Department of Electrical Electronic and Computer Engineering

Doctor of Philosophy in Systems, Energy, Computer and Telecommunications Engineering

---

Strategies and Systems for Modelling, Real-Time Control and Data Acquisition in Magnetic Confinement Fusion

PhD Thesis

Author
Dr. Gabriele Puglisi

Academic Supervisor (UniCt)
Prof. Eng. Arturo Buscarino

Company Supervisor (ENI)
Eng. Fabio Zanon

ACADEMIC YEAR 2024/2025

Dr. Gabriele Puglisi

Strategies and Systems for Modelling, Real-Time Control and Data Acquisition in Magnetic Confinement Fusion

PhD Thesis

UNIVERSITY OF CATANIA

"*Man muß sich utopische ziele setzen, um realistische zu erreichen.*"

"*You have to set utopian goals to achieve realistic ones.*"

"*Si devono stabilire obiettivi utopistici per realizzarne di realistici.*"

Hans Böck

Abstract

This thesis investigates the design and optimization of control and diagnostic systems for magnetic confinement fusion experiments, with a focus on enhancing the reliability and responsiveness of plasma regulation in Tokamak devices. The work bridges theoretical modeling, embedded system implementation, and machine learning, supporting key functions in real-time supervision and diagnostics.

The research is funded under the framework of the PNRR, Missione 4, Componente 2 "Dalla Ricerca all'Impresa"– Investimento 3.3 "Introduzione di dottorati innovativi che rispondono ai fabbisogni di innovazione delle imprese e promuovono l'assunzione dei ricercatori dalle imprese"- DM 351/2022, and supported by the Italian Ministry of University and Research, by ENI S.p.A. and collaborators: RFX Consortium, ENEA.

The research has been conducted over three lines of investigation. The first direction investigates embedded platforms and their suitability for real-time control applications, with emphasis on identifying hardware–software configurations capable of sustaining deterministic behavior under demanding operational conditions. A second line of work, carried out in collaboration with Fusion for Energy, examines the modeling and validation of power supply control strategies for poloidal and toroidal field coils, which play a key role in plasma shaping and stability. Detailed PSIM models of JT-60SA's Poloidal and Toroidal Field power supplies were developed based on manufacturer control logic. Finally, the thesis explores the use of neural networks, both in their standard multilayer perceptron formulation and in more advanced paradigms such as hypercomplex architectures and learning-on-learning algorithms, as tools for feedforward prediction, disturbance mitigation, and enhanced diagnostic capabilities.

Taken together, these contributions illustrate how the integration of embedded technologies, power electronics modeling, and intelligent data-driven techniques can provide a robust framework for next-generation fusion control and diagnostic systems. By bridging diverse methodologies, this work aims to support the realization of scalable and reliable solutions for ITER, JT-60SA, DTT, and future fusion reactors.

Contents

Abstract iii

1 Introduction **1**
1.1 Nuclear fusion: Background and key points 2
1.1.1 Fundamentals of Nuclear Fusion and Tokamaks 3
1.1.2 Control and Diagnostic systems 5
1.2 Overview on Tokamak features 7
1.2.1 International Thermonuclear Experimental Reactor (ITER) 7
1.2.2 Japan Torus-60 Super Advanced (JT-60SA) 7
1.2.3 Divertor Tokamak Test facility (DTT) 9
1.3 Plant fast-control environment 9
1.3.1 Multithreaded Application Real-Time executor V.2 (MARTe2) 9
1.4 Preliminaries on Neural Networks 10
1.4.1 Multilayer perceptrons 11
1.4.2 Hypercomplex multilayer perceptrons 13

2 Embedded Platforms for Real-Time Acquisition in Plasma Control Systems **15**
2.1 Experimental setup configuration 16
2.2 Hardware overview and key aspects 17
2.2.1 STM32H723ZG Configuration 17
BareMetal 18
FreeRTOS 18
2.2.2 PicoZed solution 18
2.2.3 Pure FPGA configuration 19
2.2.4 Tested device loop frequencies 19
2.3 Results 19

3 Modeling, Simulation and Mitigation Strategies for JT-60SA coil Power Supplies and Quench Detection System **23**
3.1 PSIM simulation of the control of the poloidal field coils power supply and toroidal field coils power supply 24
3.1.1 Poloidal field coil power supply 25

3.1.2 Toroidal field coil power supply . . . 33
3.1.3 PS control code implementation . . . 34
Initialization of the resources: SimulationBegin() function 36
Control Loop: SimulationStep() function . . . 36
3.1.4 Validation of the PSIM simulation models of the power supplies . . . 37
PF PS validation . . . 38
TF PS validation . . . 39
3.2 Using multilayer perceptron to avoid false-positive quench signals in Toroidal Field quench detection voltage system . . . 45
3.2.1 A quick overview of JT-60SA quench detection system issue . . . 45
3.2.2 A novel ANN-based approach . . . 46
3.2.3 Preprocessing and choice of the signals to process . . . 49
3.2.4 Performance index . . . 54
3.2.5 Training parameters and Structure of the MLP . . . 54
3.3 Feedforward phase results . . . 56

4 Hypercomplex multilayer perceptrons and Learning-on-Learning 61
4.1 Hypercomplex multilayer perceptrons: architecture and routines 62
4.1.1 Feedforward propagation . . . 63
4.1.2 Network error computation . . . 63
4.2 Learning algorithms . . . 63
4.2.1 Gradient descent algorithm . . . 64
4.2.2 Levenberg-Marquardt algorithm . . . 65
4.3 Learning-on-Learning approach . . . 69
4.3.1 A first demonstrative example . . . 70
4.4 Results and performance comparison between different networks 73
4.4.1 HMLP for Planetary Orbits Prediction . . . 73
4.4.2 Prediction of chaotic time series: Hyperchaotic Lorenz system . . . 76
Performance analysis . . . 78
Weight trend analysis . . . 81
4.5 Spacecraft rotatory dynamics . . . 85

5 Conclusions 91

A Fondamentals on Quaternion algebra 93
A.0.1 Hamiltonian-Real Calculus . . . 94

Bibliography 97

List of Figures 103

List of Tables 107

Chapter 1

Introduction

FIGURE 1.1: *The flight of Icaro to the sun. Author: unknown.*

Advancements in magnetic confinement fusion represent one of the most complex and multidisciplinary challenges of contemporary research. The scope of magnetic confinement fusion extends across a broad spectrum of scientific and engineering domains, encompassing plasma physics, control systems engineering, and computer science, as well as considerations of the socio-economic and cultural implications arising from the potential realization of an abundant and sustainable energy source.

Within this context, the overarching aim of this doctoral work is to demonstrate how advanced control and data acquisition methodologies can be effectively designed for magnetic confinement fusion plants. The research addresses different complementary directions, each developed in a dedicated chapter, as breifly outlined in the following:

- The background information required for these contributions is provided in this introductory chapter. The basic ideas of magnetic confinement fusion and the primary experiments now in operation are briefly reviewed at the outset. For instance, the chapter proposes a synopsis of the main contributions and activities discussed in this work, guaranteeing a smooth transition into the technical chapters that follow.

- Chapter 2 examines embedded system implementations for the reliable and deterministic transfer of diagnostic information in real-time environments, with particular emphasis on evaluating the temporal footprint introduced by operating systems in comparison to BareMetal solutions. In this context, determinism is understood as the enforcement of fixed cycletimes in real-time execution, a property that is essential to guarantee system reliability and robust control.

- Chapter 3 examines the modelling and control of power supply systems for poloidal and toroidal field coils, which are crucial for plasma shaping and stability. Alongside simulation-based approaches and experimental validation, this chapter also investigates the use of neural network models for feedforward prediction of plasma control signals, demonstrating their potential in enhancing performance quech detection systems.

- Chapter 4 explores the application of innovative strategies aimed at "accelerating" learning dynamics, such as the Learning-on-Learning algorithm applied to hypercomplex neural networks, for tasks of predictive control and diagnostic enhancement, including feedforward trajectory and attitude prediction, and image-based thermal monitoring of Tokamak components.

When considered collectively, these endeavors show a logical attempt to incorporate intelligent data processing, power electronics control, and embedded technologies into the framework of fusion control and diagnostics, offering both methodological insights and real-world examples pertinent to upcoming reactor-scale deployments.

1.1 Nuclear fusion: Background and key points

This section offers a thorough background to contextualize the research discussed in this thesis by outlining key ideas, technical terms, and the primary difficulties faced in the magnetic confinement fusion field, with particular emphasis on the control engineering related problems.

Recognizing the vast array of technologies and methods available, the discussion is intentionally narrowed to emphasize topics directly relevant to the objectives of this study. Additionally, the scope of the proposed solutions is clearly defined, offering the reader a precise understanding of the specific domain addressed throughout this work.

1.1.1 Fundamentals of Nuclear Fusion and Tokamaks

Nuclear fusion fundamentally involves the combination of two light atomic nuclei to form a heavier nucleus, accompanied by a release of energy resulting from the mass defect, in accordance with Einstein's mass-energy equivalence principle [1], [2]:

$$E = \Delta m c^2, \tag{1.1}$$

where Δm is the mass difference between the reactants and the products, and c is the speed of light in vacuum. Among various fusion reactions, the deuterium-tritium (D-T) reaction,

$$^{2}_{1}\mathrm{D} + ^{3}_{1}\mathrm{T} \longrightarrow ^{4}_{2}\mathrm{He}\,(3.5\ \mathrm{MeV}) + ^{1}_{0}\mathrm{n}\,(14.1\ \mathrm{MeV}), \tag{1.2}$$

is considered the most feasible for energy production due to its relatively low ignition temperature and high energy yield. The same reaction is represented in fig 1.2. The total energy released in this reaction is approximately 17.6 MeV, making it a prime candidate for early fusion power systems.

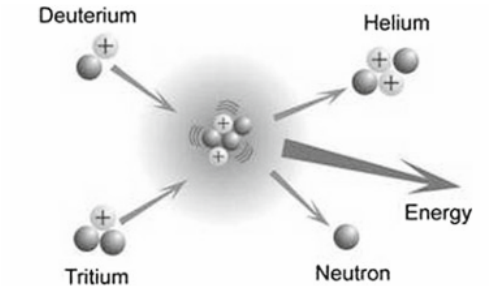

FIGURE 1.2: Graphical representation of the deuterium-tritium (D-T) reaction. *Source:* [3]

Achieving fusion necessitates that reacting nuclei overcome their mutual electrostatic repulsion, commonly referred to as the Coulomb barrier, to approach sufficiently close for the strong force to induce fusion. The performance of a fusion plasma is commonly characterized by the *triple product*, the product of plasma temperature, density, and energy confinement time, which collectively define the ignition threshold [4]. Upon reaching this ignition condition, the fusion process becomes self-sustaining, allowing external heating sources to be reduced or deactivated.

Initial plasma heating is typically achieved through ohmic heating, where an induced current passing through the plasma generates resistive heating analogous to the Joule effect in conductors. However, ohmic heating alone is insufficient to reach the temperatures (approximately 15 keV) required for efficient D-T fusion, as plasma resistivity decreases with increasing temperature and declining collision frequency. To attain fusion-relevant temperatures, auxiliary heating methods are employed, primarily neutral beam injection (NBI) and radiofrequency (RF) heating. NBI introduces high-energy neutral particles into the plasma, which transfer their energy via collisions, while RF heating techniques, such as electron cyclotron resonance heating (ECRH) and ion cyclotron resonance heating (ICRH), employ electromagnetic waves tuned to the plasma's characteristic frequencies to transfer energy efficiently [5].

Effective plasma confinement is equally critical, as high temperatures alone do not guarantee sufficient reaction rates. Magnetic confinement devices utilize strong, precisely controlled magnetic fields generated by external coils to restrict plasma motion perpendicular to magnetic field lines, thus maintaining plasma stability and density.

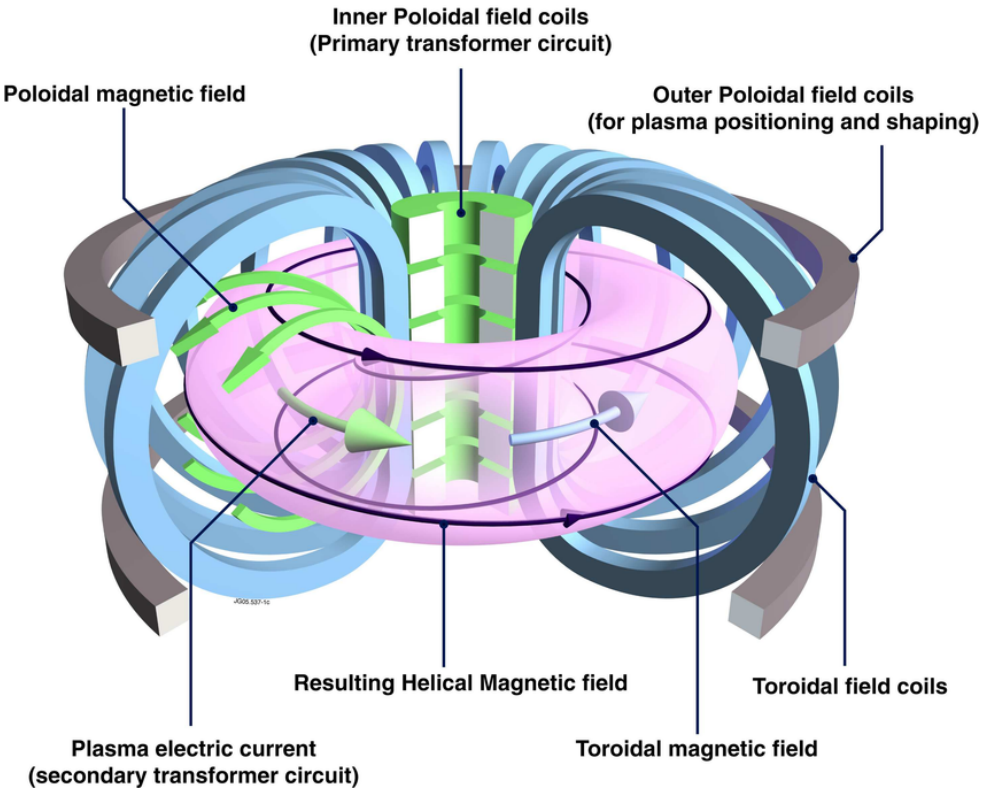

FIGURE 1.3: Illustration of the Tokamak concept. The toroidal magnetic field is indicated with a blue arrow. Poloidal field (green arrows) are employed to control the plasma position and shape. *Source:* [6]

The *Tokamak* configuration, Figure 1.3, featuring a toroidal chamber surrounded by poloidal and toroidal magnetic coils, represents the most advanced and widely adopted magnetic confinement approach. In this design, the plasma current itself produces the poloidal magnetic field, effectively creating a helical magnetic field structure characterized by the *safety factor* q, which influences plasma stability and confinement quality [7]. Additional vertical magnetic fields prevent plasma displacement, ensuring positional control.

