M. T. Hagan, H. B. Demuth, and O. D. Jesús, "An introduction to the use of neural networks in control systems," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 12, no. 11, pp. 959–985, 2002.

[93] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 388–399, 1996.

[94] L. Chen and K. S. Narendra, "Nonlinear adaptive control using neural networks and multiple models," *Automatica*, vol. 37, no. 8, pp. 1245–1255, 2001.

[95] B. Yao and M. Tomizuka, "Adaptive robust control of siso nonlinear systems in a semi-strict feedback form," *Automatica*, vol. 33, no. 5, pp. 893–900, 1997.

[96] O. S. Patil, D. M. Le, M. L. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Systems Letters*, vol. 6, pp. 1855–1860, 2021.

[97] O. S. Patil, D. M. Le, E. J. Griffis, and W. E. Dixon, "Deep residual neural network (resnet)-based adaptive control: A lyapunov-based approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 3487–3492, IEEE, 2022.

[98] D. S. Bernstein, "Matrix mathematics," in *Matrix Mathematics*, Princeton university press, 2009.

[99] N. Sacchi, G. P. Incremona, and A. Ferrara, "Neural network-based practical/ideal integral sliding mode control," *IEEE Control Systems Letters*, vol. 6, pp. 3140–3145, 2022.

[100] E. Vacchini, N. Sacchi, G. P. Incremona, and A. Ferrara, "Design of a deep neural network-based integral sliding mode control for nonlinear systems under fully unknown dynamics," *IEEE Control Systems Letters*, vol. 7, pp. 1789–1794, 2023.

[101] N. Sacchi, E. Vacchini, G. P. Incremona, and A. Ferrara, "On neural networks application in integral sliding mode control," *Journal of the Franklin Institute*, vol. 361, no. 13, p. 106989, 2024.

[102] V. Strassen *et al.*, "Gaussian elimination is not optimal," *Numerische mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[103] C++ Reference, "std::chrono::high_resolution_clock." `https://en.cppreference.com/w/cpp/chrono/high_resolution_clock`.

[104] N. Sacchi, G. P. Incremona, and A. Ferrara, "Integral sliding modes generation via drl-assisted lyapunov-based control for robot manipulators," in *2023 European Control Conference (ECC)*, pp. 1–6, IEEE, 2023.

[105] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*, pp. 321–384. Springer International Publishing, 2021.

[106] N. Sacchi, E. Vacchini, and A. Ferrara, "Neural network based integral sliding mode control of systems with time-varying state constraints," in *2023 31st Mediterranean Conference on Control and Automation (MED)*, pp. 624–629, IEEE, 2023.

[107] M. Rubagotti, D. M. Raimondo, A. Ferrara, and L. Magni, "Robust model predictive control with integral sliding mode in continuous-time sampled-data nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 556–570, 2010.

[108] A. Ferrara, G. P. Incremona, and L. Magni, "Model-based event-triggered robust mpc/ism," in *2014 european control conference (ECC)*, pp. 2931–2936, IEEE, 2014.

[109] G. P. Incremona, A. Ferrara, and L. Magni, "Hierarchical model predictive/sliding mode control of nonlinear constrained uncertain systems," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 102–109, 2015.

[110] A. Ferrara, G. P. Incremona, and L. Magni, "A robust MPC/ISM hierarchical multi-loop control scheme for robot manipulators," in *52nd IEEE Conference on decision and control*, pp. 3560–3565, IEEE, 2013.

[111] G. P. Incremona, A. Ferrara, and L. Magni, "Asynchronous networked mpc with ism for uncertain nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4305–4317, 2017.

[112] N. Sacchi, E. Vacchini, G. P. Incremona, and A. Ferrara, "Model predictive control with deep neural network based integral sliding modes generation for a class of uncertain nonlinear systems," *IFAC-PapersOnLine*, vol. 58, no. 5, pp. 84–89, 2024.

[113] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control.*, vol. 62, no. 8, pp. 3861–3876, 2016.