Tokamak vessels are often engineered with a D-shaped plasma cross-section, as shown in Figure 1.4, optimized through shaping parameters such as *triangularity* and *elongation*, to enhance confinement performance and plasma stability. These design choices have been critical in the evolution of Tokamaks toward practical reactor configurations.

Alternatively, Inertial Confinement Fusion (ICF) employs rapid compression of a fuel pellet to extremely high densities and temperatures via intense laser or particle beam irradiation, causing implosion and fusion on microsecond timescales. While ICF and Magnetic Confinement Fusion (MCF) pursue distinct approaches, both aim to harness fusion energy effectively.

Currently, Tokamak devices, exemplified by ITER, JT-60SA, and DTT,

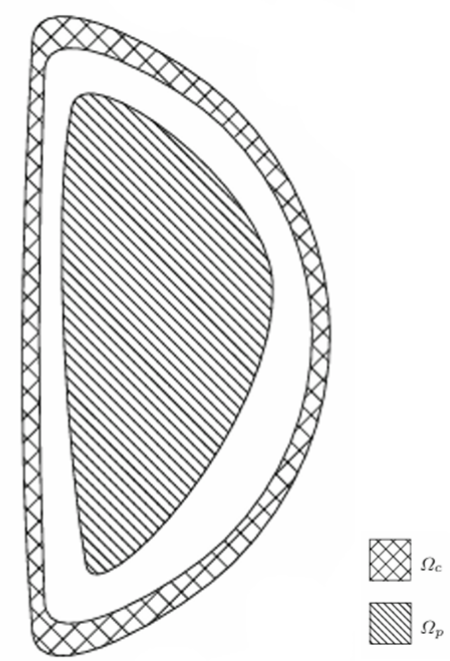

FIGURE 1.4: Poloidal D-shaped cross-section of a Tokamak machine, partitioned into regions occupied by the plasma (Ω_p) and the conducting structure (Ω_c). *Source:* [3]

represent the forefront of fusion research and development, with ongoing efforts to refine plasma control, heating, and confinement techniques aimed at achieving sustained net energy gain.

Plasma confinement within a Tokamak presents significant challenges, due to the inherently unstable nature of the plasma state. A wide variety of instabilities, both linear and nonlinear, arise across different spatial and temporal scales. These instabilities can range from small-scale turbulence driven by gradients in temperature or density, to large-scale magnetohydrodynamic (MHD) phenomena. In particular, the plasma exhibits self-organizing behavior, which can sometimes enhance stability, but also lead to sudden disruptive events [4]. A disruption occurs when the plasma becomes uncontrollable and rapidly collapses, typically following the onset of a major MHD instability. These events are not only detrimental to plasma performance, but also pose significant risks to the integrity of the reactor's structural components. Moreover, micro-instabilities contribute to anomalous transport, increasing energy and particle losses and potentially acting as seeds for larger-scale instabilities [8], [9].

1.1.2 Control and Diagnostic systems

To prevent plasma disruption and to achieve a steady-state fusion regime, effective plasma control is indispensable. Control systems rely on detailed modelling of plasma dynamics to anticipate and mitigate instability growth. These models, commonly referred to as plasma "codes", simulate various aspects of plasma behavior and are essential for both offline controller design

and online state estimation during operation [10], [11]. They include equilibrium solvers, transport codes, and MHD stability tools, among others.

Modelling efforts are continuously refined through feedback from experimental observations across multiple international fusion devices. This iterative process fosters rapid advancement in both theoretical understanding and practical control strategies.

A crucial prerequisite for both modelling and control is the availability of accurate and real-time measurements of key plasma parameters. This is the task of plasma diagnostics, a multidisciplinary research field that develops instruments and techniques to measure physical quantities such as temperature, density, magnetic fields, and current distributions [12], [13]. Many of these quantities cannot be measured directly and must be inferred from indirect measurements using sophisticated data analysis or inversion techniques.

Furthermore, diagnostic systems must be highly adaptable and upgradable, often outpacing other subsystems in complexity and development cycles. Their integration into the reactor control architecture ensures safe operation, contributes to physics studies, and supports machine learning or data-driven control strategies that are gaining traction in modern fusion research.

Fusion devices are composed of numerous subsystems that must work in coordination to ensure safe and efficient operation. These subsystems are organized in a hierarchical structure, starting from low-level field sensors and actuators, and progressing up to high-level supervisory control, monitoring, and data acquisition systems. Within the ITER framework, this entire infrastructure is referred to as Plant Instrumentation and Control (Plant I&C) [14].

The integration of fast and slow controllers, along with their seamless coordination within the control hierarchy, is fundamental to achieving reliable and precise operation of modern fusion facilities. This layered and modular architecture enables scalability, flexibility, and resilience across the control infrastructure.

- Slow controllers are standardized systems such as programmable logic controllers (PLCs), often operating independently, and reconfigurable I/O platforms (RIOs), typically hosted within rack-mounted industrial PCs. In contrast, non-standardized systems are based on custom-designed hardware running bespoke software, tailored to meet specific requirements of the controlled process.

- Fast controllers are those used in applications requiring high sampling rates and strict real-time constraints, such as plasma control, diagnostics, and closed-loop feedback systems. These controllers typically operate at frequencies ranging from several kilohertz up to tens of kilohertz and rely on high-performance computing platforms. The hardware supporting fast control is often based on dedicated real-time systems equipped with RIOs or entirely custom boards designed for high-throughput, low-latency operation.

All components within the Plant I&C hierarchy communicate over a variety of physical and logical networks. These may be based on optical fibre or copper media and are organized to segregate different classes of traffic, such as data archiving, time synchronization, safety interlocks, audio/video streams, and real-time control signals.

1.2 Overview on Tokamak features

A brief overview of the main specifications of the experiments referenced in this thesis is provided.

1.2.1 International Thermonuclear Experimental Reactor (ITER)

ITER is a large-scale fusion experiment currently under construction in Saint-Paul-lès-Durance, France [15]. It is the result of an international collaboration among seven parties: the European Union, United States, Russia, Japan, China, South Korea, and India. Its primary objective is to demonstrate the scientific and technological feasibility of magnetic confinement fusion energy on a power-plant scale.

The ITER project aims to achieve a tenfold return on input energy, targeting a fusion gain factor of $Q = 10$, which means producing 500 MW of fusion power from 50 MW of heating input. Unlike a power plant, ITER is not intended to generate electricity, but rather to validate integrated technologies such as superconducting magnets, remote handling systems, tritium breeding modules, and complex cryogenic infrastructure, all necessary for DEMO and subsequent commercial reactors.

Construction began in 2010, and first plasma is planned for 2034 and full operations are expected in 2039. Full deuterium-tritium operations are expected in the 2030s. ITER's design features include a highly shaped D-shaped plasma cross-section, a large vacuum vessel (840 m^3), powerful neutral beam and radiofrequency heating systems, and an advanced diagnostic suite for real-time plasma control.

Finally, the main parameters relative to ITER are summarized in Table 1.1.

Parameter	**Value**
Major radius	6.2 m
Minor radius	2.0 m
Plasma volume	$\sim 840\,m^3$
Plasma current	up to 15 MA
Toroidal field	5.3 T
Heating power	up to 110 MW (NBI, ECRH, ICRH)

Table 1.1: Main parameters of ITER.

1.2.2 Japan Torus-60 Super Advanced (JT-60SA)

JT-60SA is a large superconducting Tokamak constructed collaboratively by Japan and Europe under the Broader Approach Agreement, serving as a satellite device to ITER [16]. Construction commenced in 2007, and assembly was completed in early 2020. The device achieved its first plasma on October 23, 2023. In Figure 1.5 it is shown the full assembled Tokamak.

JT-60SA is hosted by the National Institutes for Quantum Science and Technology (QST) in Naka, Japan. It is currently the largest operational superconducting Tokamak, with a plasma volume of approximately 160 m^3, a feature that earned it a Guinness World Record in October 2024. The machine

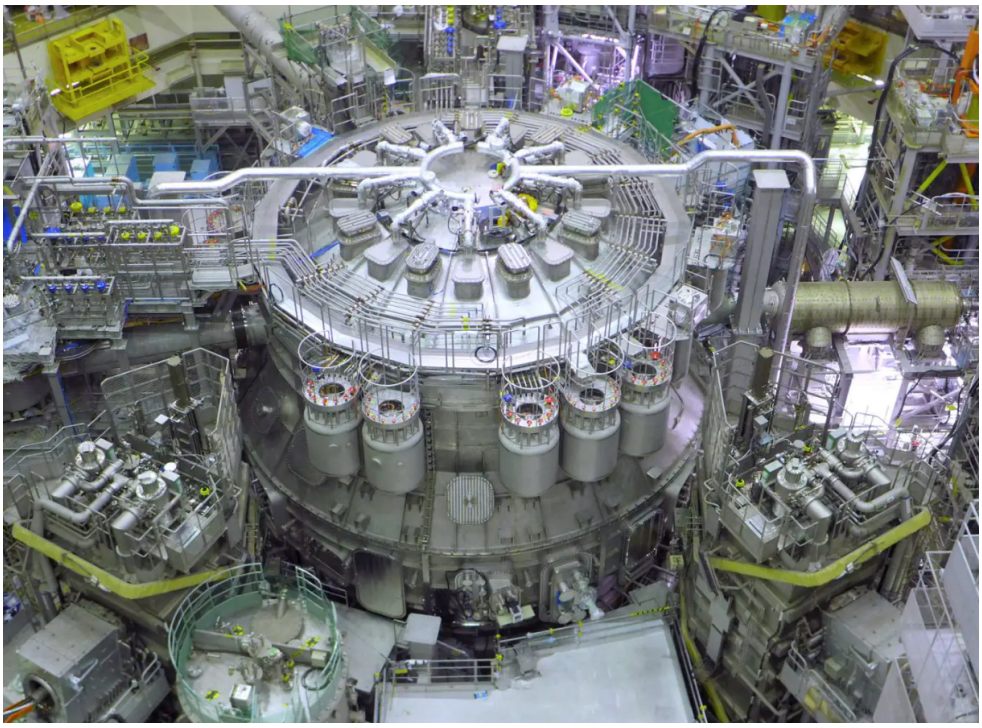

FIGURE 1.5: JT-60SA Tokamak in QST Tokamak building.

is designed to confine high-temperature deuterium plasmas, reaching up to 200 million °C, for durations of up to 100 s, with typical plasma currents of 5.5 MA and toroidal magnetic fields ranging from 2.25 to 2.3 T.

The primary mission of JT-60SA is to support the scientific goals of ITER and contribute to research activities aimed at the design of DEMO. This includes the optimization of plasma shaping (e.g., elongated and triangular cross-sections) and the exploration of steady-state, high-density plasma regimes that exceed the H-mode threshold observed in ITER. JT-60SA is equipped with a powerful suite of heating systems, providing up to 41 MW through a combination of Neutral Beam Injection (NBI) and Electron Cyclotron Resonance Heating (ECRH), as well as advanced divertor configurations (both single- and double-null) and a comprehensive set of diagnostics and real-time control systems.

Following its first plasma, the device has entered an upgrade phase in preparation for its full scientific exploitation campaign, expected to begin in the latter half of 2026. This phase includes the installation of additional diagnostic tools, heating systems, and in-vessel components necessary to support its evolving research objectives.

Parameter	**Value**
Major radius	2.96 m
Minor radius	1.18 m
Plasma current	up to 5.5 MA
Toroidal field	2.25–2.3 T
Pulse duration	typical flat-top $\sim$ 100 s
Heating power	up to 41 MW (NBI + ECRH)

TABLE 1.2: Main parameters of JT-60SA.

Finally, the main parameters relative to JT-60SA are summarized in Table 1.2.

1.2.3 Divertor Tokamak Test facility (DTT)

DTT is a fully superconducting Tokamak being built at the ENEA Frascati Research Centre in Italy [17]. DTT is a national strategic infrastructure and a key element of the EUROfusion roadmap, conceived to address one of the most critical challenges in future fusion reactors: the handling of power exhaust in the divertor region.

DTT's design is based on a highly elongated and triangular plasma shape within a D-shaped vacuum vessel, allowing flexible divertor configurations, including advanced magnetic topologies. It will support high-performance scenarios with steady-state-like conditions and realistic power fluxes, providing experimental validation for divertor materials, configurations, and control schemes relevant for DEMO.

Construction began in 2018, and operations are expected to start in 2026. DTT is equipped with superconducting toroidal and poloidal field coils, and will initially use over 25 MW of auxiliary heating, expected to rise to around 45 MW. The device also features extensive diagnostic coverage and high-speed real-time control systems.

Finally, the main parameters relative to DTT are summarized in Table 1.3:

Parameter	**Value**
Major radius	2.19 m
Minor radius	0.70 m
Plasma current	up to 5.5 MA
Toroidal field	6 T
Pulse duration	up to 100 s
Heating power	25–45 MW (NBI, ECRH, ICRH)

Table 1.3: Main parameters of DTT.

1.3 Plant fast-control environment

Control systems in magnetic confinement fusion experiments are typically composed of a mix of custom and off-the-shelf hardware and software components. In this context, having a standardized development framework is crucial for ensuring scalability, maintainability, and interoperability across systems. In the following sections, the focus will shift to fast plant control environments, with particular emphasis on the MARTe2 framework, which is widely used in fusion research for implementing high-performance real-time control applications.

1.3.1 Multithreaded Application Real-Time executor V.2 (MARTe2)

Among the software frameworks adopted in this field, MARTe2 (Multithreaded Application Real-Time executor version 2) is one of the most prominent [18]. It is an open-source real-time software framework written in C++, specifically

designed for high-performance control applications. It is widely employed in fusion experiments that require fast controllers, systems operating with high sampling rates and strict real-time constraints [19], [20], [21], [22].

MARTe2 encourages a modular and reusable codebase that supports long-term maintainability as well as portability, while common real-time systems employ a vertical approach that is closely tied to specific platforms, which can be restrictive for systems that are evolving or for multidisciplinary projects.

MARTe2 supports two main development perspectives: *the developer view*, which allows a clean separation between hardware interfaces, algorithms, and execution environments; and *the user view*, which relies on a configuration-driven model where data sources, processing modules, and outputs are interconnected in a logical flow. Users configure the system by specifying how signals flow between components and how execution is distributed across CPU cores and threads.

The architecture of MARTe2 is based on two principal software entities:

- **DataSources**, which encapsulate hardware input/output interfaces [23].
- **Generic Application Modules (GAM)**, which implement the user-defined algorithms [24].

These components are interconnected to form a *RealTimeApplication*, the executable configuration of a MARTe2 system. GAMs are strictly isolated from hardware access and focus purely on algorithm execution, while DataSources handle all memory transfers and signal interfacing.

MARTe2 is structured in three tiers (BareMetal, FileSystem, and Scheduler) and seven layers [25]. The BareMetal tier is designed to be self-sufficient, allowing deployment on embedded systems with limited resources. This layered design improves modularity and dependency isolation across the framework.

MARTe2 adheres to strict software quality assurance practices, including compliance with *MISRA C++:2008* standards, and is maintained by a growing user community. Its robustness and versatility have led to widespread adoption in magnetic confinement fusion projects. It has been deployed in real-time systems for the *JET Tokamak*, and it is expected to be integrated into the *ITER Plant Instrumentation and Control (I&C)* system.

MARTe2 and its predecessor MARTe continue to evolve through collaborative development, extending their applicability to new hardware and control challenges in fusion research.

1.4 Preliminaries on Neural Networks

The increasing complexity of modern fusion devices and the need for adaptive, nonlinear modeling and control strategies have motivated the adoption of data-driven approaches such as artificial neural networks (ANNs) [26], [27]. Neural networks are massively parallel, distributed computing systems composed of interconnected simple processing units, and cause of their architecture, they are particularly suited for extracting, encoding, and generalizing knowledge acquired through experience.

Neural networks mimic the brain in two fundamental aspects. They acquire knowledge from their environment through a learning process and exploit this knowledge stored in the form of synaptic weights, numerical values

associated with the connections between neurons, that are continuously adjusted to improve performance over time. The mechanism governing this adaptation is known as the *learning algorithm*, which updates the synaptic weights in an orderly fashion to achieve a given design objective.

One of the most important features of neural networks is their ability to model nonlinear relationships, and this is especially valuable in fusion systems, where the underlying physical processes are often nonlinear in nature [28]. Through the use of nonlinear activation functions and layered representations, a neural network can approximate complex input-output mappings that are otherwise difficult to describe analytically.

In supervised learning, the network is trained using labeled examples that associate a unique input with a target output. The learning process consists of iteratively modifying the synaptic weights to minimize the error between the actual and desired outputs, according to a statistical cost function. Through repeated exposure to representative data, the network generalizes its understanding, allowing it to correctly infer outputs for previously unseen inputs.

Another important capability of neural networks is their adaptivity. When deployed in changing or nonstationary environments, where signal statistics evolve over time, a neural network can adjust its internal parameters accordingly. This makes it highly suitable for applications such as adaptive signal processing, real-time diagnostics, and control systems that must cope with evolving operational conditions.

Finally, the inherent parallelism of neural networks makes them well suited for hardware implementation using very-large-scale integration (VLSI) technologies. This enables high-throughput, low-latency computation, opening the possibility of embedding neural networks directly into real-time systems, including those with strict execution constraints. These properties, coupled with recent advances in training algorithms and embedded platforms, have led to the increasing adoption of neural networks in fusion research [29], [30].

In Chapters 3 and 4, we will explore the design, deployment, and evaluation of neural models in the context of modeling and forward prediction of systems that present strong non-linearity.

1.4.1 Multilayer perceptrons

Multilayer perceptrons (MLPs) are a class of feedforward artificial neural networks distinguished by several architectural and functional properties. First, each neuron in the network implements a differentiable nonlinear activation function, enabling the network to model complex, non-linear input–output relationships [26]. Second, the network includes one or more hidden layers, layers that are not directly connected to the input or output, allowing hierarchical feature extraction. Third, MLPs exhibit high connectivity, wherein each neuron in a given layer is typically connected to all neurons in the preceding layer through synaptic weights.

A standard MLP architecture, as shown in Figure 1.6, comprises an input layer, one or more hidden layers, and an output layer. In fully connected configurations, each neuron in a layer receives inputs from all neurons in the previous layer. In Figure 1.7 it is reported a simplified scheme of a neuron (or perceptron) with its input i_n and weights w_n, with $n = 0, \ldots, N-1$, the activation function and the output. The information flow proceeds strictly in

the forward direction, from the input to the output layer, on a layer-by-layer basis.

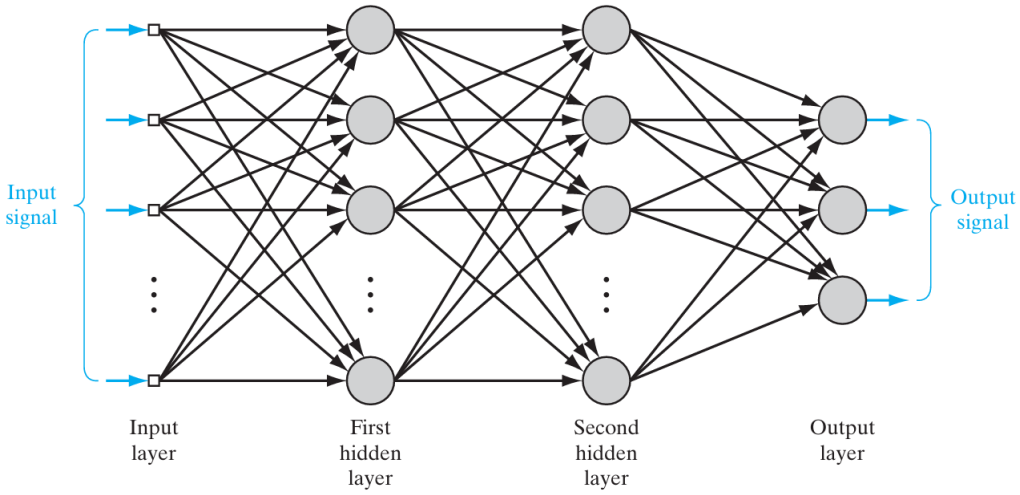

FIGURE 1.6: Architectural graph of a multilayer perceptron with two hidden layers. *Source:* [26]

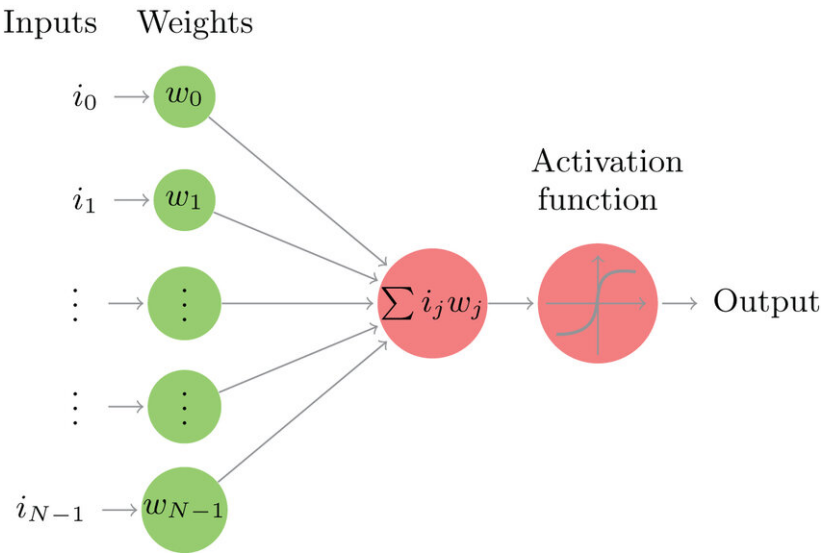

FIGURE 1.7: Scheme of a single neuron (or perceptron) with its inputs, weights and outputs. *Source:* [31]

Within this structure, two types of signals are essential. Function signals propagate forward through the network, from the input layer to the output layer, carrying stimulus information. In contrast, error signals are computed at the output and propagate backward through the layers during the training phase. These error signals guide the adjustment of the synaptic weights according to a chosen cost function.

Among the various choices for nonlinear activation functions, the selection is often guided by the specific learning algorithm and the desired performance metric [32]. When employing gradient-based optimization techniques such as backpropagation, it is essential to adopt activation functions that are differentiable, ensuring smooth and continuous gradients for effective weight updates.

Another important consideration is the approximation criterion used to evaluate the performance of the network. For instance, when uniform approximation is required, sigmoidal activation functions are particularly suitable. In this case, the multilayer perceptron constructs an output by performing a weighted sum of sigmoid functions, which are known to be universal approximators of continuous real-valued functions. This theoretical property underlines the versatility of MLPs in modeling a wide range of nonlinear input–output relationships.

Each hidden or output neuron executes two primary computations:

1. the generation of the function signal, which is the nonlinear transformation of its inputs and associated synaptic weights;

2. the computation of an estimate of the gradient vector, which captures the sensitivity of the error with respect to the input weights. The latter operation is critical for performing the backward pass in gradient-based learning algorithms.

Hidden neurons function as nonlinear feature detectors. During the training process, these neurons gradually learn to extract salient representations of the input data by transforming it into a feature space more amenable to classification or regression tasks. The capacity of MLPs to uncover these latent structures makes them well-suited for a variety of applications in signal processing, control, and diagnostics.

The practical adoption of MLPs has been significantly facilitated by the development of the back-propagation algorithm, which enables efficient supervised training through online learning. This iterative method adjusts synaptic weights to minimize the error between the actual and desired outputs, ultimately improving generalization performance across the training set.

1.4.2 Hypercomplex multilayer perceptrons

The inherently multidimensional nature of real-world data demands a correspondingly expressive representational model [33]. Attempts to reduce such data to one-dimensional parameters often result in significant loss of information, compromising both interpretability and classification performance. If this complexity is ignored, the classification and processing of data typically require a dramatic increase in computational resources, especially when employing architectures ill-suited for high-dimensional representation. Traditionally, MLPs are known to be inefficient when handling multidimensional signals, often requiring a large number of parameters. Although deep architectures such as Convolutional Neural Networks (CNNs) and other deep learning frameworks have partially addressed these limitations, there remains a fundamental inefficiency in how real-valued networks represent spatial and temporal relationships among multidimensional inputs [34].

Moreover, as the number of input and output variables increases, standard real-valued networks require a proportionally larger number of connections, thereby expanding the parameter space to be optimized during training. This not only increases the computational burden but also elevates the risk of convergence to local minima. Moreover, it necessitates a larger dataset of representative examples to ensure successful learning [32], [33].

To overcome these drawbacks, researchers have turned toward alternative algebraic structures, particularly hypercomplex numbers [32], [35]. By

leveraging algebras such as complex numbers and quaternions, it is possible to achieve a more compact and structured representation of multidimensional data while reducing the number of learnable parameters. For instance, complex-valued neural networks have been shown to outperform their real-valued counterparts in tasks requiring 2D geometric transformations.

Hypercomplex algebra, or Quaternion algebra, offers unique advantages in representing affine geometrical transformations as single computational entities. Quaternions, introduced by Hamilton (Appendix A)in 1844, extend this capability into four dimensions. A single quaternion can simultaneously encode four pieces of information, making it particularly suitable for problems involving rotations, orientations, or spatiotemporal modeling.

In the context of deep learning, Quaternion-valued Neural Networks (QNNs) and, specifically, Hypercomplex Multilayer Perceptrons (HMLPs) provide a promising framework for the efficient and natural encoding of multidimensional features. These networks allow operations to exploit the internal structure and correlations of the input data, rather than treating each channel or variable independently as in real-valued networks. This has been shown to yield state-of-the-art performance in several applications, including speech recognition, image processing, and 3D modeling.

Chapter 2

Embedded Platforms for Real-Time Acquisition in Plasma Control Systems

The achievement of controlled magnetic confinement fusion in Tokamak reactors hugely depends on real-time CODAS (COntrol and DAta acquisition Systems), which must ensure high timing determinism and minimal latency in order to acquire, process, and respond to diagnostic signals under stringent timing constraints [36, 37, 38]. We recall that the concept of determinism herein refers to the well-known problem of the introduction of deterministic cycletimes in real-time environments, which ensures higher reliability.

A core function of these systems lies in the real-time acquisition loop, where analog signals originating from diagnostics or sensors are sampled, processed, and exchanged among communicating units according to a deterministic timeline.

This chapter attempts to assess how different embedded hardware and software architectures influence real-time performance through a unified experimental scenario. In particular, it investigates the impact of different operating systems on acquisition and communication tasks within the context of a generic fast control loop.

To guarantee a fair and unbiased comparison across platforms, the MARTe2 framework (Multithreaded Application Real-Time executor, version 2) is employed and consistently adapted to each device through a dedicated *porting*, defined as the pairing of a specific hardware platform (*architecture*) with its corresponding execution context (*environment*, such as an operating system or bare-metal setup) [39, 40]. This approach eliminates variability stemming from differing software implementations.

The platforms under evaluation include: an STM32H723GZ microcontroller (tested under both BareMetal and FreeRTOS) [41], a low price microcontroller with an ARM Cortex-M7 board by STMicroelectronics, a PicoZed

system-on-chip running Linux [42], based on the Zynq-7000, and a fully deterministic reference implementation on a pure FPGA [43], free from operating system or middleware overhead.

Each platform is evaluated under real-time constraints within an acquisition loop designed to sample a single-channel analog signal through the device onboard ADC and transmit it via UDP to a server.

The investigation centers on how hardware and software architectural decisions affect the determinism and responsiveness of a fast control path for plasma regulation. Specifically, we examine critical timing-related indicators, including the *cycle time*, the *jitter*, i.e., the variability of cycle time across iterations, and the *round-trip time*, defined as the overall latency from signal acquisition to actuation under load conditions. An identical application is deployed across heterogeneous environments to enable an objective evaluation of timing performance and CPU resource utilization.