[114] K. P. Tee, S. S. Ge, and E. H. Tay, "Barrier lyapunov functions for the control of output-constrained nonlinear systems," *Automatica*, vol. 45, no. 4, pp. 918–927, 2009.

[115] K. P. Tee and S. S. Ge, "Control of nonlinear systems with full state constraint using a barrier lyapunov function," in *Proc. of the 48h IEEE Conf. Decis. Control (CDC)*, pp. 8618–8623, 2009.

[116] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, 2019.

[117] K. H. Johansson, "The quadruple-tank process: A multivariable laboratory process with an adjustable zero," *IEEE Trans. Control Syst.*, vol. 8, no. 3, pp. 456–465, 2000.

[118] R. Isermann, *Fault-Diagnosis Systems*, vol. 1. Berlin: Springer, 2006.

[119] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault Detection and Diagnosis in Industrial Systems*. Verlag, London: Springer, 2000.

[120] R. Isermann, *Fault-Diagnosis Applications: Model-Based Condition Monitoring: Actuators, Drives, Machinery, Plants, Sensors, and Fault-Tolerant Systems*. Verlag, Berlin, Heidelberg: Springer, 2011.

[121] I. Andjelkovic, K. Sweetingham, and S. L. Campbell, "Active fault detection in nonlinear systems using auxiliary signals," (Seattle, Washington, USA), pp. 2142–2147, American Control Conference, Jun. 2008.

[122] M. Šimandl and I. Punčochář, "Active fault detection and control: Unified formulation and optimal design," *Automatica*, vol. 45, no. 9, pp. 2052–2059, 2009.

[123] J. K. Scott, R. Findeisen, R. D. Braatz, and D. M. Raimondo, "Input design for guaranteed fault diagnosis using zonotopes," *Automatica*, vol. 50, no. 6, pp. 1580–1589, 2014.

[124] D. Henry, J. Cieslak, A. Zolghadri, and D. Efimov, "A non-conservative $\mathcal{H}_/\mathcal{H}_\infty$ solution for early and robust fault diagnosis in aircraft control surface servo-loops," *Control Engineering Practice*, vol. 31, pp. 183–199, 2014.

[125] A. Wolfram, D. Fussel, T. Brune, and R. Isermann, "Component-based multi-model approach for fault detection and diagnosis of a centrifugal pump," (Arlington, VA, USA), pp. 4443–4448, American Control Conference, Jun. 2001.

[126] Y. Zhan and J. Jiang, "An interacting multiple-model based fault detection, diagnosis and fault-tolerant control approach," (Phoenix, AZ, USA), pp. 3593–3598, 38th IEEE Conference on Decision and Control, Dec. 1999.

[127] A. Jarrou, D. Sauter, and K. Alami, "Fault diagnosis and fault tolerant control based on model predictive control for nearly zero energy buildings," (Casablanca Morocco), pp. 219–225, 4th Conference on Control and Fault Tolerant Systems, Sep. 2019.

[128] J. Davila, L. Fridman, and A. Poznyak, "Observation and identification of mechanical systems via second order sliding modes," *International Journal of Control*, vol. 79, no. 10, pp. 1251–1262, 2006.

[129] S. K. Spurgeon, "Sliding mode observers: A survey," *International Journal of Systems Science*, vol. 39, no. 8, pp. 751–764, 2008.

[130] L. M. Capisani, A. Ferrara, and L. Magnani, "Design and experimental validation of a second-order sliding-mode motion controller for robot manipulators," *International Journal of Control*, vol. 82, no. 2, pp. 365–377, 2009.

[131] N. Sacchi, G. P. Incremona, and A. Ferrara, "Actuator fault diagnosis with neural network-integral sliding mode based unknown input observers," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 773–778, 2023.

[132] B. Halder and N. Sarkar, "Robust fault detection of a robotic manipulator," *The International Journal of Robotics Research*, vol. 26, no. 3, pp. 273–285, 2007.

[133] A. De Luca and R. Mattone, "An adapt-and-detect actuator FDI scheme for robot manipulators," (Barcelona, Spain), pp. 4975–4980, International Conference on Robotics and Automation, apr 2004.