2.1 Experimental setup configuration

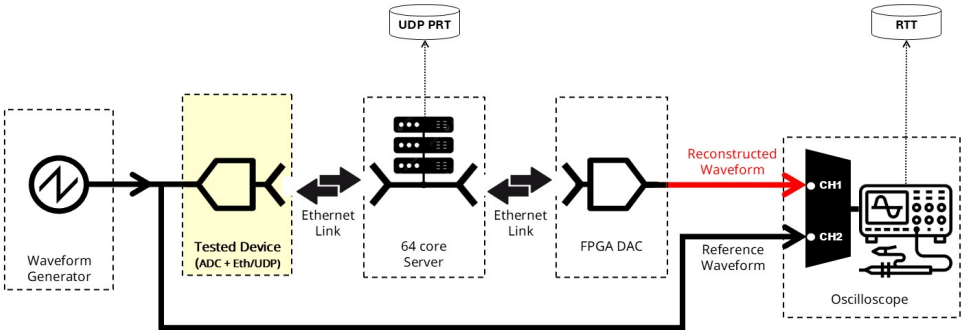

FIGURE 2.1: Experimental setup for performance comparison between different devices. The Ethernet link is not point-to-point, but consists of a switch within an isolated network operating at 1 GHz bandwidth that cannot affect the given results.

Figure 2.1 shows the experimental setup used for performance comparison. Each embedded platform employs its on-board ADC peripheral to sample a one-channel analog reference waveform, specifically a sawtooth signal. The acquired samples are subsequently transmitted via UDP.

This choice is motivated by the protocol's suitability for real-time communication:

1. its simplified structure reduces variability compared to TCP/IP by avoiding retransmissions that would otherwise introduce non-deterministic delays,

2. the occasional loss of a packet can be interpreted as an additional noise source, which is tolerable if the control system is sufficiently robust, and

3. in an isolated network with synchronous data transmission, packet loss is practically absent, since in this case the maximum queue length inside routers is deterministic.

The transmitted data are received by a 64-core server running MARTe2 on Linux MRG. The server application records the UDP Packet Reception

Times (PRT) and, still using UDP, sends the received samples back to the FPGA device hosting a DAC. The DAC output is then compared with the original reference signal using an oscilloscope in order to measure the relative round-trip time (RTT), specifically by evaluating the phase shift between corresponding phases of the reference and reconstructed signals.

In this way, both the UDP PRT and the relative RTT provide timing metrics primarily influenced by the performance of the devices under evaluation.

It is worth noting that the measured RTT and jitter also depend on other components in the acquisition chain, namely the server and the UDP retransmission step. However, since these elements are identical in all experiments, the comparison isolates the performance of the ADC architectures and their associated implementations.

All acquired measurements, including timing data and waveform traces, are systematically stored in a database for offline analysis.

2.2 Hardware overview and key aspects

This section outlines the key configuration parameters for each device under test.

2.2.1 STM32H723ZG Configuration

Both BareMetal and FreeRTOS configurations deploy the MARTe2 framework on the STM32H723ZG, leveraging its high-performance architecture to meet real-time control requirements in the order of microseconds. Several configuration choices have been taken to optimize the time-related performance of the board:

Figure 2.2: STM32H723ZG Nucleo-board.

- The Instruction Cache (I-Cache) and Data Cache (D-Cache) are both enabled to enhance code execution and memory access efficiency, thereby reducing CPU stalls and minimizing per-cycle latency. Although the STM32H723 features Tightly Coupled Memory (TCM) blocks for deterministic access, their usage mainly benefits code sections with intensive arithmetic operations. In this work, since the acquisition loop mostly involves memory copy and lightweight data formatting, the main bottleneck lies in the bus transfer rather than CPU throughput. For this reason, enabling cache memories provided a more effective trade-off

between performance and implementation simplicity. Access to both memories typically requires a small number of CPU cycles (3–8 per access).

- The Ethernet MAC operates via RMII interface lines connected to designated STM32 pins. A lightweight IP (lwIP) stack is used to implement full TCP/UDP communication capabilities. To maintain predictable and constant-time memory behavior, dynamic packet memory is entirely allocated from statically defined MEMPOOLs consisting of fixed-size buffers located in cacheable SRAM.
- Analog input acquisition is managed by ADC3, which samples a single channel in circular DMA mode. Synchronization between periodic sampling and the control loop is handled via semaphores, which are signaled by the completion interrupt of each conversion.

BareMetal

In the BareMetal configuration, as already exposed in [44] MARTe2 runs directly on the STM32H723ZG without an underlying OS, minimizing software overhead and it is configured following the configuration described above.

FreeRTOS

In the FreeRTOS configuration, MARTe2 executes as a high-priority thread, enabling preemptive multitasking and priority-based scheduling for modular and scalable control software. However, some perturbations are caused by the link status detection of the LAN8742 PHY:

- The PHY link status is periodically checked by the `ethernet_link_thread()` every 100 ms by default.
- This periodic task introduces blocking transactions that result in jitter spikes. Possible mitigation strategies will be discussed in Sec. 2.3.

2.2.2 PicoZed solution

The second architecture is based on a Zynq-7010 SoC that integrates a configurable FPGA and a dual-core ARM processor. An external ADC communicates with the FPGA through LVDS lines, and the acquired samples are transferred via the internal FPGA AXI bus to a stream FIFO IP, which serves as the interface between the FPGA and the processor memory. On the processor side, a Linux thread polls the FIFO and, as soon as a sample is received, it is encapsulated in a UDP datagram and transmitted via the Linux IP stack. This solution benefits from the flexibility of an operating system in managing the data stream (and indeed provides a variety of support functions in the deployed setup), but it introduces additional delay and jitter in the overall transmission chain due to the multiple OS layers involved. In this case, RTT and jitter are strongly affected by the Linux footprint in data handling and transmission. This system was implemented for the plasma control system of the RFX-mod 2 magnetic confinement fusion experiment [45].

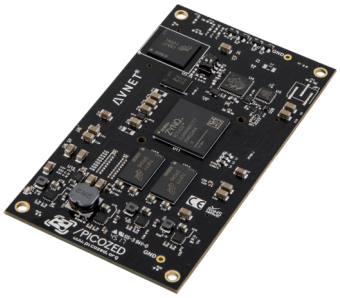

FIGURE 2.3: Zynq 7010 solution embedding a configurable FPGA.

2.2.3 Pure FPGA configuration

This system is implemented using a D-TACQ [46] device, which performs ADC acquisition and UDP transmission entirely in hardware on a Zynq FPGA. In this implementation, the generation and handling of UDP packets are managed directly by the FPGA, while the embedded ARM processor is used exclusively for configuration and supervision, without participating in data acquisition or transmission. For this reason, this represents our reference implementation: transmission times and their jitter can be considered intrinsic to the network switches and to the server-side network stack, which is common to all experiments.

The same device is also employed to receive samples from the server and perform DAC conversion.

2.2.4 Tested device loop frequencies

For each device under test, the loop rate was set to the maximum frequency that avoids packet loss. Packet drops may occur when the microcontroller cannot perfectly synchronize with the high acquisition rate of the DMA, thereby compromising reliable data transmission. For the PicoZed and FPGA implementations, this limit corresponds to 10 kHz (100 μs), whereas the STM32H723 implementations are constrained to approximately 7.1 kHz (140 μs) due to synchronization issues.

It is important to note that the objective of this study is not the determination of the maximum achievable acquisition frequency, but rather the characterization of jitter and the ability of each device to sustain the highest degree of determinism at its operational limit.

2.3 Results

The first analyzed parameter is the average RTT, shown in Table 2.1. These values include ADC sampling, UDP transmission, server logging and retransmission, and final DAC response. As expected, the pure FPGA implementation achieves the lowest and most stable RTT, with an average of 55 μs. Both STM32 BareMetal and FreeRTOS implementations report an average RTT of about 65 μs, demonstrating that the thread scheduling overhead of FreeRTOS is negligible, since the applied task prioritization ensures that the overhead introduced by the operating system does not degrade performance. The

Linux-based PicoZed solution shows an RTT oscillating between 60–100 μs, resulting in a higher mean value of about 80 μs.

Device	**RTT** [μs]
STM32BareMetal	65
STM32FreeRTOS	65
PicoZedLinux	80
FPGA	55

TABLE 2.1: Average RTT for each device under test during the experiment.

The second key metric is the jitter of the interval between ADC acquisition and UDP packet transmission, derived from the deviation of the UDP PRT from its target value. Figure 2.4 shows the comparison over 600,000 cycles for each device under test.

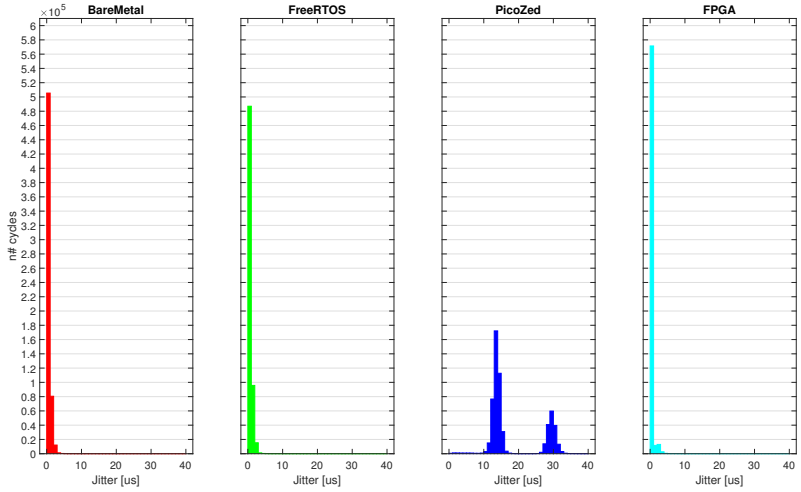

FIGURE 2.4: Comparison of the jitter on 600,000 cycles for each device under test during the experiment.

The pure FPGA implementation exhibits near-zero jitter, with only ~5000 cycles deviating from the 100 μs target and a maximum jitter of $\pm 2\,\mu$s. The PicoZed platform with Linux shows higher jitter, in the 10–30 μs range, mainly due to kernel task scheduling and context switches, though still without packet loss. A particularly relevant result is that both the BareMetal and FreeRTOS implementations on the STM32H723 exhibit tightly bounded cycle times, consistently centered at approximately 140 μs, with only negligible deviations. While the BareMetal implementation attains marginally lower jitter compared to FreeRTOS, the difference between the two approaches remains minimal, with a maximum jitter of $\pm 2\,\mu$s. These results indicate that both implementations achieve a level of determinism approaching that of the pure FPGA solution, thereby confirming their capability to deliver highly stable real-time performance.

As anticipated, the default `ethernet_link_thread()` for link status detection introduces jitter spikes every 100 ms due to blocking transactions. Mitigation can be attempted by:

- *Thread suppression with ARP-aware constraints*, which reduces load by disabling the polling thread after several successful checks, at the cost of possible delays in ARP resolution;
- *FreeRTOS primitives coordination*, where the link thread is suspended via `vTaskSuspend()` and resumed after n UDP packets using `vTaskResume()`, ensuring checks only after transmissions are completed.

Figure 2.5 compares FreeRTOS with and without this mitigation, confirming that jitter spikes otherwise appear with a recurring pattern.

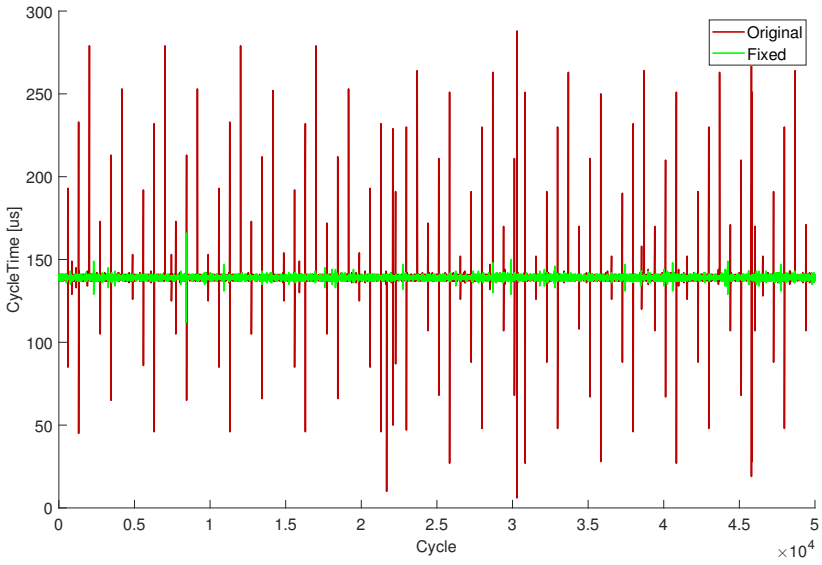

FIGURE 2.5: Comparison between FreeRTOS with persistent execution of link status detection (bordeaux) and the fixed version (green) at 140 μs.

Overall, the results indicate that:

- All platforms can sustain control loops in the 7–10 kHz range when properly tuned.
- FreeRTOS offers almost identical performance to BareMetal, proving suitable for hard real-time tasks.
- Linux on PicoZed, while less deterministic, remains acceptable for soft or firm real-time use cases, benefiting from higher flexibility and processing power.

This work assessed the real-time performance of embedded platforms executing a fast acquisition and communication loop typical of plasma fusion systems. The analysis focused on round-trip time and jitter under maximum sustainable loop rates.

The FPGA-based solution provides the reference performance, with near-perfect determinism. The STM32H723, in both BareMetal and FreeRTOS configurations, demonstrates highly predictable timing, only limited by a longer acquisition loop (140 μs instead of 100 μs). FreeRTOS, in particular, proves to be a practical alternative to BareMetal, adding abstraction and

modularity without compromising timing. The PicoZed/Linux implementation suffers from higher jitter (up to 30 μs), yet still meets the requirements of many real-time applications where strict hard determinism is not essential.

In conclusion, FreeRTOS and even Linux-based solutions, if carefully configured, can support real-time control tasks in plasma fusion environments. FreeRTOS stands out as a robust and flexible option, providing nearly BareMetal-level determinism while retaining the scalability of an RTOS.

Chapter 3

Modeling, Simulation and Mitigation Strategies for JT-60SA coil Power Supplies and Quench Detection System

Under the framework of the EU-Japan Broader Approach (BA), the JT-60SA (Japan Torus-60 Super Advanced) Tokamak represents a significant milestone in collaborative efforts aimed at supporting the construction and operation of next-generation fusion devices, particularly in preparation for ITER [47]. The designation "Super Advanced" refers both to the implementation of Superconducting Coil Magnets (SCM) and to the exploration of advanced plasma operation modes.

The SCM system of JT-60SA comprises 18 Toroidal Field (TF) coils and 10 Poloidal Field (PF) coils. The PF coil system is further subdivided into four Central Solenoid (CS) coils and six Equilibrium Field (EF) coils, each serving distinct roles in plasma shaping and confinement.

The complete assembly of the JT-60SA device was finalized in late 2020 [48], marking the transition to the commissioning phase of the magnet system. This phase includes the validation of superconducting performance, power supply integration, and the implementation of diagnostic and protection systems essential for safe and efficient operation.

In accordance with the research activities carried out during my period at Fusion for Energy (F4E), the remainder of this chapter is structured as follows:

- Section 3.1 presents the PSIM-based modelling and simulation developed for both the Poloidal Field (PF) and Toroidal Field (TF) coil power supply systems, together with the validation of the designed tool.

- Section 3.2 describes the application of a Multilayer Perceptron (MLP) neural network to mitigate affected signals in the Toroidal Field quench detection system.

3.1 PSIM simulation of the control of the poloidal field coils power supply and toroidal field coils power supply

The objective of this activity is the modeling of the electronic and control systems associated with the JT-60SA coil power supplies (QST, Naka, Japan), utilizing the PSIM simulation environment. The development process adhered to the technical specifications and implementation guidelines provided by Jema Energy.

The simulation framework incorporates the digital control logic derived from the source code developed for the DSP-based control board, as programmed by the engineering team at Jema Energy.

The magnet power supply system consists of the power supplies for the superconducting and in-vessel normal conductor coils. DC power supplies for additional heating systems are to be involved as a part of their systems. Figure 3.1 shows a schematic diagram of the AC power supply system for JT-60SA at Naka Fusion Institute, from [49].

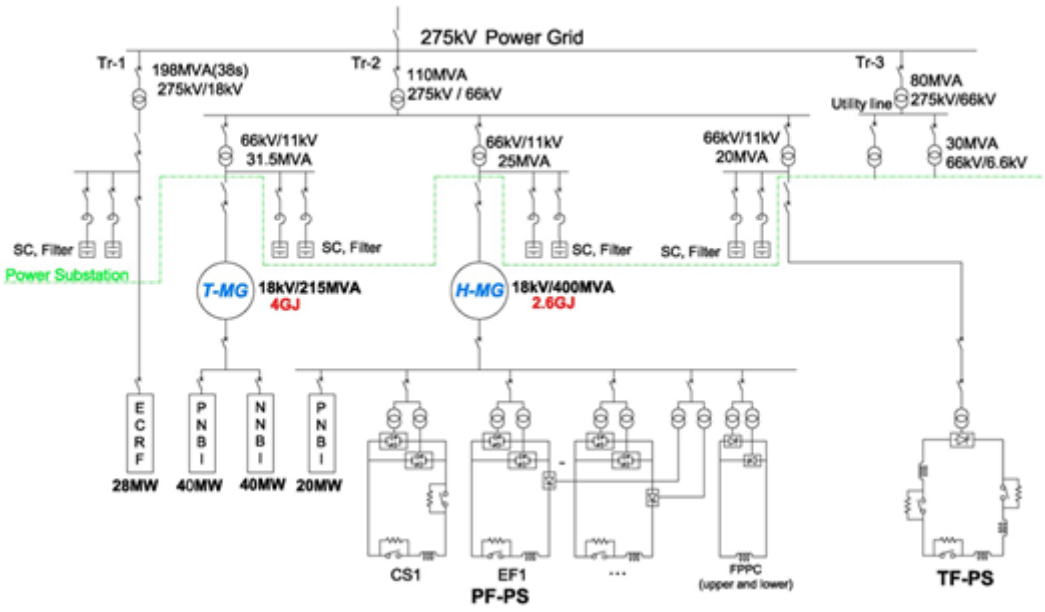

FIGURE 3.1: Schematic diagram of AC power supply for JT-60SA.

The paragraph will be organized as follows:

- the modeling and control strategy of the poloidal field coil power supply (PF PS) is presented first, as it represents the more complex configuration;
- and subsequently, the methodology is extended to the toroidal field coil power supply (TF PS), whose control architecture is a simplified adaptation of the PF PS system.

3.1.1 Poloidal field coil power supply

The poloidal field (PF) coil system of JT-60SA consists of four central solenoid (CS) units and six equilibrium field (EF) coil units, which collectively enable precise control of the plasma shape and vertical position of the centroid [50]. Consequently, ten independent magnet power supply units are required, one for each PF coil.

Due to the superconducting nature of the PF coils, which exhibit negligible electrical resistance, the magnet power supplies are capable of regulating coil current using low voltage during steady-state phases of plasma operation. To accommodate the varying operational demands, each magnet power supply is composed of two subsystems: a low-voltage supply, referred to as the "Base PS," and a high-voltage supply. The high-voltage supply is activated only during transient phases, such as plasma initiation and current ramp-up. During the remaining periods of plasma operation, the "Base PS" alone is responsible for current regulation, provided that no significant plasma perturbations are present.

Depending on the transformer configuration, the "Base PS" consists of either two or four thyristor converter units operating in parallel. Each thyristor converter is implemented as a back-to-back configuration, comprising six arms with anti-parallel connected thyristor devices. This arrangement enables full four-quadrant operation, allowing for bidirectional current and voltage control.

Figure 3.2 illustrates the detailed DC circuit configurations adopted for EF coil power supplies.

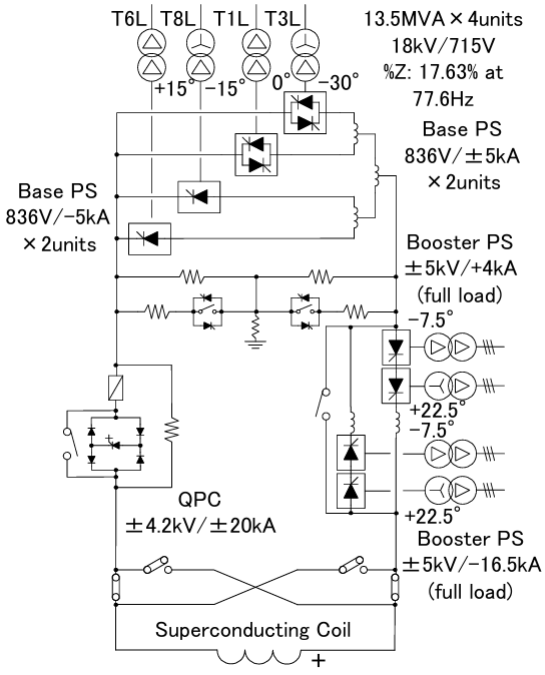

FIGURE 3.2: Circuital diagram of EF PS for JT-60SA.

The PF coil power supplies are interfaced with two pre-existing motor-generator units, designated as H-MG, with ratings of 215 MVA/4.0 GJ and 400 MVA/2.6 GJ, respectively, used to isolate the electrical loads from the electrical power supply line.

The poloidal field power supply (PF PS) architecture comprises four thyristor bridges configured in the forward and reverse directions, capable of generating rectified DC buses with 6, 12, or 24 pulses with bidirectional current flow. Practically, each bridge combines 1 forward and 1 reverse thyristor bridge, obtaining 4 independent thyristor bidirectional bridges. To achieve a maximum of 24-pulse rectification, a phase shift of 15° is introduced between the alternating current (AC) supply networks feeding the bridges. This configuration enhances the quality of the output waveform and reduces harmonic distortion in the DC bus.

The equilibrium field coil power supplies EF3 is powered by four pre-existing transformers, each rated at 13.5 MVA. The main electrical parameters of these transformers are summarized in Table 3.1. Due to the specific phase shift configuration between the secondary windings of the transformers, a cumulative phase displacement of 15° is achieved.

Nominal power	13.5MVA
Primary voltage	18kV
Secondary voltage	715V
Transformer winding group	EF3: Y/d, fed from T1L, phase shift (-30º). EF3: D/d, fed from T3L, phase shift (0º). EF3: Y/d, fed from T6L, phase shift (-15º). EF3: D/d, fed from T8L, Phase shift (+15º). EF4: Y/d, fed from T7L, phase shift (-30º). EF4: D/d, fed from T5L, phase shift (0º). EF4: Y/d, fed from T4L, phase shift (-15º). EF4: D/d, fed from T2L, Phase shift (+15º).
Impedance	17.63% @ 77.6Hz

Table 3.1: Main electrical parameters of EF3 and EF4 transformers.

Rectifier thyristors main data are reported in Table 3.2. Each thyristor bridge is formed by thyristor fuse protection, thyristor stack, and RC snubber circuit.

Each forward and reverse thyristor in the power supply configuration is equipped with an RC snubber protection circuit connected in parallel. This snubber network, composed of a resistor and capacitor in series, serves to limit the rate of voltage rise (dv/dt) during thyristor commutation events, thereby enhancing the reliability and longevity of the semiconductor devices. The selected component values for the snubber circuit are as follows:

- Snubber resistor: 22 Ω
- Snubber capacitor: 1 μF
- Average power dissipation in the snubber: 221 W

Thyristor type	Phase control thyristor. Disk type
Manufacturer	POSEICO
Model	AT940
Rated Voltage	2900V
Rated Current	4687A
Itsm	75000A
dV/dt	1000V/us.
di/dt	200A/us.
I2t	28.125x10^6 A2s / tp=10ms

Table 3.2: Rectifier thyristors main parameters.

The EF3 power supply (PS) is designed to operate under either voltage or current regulation modes, depending on a predefined reference signal. Based

on the current demanded by the load, the following operational scenarios are identified:

- Forward rectifier operations
- Reverse rectifier operations
- Combined forward and reverse rectifier operations (circulating mode)

These scenarios are determined according to the output current, which is governed by the voltage or current reference signal.

The reason why the rectifier cannot work only in the first two operating modes lies in the issue of current zero-crossing lag inherent in the four-parallel-bridge rectifier configuration, where the possible passage from four forward parallel bridges to four reverse parallel bridges could cause a delay in the change on polarity of the load current. In order to address this problem it is necessary the usage of a circulating current to maintain a constant current flux in the four bridges.

Let us give a look at Figure 3.3 that depicts the functional behavior of the bridges corresponding to each operational state and the relative state transitions, while Table 3.3 describes the respective threshold for each state transition. In State 3 and State 4 (respectively forward and reverse circulating operative modes) the current flows in the opposite way in the two couple of bridges. For instance, when the load current is positive (forward mode), bridges 1 and 2 pump a positive current while bridges 3 and 4 pump a negative one to ensure a minimum current of $I_{circ} = 1kA$ flows through them, and viceversa.

Transition	**Criteria**
A	$I_L > +6.0\,\text{kA}$
B	$I_L < +5.0\,\text{kA}$
C	$\Delta T_{trans} > 20\,\text{ms}$
D	$I_L < 1.0\,\text{kA}$
E	$\Delta T_{trans} > 20\,\text{ms}$
F	$I_L > +2.0\,\text{kA}$
G	$I_L < 0.0\,\text{kA}$
H	$I_L > 0.0\,\text{kA}$
I	$I_L > -1.0\,\text{kA}$
J	$I_L < -2.0\,\text{kA}$
K	$\Delta T_{trans} > 20\text{ms}$
L	$I_L > -6.0\,\text{kA}$
M	$\Delta T_{trans} > 20\text{ms}$
N	$I_L < -5.0\,\text{kA}$

TABLE 3.3: State transition criteria for the JT-60SA PF PS state machine.

After having explained the working principles of the power supply, let us have a quick overview on the control system organization. As shown in Figure 3.4, each power supply (PS) unit is equipped with a Local Control Cubicle (LCC), which serves as the interface for communication with the JT-60SA central control system. The PS is directly connected to the Power Supply Supervising Computer (PS SC), which is itself integrated with the

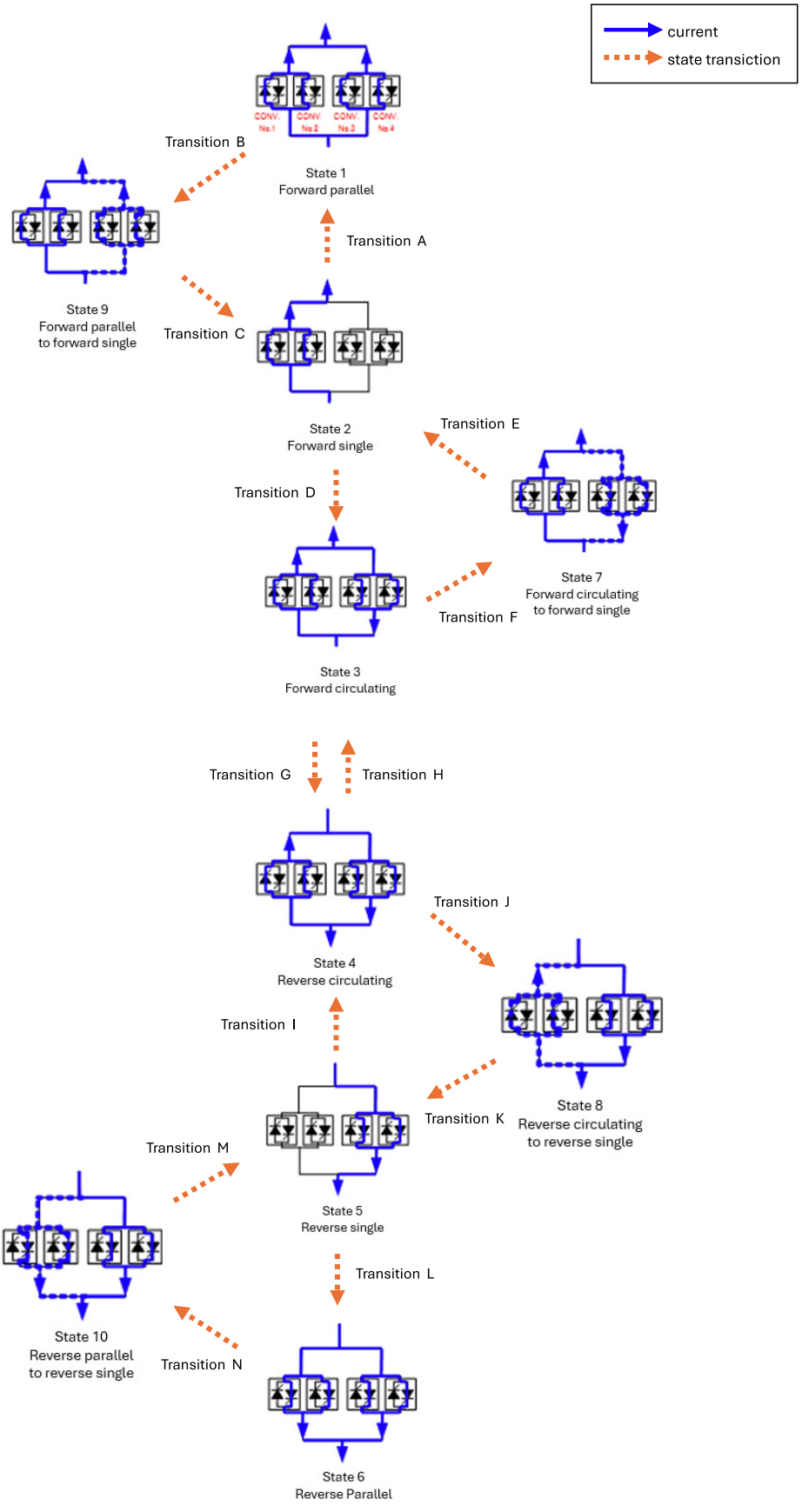

FIGURE 3.3: Operating mode for each state and state transition of the PF PS state machines.