[134] A. De Luca and R. Mattone, "An identification scheme for robot actuator faults," (Alberta, Canada), pp. 1127–1131, IEEE/RSJ International Conference on Intelligent Robots and Systems, Aug. 2005.

[135] L. M. Capisani, A. Ferrara, A. De Loza Ferreira, and L. Fridman, "Manipulator fault diagnosis via higher order sliding-mode observers," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 10, pp. 3979–3986, 2012.

[136] L. M. Capisani, A. Ferrara, and P. Pisu, "Sliding mode observers for vision-based fault detection, isolation and identification in robot manipulators," (Baltimore, Maryland, USA), pp. 4540–4545, American Control Conference, Jun. 2010.

[137] G. P. Incremona and A. Ferrara, "Fault diagnosis for robot manipulators via vision servoing based suboptimal second order sliding mode," (Naples, Italy), pp. 3090–3095, European Control Conference, Jul. 2019.

[138] N. Sacchi, G. P. Incremona, and A. Ferrara, "Sliding mode based fault diagnosis with deep reinforcement learning add-ons for intrinsically redundant manipulators," *International Journal of Robust and Nonlinear Control*, vol. 33, no. 15, pp. 9109–9127, 2023.

[139] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[140] L. Punnett and D. H. Wegman, "Work-related musculoskeletal disorders: the epidemiologic evidence and the debate," *Journal of electromyography and kinesiology*, vol. 14, no. 1, pp. 13–23, 2004.

[141] Å. Kilbom and J. Persson, "Work technique and its consequences for musculoskeletal disorders," *Ergonomics*, vol. 30, no. 2, pp. 273–279, 1987.

[142] R. Bridger, *Introduction to ergonomics*. Crc Press, 2008.

[143] A. Bestick, R. Pandya, R. Bajcsy, and A. D. Dragan, "Learning human ergonomic preferences for handovers," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3257–3264, 2018.

[144] L. Peternel, W. Kim, J. Babič, and A. Ajoudani, "Towards ergonomic control of human-robot co-manipulation and handover," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 55–60, 2017.

[145] N. Sacchi, E. Vacchini, and A. Ferrara, "Human-robot ergonomic handover via deep neural network based adaptive integral sliding mode control," in *2024 European Control Conference (ECC)*, pp. 585–590, IEEE, 2024.

[146] T.-J. Song, J. Jeong, and J.-H. Kim, "End-to-end real-time obstacle detection network for safe self-driving via multi-task learning," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2022.

[147] J. C. Martínez-Franco, A. Rojas-Álvarez, A. Tabares, D. Álvarez-Martínez, and C. A. Marín-Moreno, "Latent space representations for marker-less realtime hand–eye calibration," *Sensors*, vol. 24, no. 14, p. 4662, 2024.

[148] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[149] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[150] T. Höfer, F. Shamsafar, N. Benbarka, and A. Zell, "Object detection and autoencoder-based 6d pose estimation for highly cluttered bin picking," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 704–708, 2021.

[151] J. Langlois, H. Mouchère, N. Normand, and C. Viard-Gaudin, "3d orientation estimation of industrial parts from 2d images using neural networks," pp. 409–416, 01 2018.

[152] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, Spie, 1992.

[153] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[154] F. Blanchini and S. Miani, *Set-theoretic Methods in Control*, vol. 78. Springer, 2008.

[155] G. P. Incremona, M. Rubagotti, and A. Ferrara, "Sliding mode control of constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2965–2972, 2016.

[156] F. Emika, "Franka control interface documentation." `https://frankaemika.github.io/docs/`.

[157] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, 2019.

[158] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." `http://pybullet.org`, 2016–2021.

[159] ROS Wiki, "Unified robot description format." `http://wiki.ros.org/urdf`.

[160] Franka Emika, "Franka emika robots urdf models." `https://github.com/frankaemika/franka_description/`.

[161] Franka Emika, "Libfranka documentation." `https://frankaemika.github.io/libfranka/`.