Supervisory Control and Safety Data Acquisition System (SCSDAS). The PS SC comprises the following main functional components:

- Plant control computer
- Real-time controller
- Discharge control computer
- Local Internal Protection System (IPS)
- Local Human Safety Interlock System (SIS)

Communication between the PS LCC units and the PS SC is established via the Power Supply Internal Reflective Memory (RM) loop. The PS SC is further connected to the SCSDAS through the SCSDAS Main RM Loop. A general schematic of the PS interface with the JT-60SA control system is presented in the following figure.

The base PS configuration, as provided by Jema, illustrates the various types of signals and data exchanged with the PS SC. Specifically, the EF3/4 power supply is designed to operate under either voltage or current regulation modes, based on a reference signal. The selection of the operational mode is determined prior to activation and is fixed by the PS SC. This regulation is executed within the Main Control Rack (MC-R) using a Jema-developed digital signal processor (DSP), operating at a control frequency of 5 kHz.

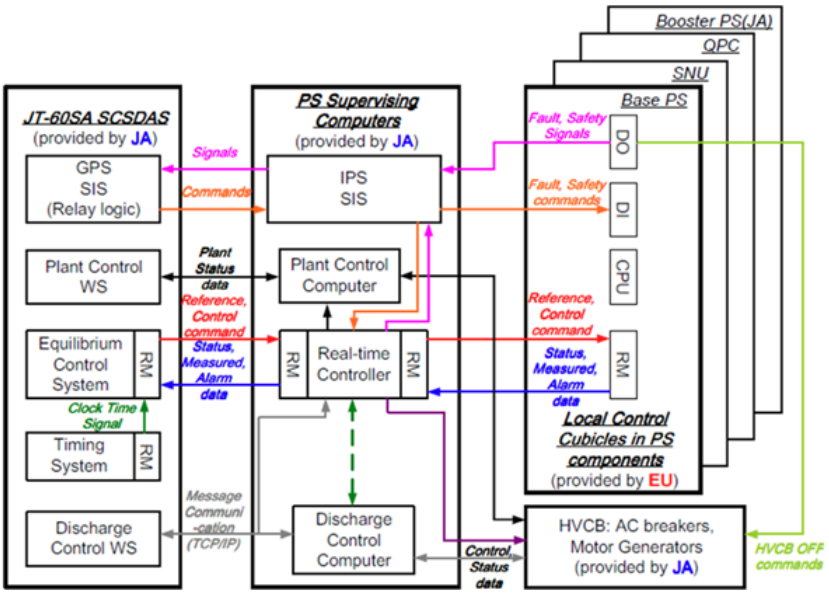

FIGURE 3.4: General schema of the PS interface with the JT-60SA control.

In Figure 3.5, it is shown the electric scheme of the PF PS that has been modeled using the simulation software PSIM.

The PF PS can operate under either voltage or current control modes. Figure 3.6 illustrates the closed-loop configuration employed for current regulation. The control architecture is based on a single current control branch, implemented through a proportional-integral (PI) controller. This controller generates a voltage reference signal, V_{ref}, which is distributed to four independent control branches, each corresponding to one of the thyristor bridges.

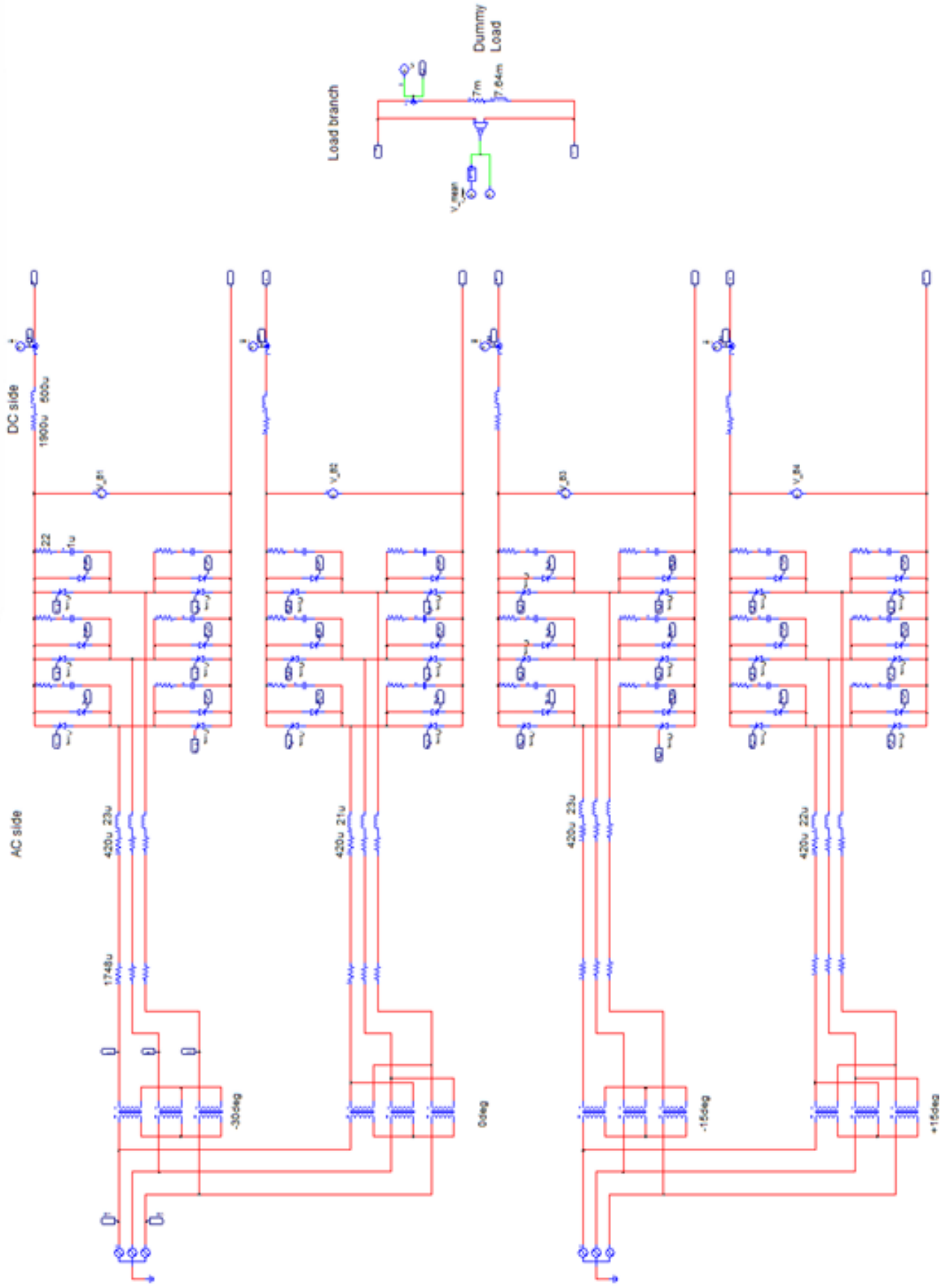

FIGURE 3.5: PSIM electric scheme of the PF PS.

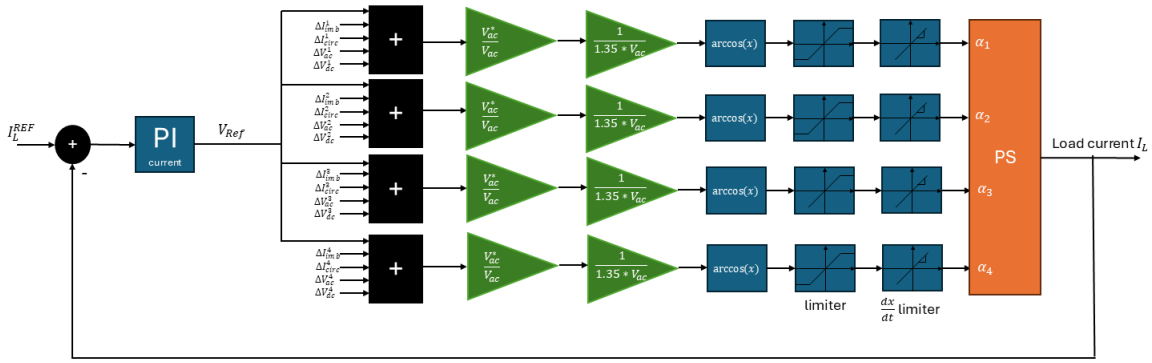

FIGURE 3.6: Current control closed-loop of the PF PS for JT-60SA.

In addition to the main voltage reference, four distinct voltage compensation terms are computed and added to V_{ref}:

- the current imbalance compensation, ΔI_{imb}. This compensation works so as to compensate the imbalance current between rectifiers in all operating modes. Depending if the PS is working in single or dual mode a different value is added or subtracted to each bridge. The relative block diagram is shown in Figure 3.7.
- the circulation current compensation, ΔI_{circ}. In the relative operational mode, the circulating current controller is implemented to regulate a circulating current of ± 1 kA for each forward and reverse thyristor bridge within the power supply. This control is achieved through a proportional-integral (PI) regulator that ensures stable and accurate current flow. The relative block diagram is shown in Figure 3.8.
- the AC-side drop voltage compensation, ΔV_{dc} and the DC-side drop voltage compensation, ΔV_{dc}. These compensations are the sum of thyristors voltage drop and the DC reactor voltage drop. This term concerns the voltage drop due to the AC inductance value. It is dependent on the grid frequency and DC current value. This two compensation are not implemented in this PSIM model control.

The resulting composite voltage reference ensures accurate and stable current regulation across all bridges. Following the computation of the composite voltage references for each rectifier bridge, an additional parameter is introduced to compensate for variations in the AC grid voltage. The formula that gives the α_i firing angles given the compensated voltage reference V^i_{Comp} is:

$$\alpha_i = \arccos\left(\frac{V^i_{Comp}}{1.35 * V_{ac}}\right) \tag{3.1}$$

where V_{ac} is the measured phase-to-phase rms voltage at the secondary of the transformer.

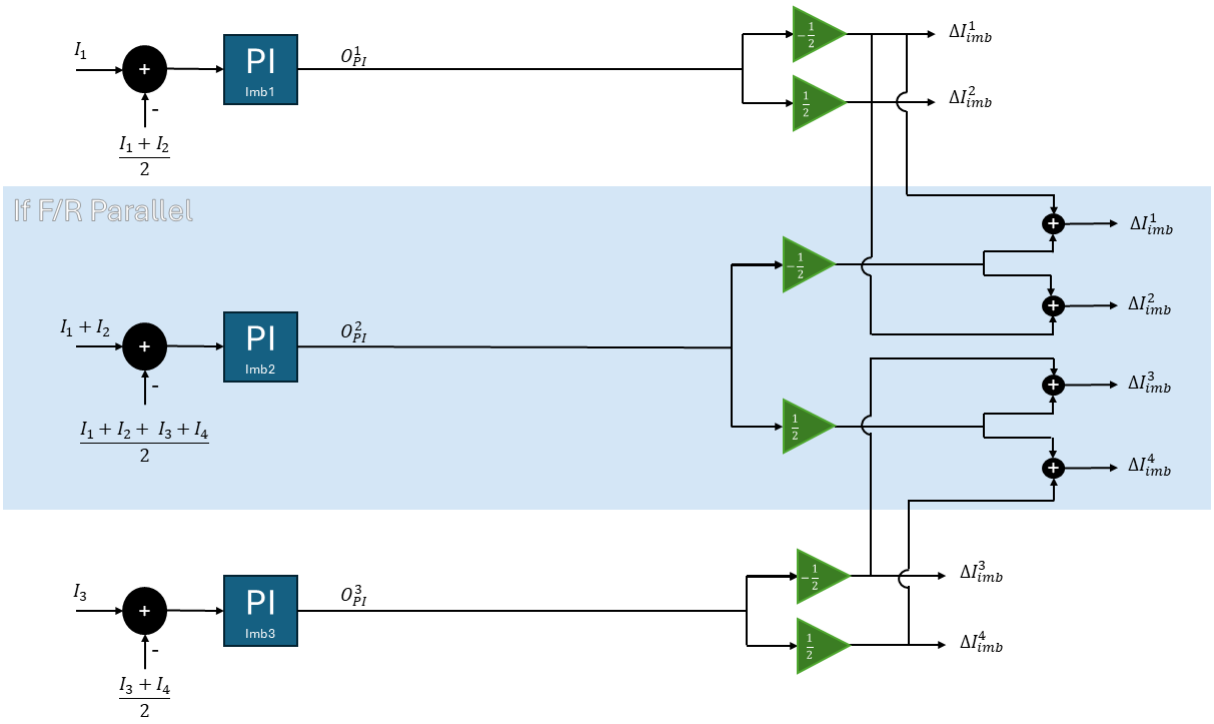

FIGURE 3.7: Current imbalance compensation block diagram.

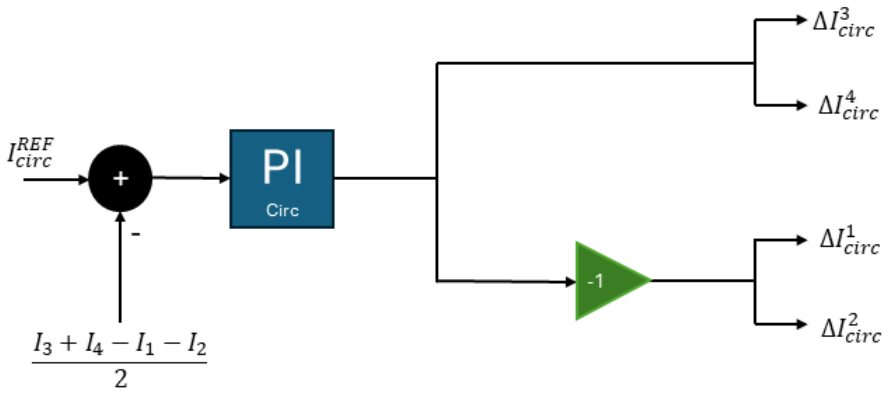

FIGURE 3.8: Circulating current compensation block diagram.

Once the alpha value is determined, it is processed through an angle limitation block that imposes two distinct constraints. The first constraint is a static angle limitation, which accounts for the grid frequency, grid voltage, and the output current. The second constraint is a dynamic angle rate limitation, which restricts the rate of change of the firing angle. This rate limitation is dependent on the grid frequency and is set to a maximum of 10,000°/s.

The result is the firing angle for thyristors in rectifier mode. In order to have the value for thyristors in inverter mode, the same firing angle have to be subtracted from 180° is applied to the reverse part of the rectifiers.

3.1.2 Toroidal field coil power supply

This power supply provides the required DC current continuously to Toroidal Field (TF) coils. The nominal DC voltage is enough to charge or discharge the full current in less than 25 minutes (assuming 11kV in HV network).The TF power supply circuital diagram is shown in Figure 3.10

The typical normal operation of the TF coil power supply is that the TF coils are energized every day in the morning, before the starting of the plasma experiments, and demagnetized after the end of the daily experiment period (see Figure 3.9).

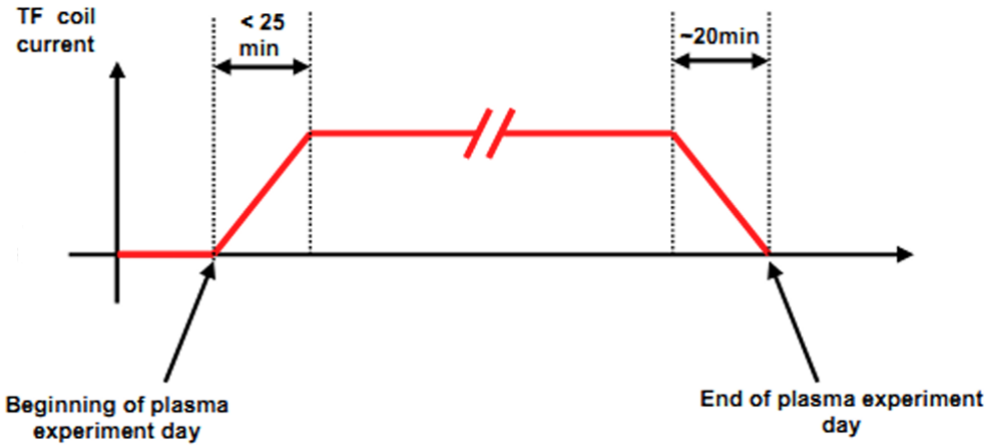

FIGURE 3.9: Typical operation of the TF coil power supply during an experiment day.

The TF power supply (TF PS) architecture comprises four parallel-connected thyristor bridges [51]. Each bridge consists of six water-cooled thyristors. These bridges are organized into two pairs, with each pair being fed by a dedicated three-phase dry-type transformer rated at 1.1 MVA and featuring a voltage specification of 11 kV/60 V RMS. The TF power supply circuital diagram is shown in Figure 3.10

To ensure system protection, a unidirectional crowbar circuit is installed at the converter output. In the event of a converter fault, the crowbar provides a low-impedance path for the superconducting coil current, effectively isolating the faulty converter. The crowbar assembly consists of a stack of thyristors connected in parallel with a mechanical switch to provide redundancy. Upon receiving a trigger signal, the crowbar is capable of short-circuiting the converter output within a few microseconds.

It is important to note that the crowbar is not designed to conduct the coil current under steady-state conditions. Therefore, its activation must be promptly followed by the engagement of the Quench Protection Circuit (QPC) to ensure safe current diversion and system integrity.

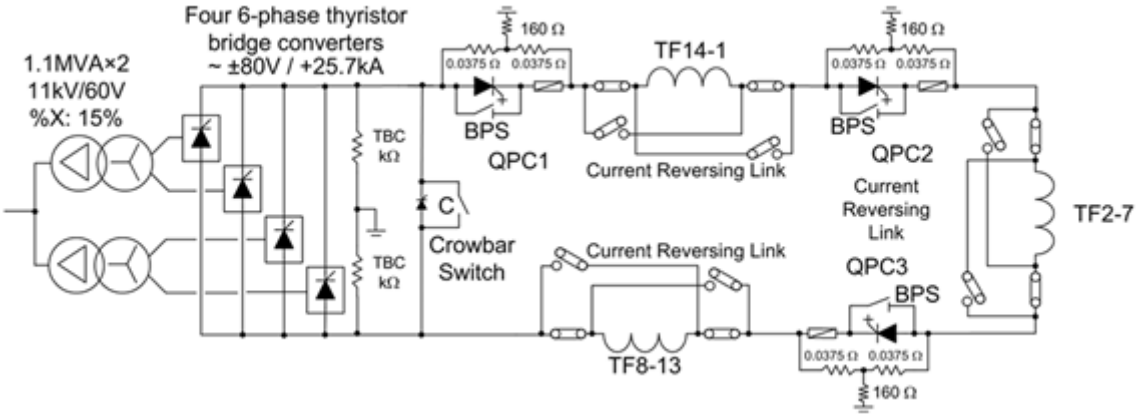

FIGURE 3.10: Circuital diagram of TF PS for JT-60SA.

Rectifier thyristors main data are reported in Table 3.4. Each thyristor bridge is formed by thyristor fuse protection, thyristor stack, and RC snubber circuit.

Each forward and reverse thyristor in the power supply configuration is equipped with an RC snubber protection circuit connected in parallel. This snubber network, composed of a resistor and capacitor in series, serves to limit the rate of voltage rise (dv/dt) during thyristor commutation events, as already said for the PF PS case. The selected component values for the snubber circuit are as follows:

Thyristor type	Phase control thyristor. Disk type
Manufacturer	POSEICO
Model	AT906
Rated Voltage	800V
Rated Current	5714A
Itsm	95KA
dV/dt	1000V/us.
di/dt	200A/us.
I2t	45.125x10^6 A2s / tp=10ms

TABLE 3.4: Rectifier thyristors main parameters.

Since this power supply works only in rectifier mode, there is not a state machine to manage different operational modes and, then, it works only in forward parallel mode. The control results simplified since it does not need a circulating current compensation as for the PF PS.

In Figure 3.11, it is shown the electric scheme of the PF PS that has been modeled using the simulation software PSIM.

3.1.3 PS control code implementation

As already said at the beginning of this chapter, the control of both the power supplies is implemented in a Digital Signals Processor (DSP) board, a specialized electronic circuit board designed to perform digital signal processing tasks. It houses a digital signal processor chip, which is a type of microprocessor optimized for fast, real-time processing of digital signals.

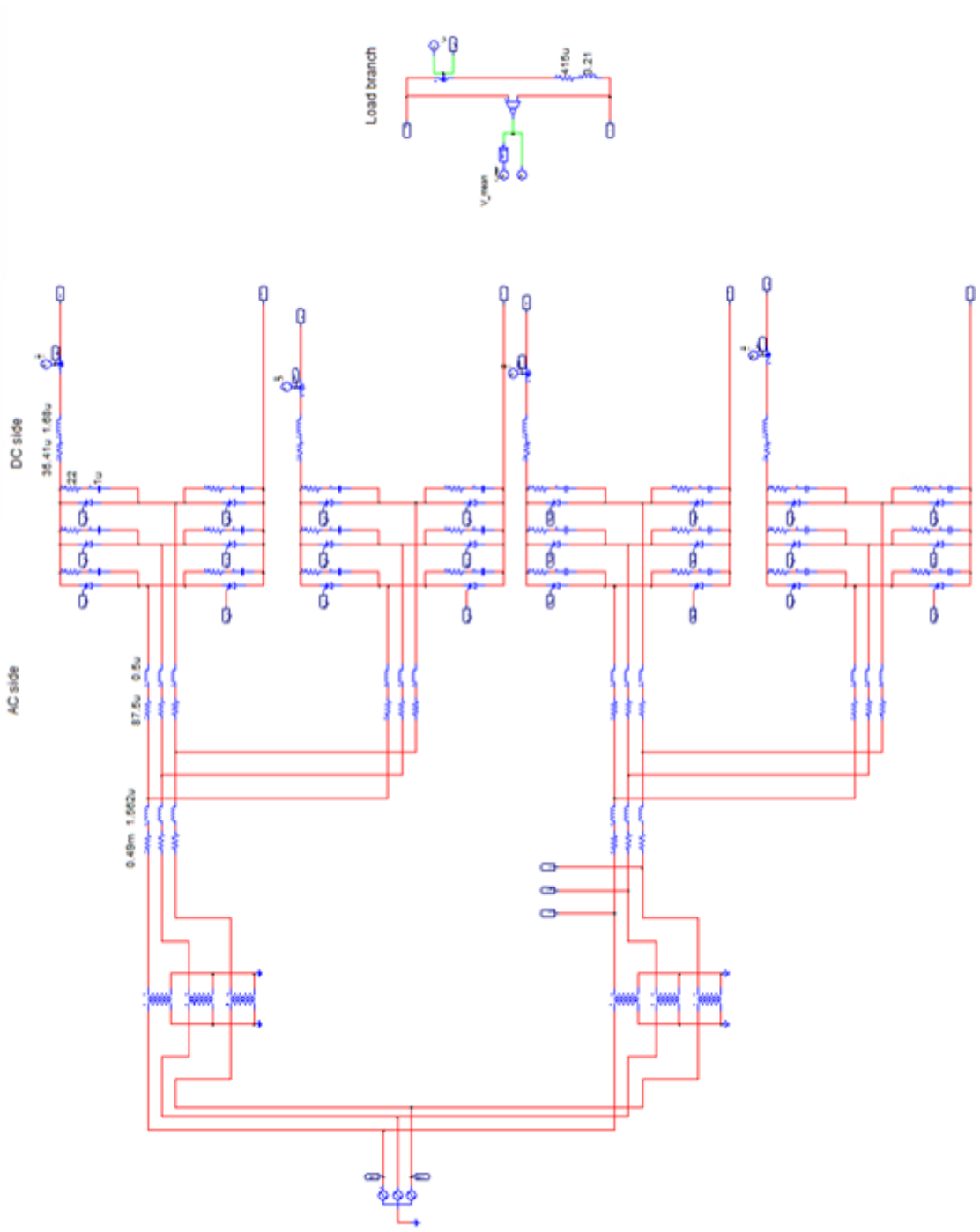

FIGURE 3.11: PSIM electric scheme of the TF PS.

The source code has been originally implemented by JEMA using C language in a BareMetal environment (without a OS sublayer) and describes the flow of control operations for both the PF PS and TF PS.

This code has been reorganized and optimized in order to work even in a constrained environment such as the PSIM C control block.

The key difference between the two different PS lies principally in the number of operative modes: the PF PS operates in nine different modes, whereas the TF PS functions only in forward parallel mode (or forward dual mode). This results in a significant simplification for TF control since it does not require the implementation of state machine. For this reason, we will focus on the PF PS control code, specifying only the cases where the TF PS control code coincides and highlighting any differences in parameters.

Initialization of the resources: SimulationBegin() function

This section describes the default setup function implemented within the PSIM Cblock. The subsequent subsections detail the individual functions invoked during initialization.

- **Initialize_st_PI():** This function initializes the discrete Proportional-Integral (PI) controllers.
- **Init_State():** This function sets the initial state of the system. It defines the starting conditions for various operational zones, such as forward, reverse, and circulating modes.
- **Init_Angles():** This function initializes the firing angles of the thyristors to zero. This precaution ensures that the thyristors remain non-conductive at startup, thereby enabling a controlled and safe system activation.

All the functions described above are utilized in the initialization procedures of both the Poloidal Field (PF) and Toroidal Field (TF) power supply systems.

Control Loop: SimulationStep() function

This function is executed repeatedly during the simulation to manage the control of the system. It performs a sequence of operations essential for the real-time regulation of the power supply. The main steps are described below:

- **Set Sampling Time:** The system defines the time interval for each control loop iteration. This sampling time is critical for the performance of the discrete PI controller algorithms implemented in the system.
- **Read_ADC():** This function acquires input signals from various sensors, including:
  - I_L: Load current
  - I_1: Bridge 1 DC current
  - I_2: Bridge 2 DC current
  - I_3: Bridge 3 DC current
  - I_4: Bridge 4 DC current

  - PI_I_out: Input from the current reference control loop, implemented as a separate control block
  - $VAC_{measured}$: Average of the RMS line-to-line voltages

- **Actualize_Current_Zone():** This function determines the current operating zone based on the measured currents and updates the system state accordingly. It evaluates whether the system should operate in forward parallel/single, reverse parallel/single, or circulating mode. Additionally, it detects state transitions that may trigger temporary operational modes. The PI controller parameters can be dynamically adjusted based on the active mode to optimize performance. For the TF power supply, this function is simplified, as it operates exclusively in forward parallel mode.

- **Calculate_Maximum_Firing_Angle():** This function computes the maximum permissible firing angle for the four bridges. The calculation considers both static parameters, such as the AC-side inductance, and dynamic parameters, including the load current and grid frequency.

- **Circulating_Current_Regulation():** This function governs the control of circulating current and current imbalance. Its behavior is state-dependent. In the case of the TF power supply, it is used solely for current imbalance control in forward parallel mode.

- **Calculate_Firing_Angles():** This function determines the firing angles of the thyristors required to regulate power flow. It converts voltage references into firing angles and applies slope adjustments to ensure smooth transitions, taking into account operational constraints.

- **Quantize_Angles():** The calculated firing angles are quantized to discrete values to emulate the behavior of the actual DSP discrete outputs. This quantization ensures high fidelity between the simulated and real-world control behavior. For instance, the values of the firing angles α_i of each bridges is normalized in a discrete interval of values between 0 and 4092 (12bit) that is then converted in the denormalized interval $[0°, 180°]$.

- **Write_Registers():** This function writes the final values of the firing angles and manages the enabling or disabling of the rectifier bridges. The primary outputs include:

  - `alphaOut1`, `alphaOut2`, `alphaOut3`, `alphaOut4`: the computed firing angles for each one of the four bridges.
  - `onForward12`, `onReverse12`, `onForward34`, `onReverse34`: these boolean values describes if each one of the two couple of bridges is enabled or disabled, and in which direction the current flows (forward or reverse).

3.1.4 Validation of the PSIM simulation models of the power supplies

In order to validate the two designed PSIM models, the waveforms obtained from the simulations have been compared with the waveforms recorded during

a factory test with a dummyload, since the tests of the new PS models with the actual PS coils still have not been performed and, then, it is not possible to have any comparison term. The dummy load consists of a RL load with $R_L = 100\,\Omega$ and $L_L = 0.07\,\mathrm{H}$.

PF PS validation

Due to the lack of real experimental data on the PF PS, there is currently no extensive dataset characterized by well-defined operational features. Consequently, the selected objective is to replicate a dummy load factory test scenario in which the current reference follows a piecewise linear bipolar triangular waveform. This waveform is defined as follows:

Time [s]	Current [kA]
1.000	0
1.250	20.0
1.750	-20.0
2.250	20.0
2.750	-20.0
3.000	0

In Fig. 3.12 shows the whole test waveforms achieved with the PSIM simulation model of the PF PS.

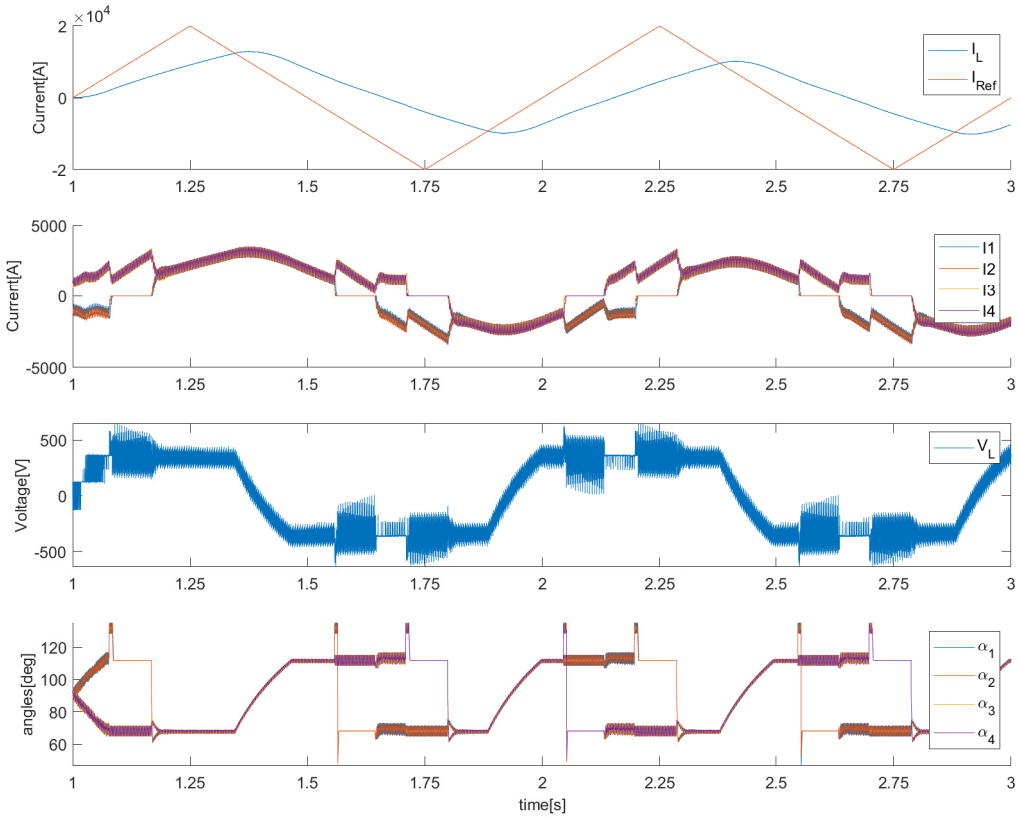

FIGURE 3.12: Waveforms from the simulation of the PF PS. The current reference signal is a bipolar triangular waveform.

In Figures 3.13 and 3.15 are reported two transitory phase: the first figure in which the current profile decreases crossing the zero, and the second one in which it increases crossing the zero. In these two frames, it is possible to observe all the PF PS operative modes starting from the reverse parallel, going to the forward parallel, and viceversa. Figures 3.14 and 3.16 show the real waveform for the currents of each bridge, the load current and the load voltage.

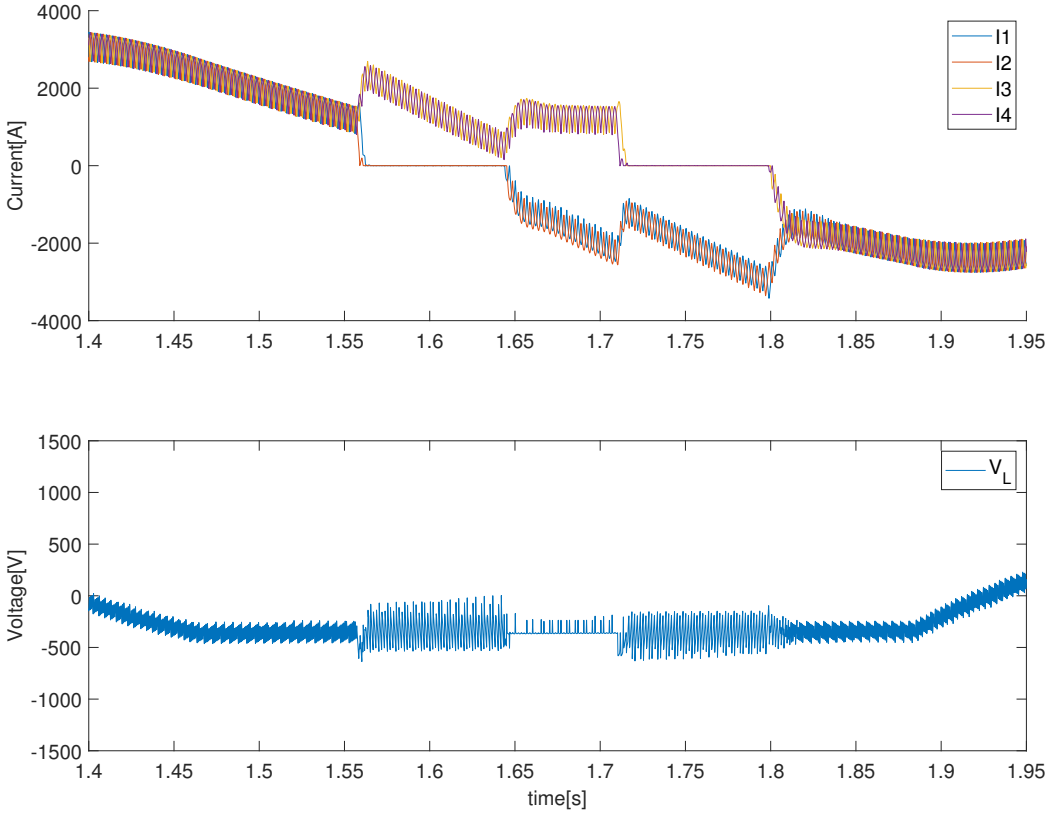

FIGURE 3.13: Zoom of the waveforms from the simulation of the PF PS with dummy load. The current profile decreases crossing the zero

TF PS validation

This waveform is defined as follows:

Time [s]	Current [kA]
0	0
1800	25.7
2200	25.7
4000	0

In Fig. 3.17 shows the whole test waveforms achieved with the PSIM simulation model of the PF PS.

In Figures 3.13 is reported the initial transitory phase in which the current profile increases starting from zero.

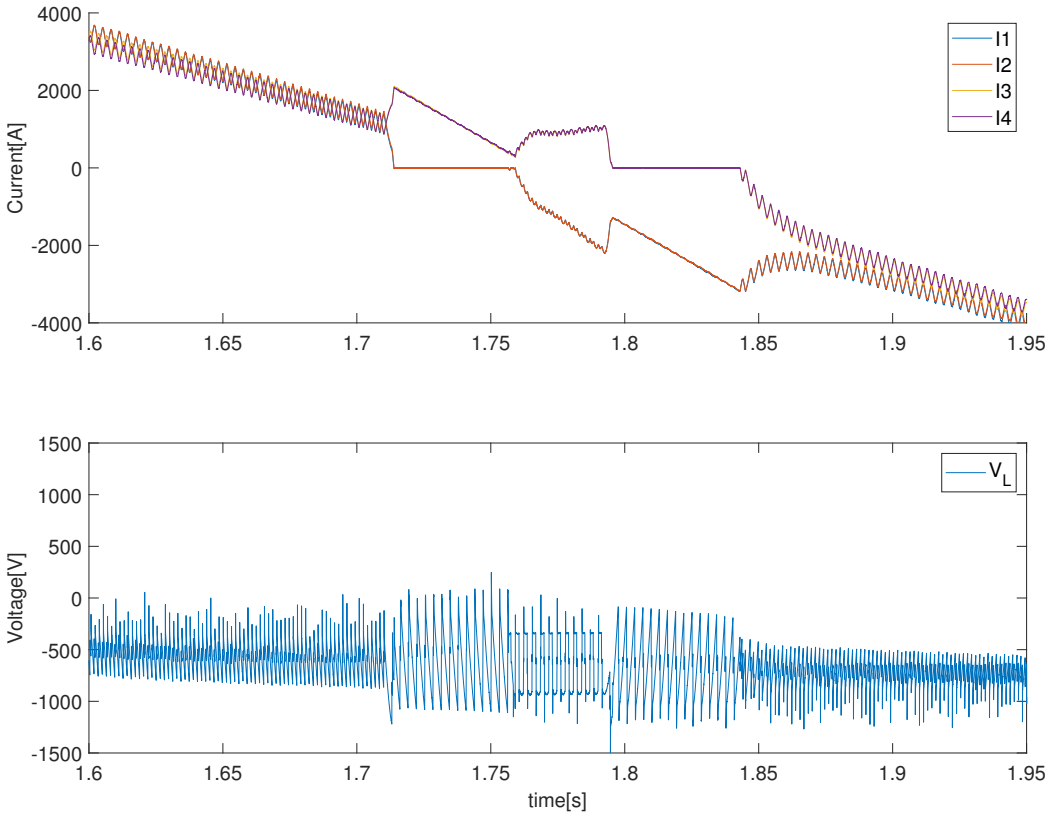

FIGURE 3.14: Real waveforms from the simulation of the PF PS with dummy load. The current profile decrease crossing the zero.

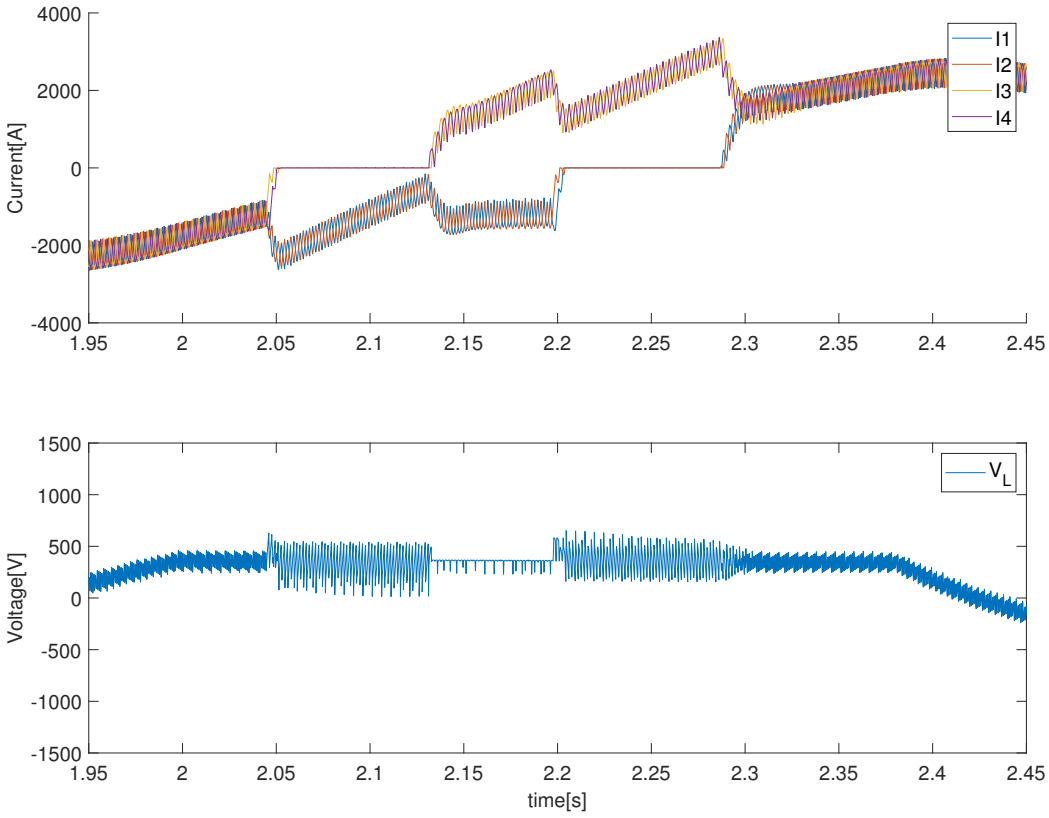

FIGURE 3.15: Zoom of the waveforms from the simulation of the PF PS with dummy load. The current profile increases crossing the zero.

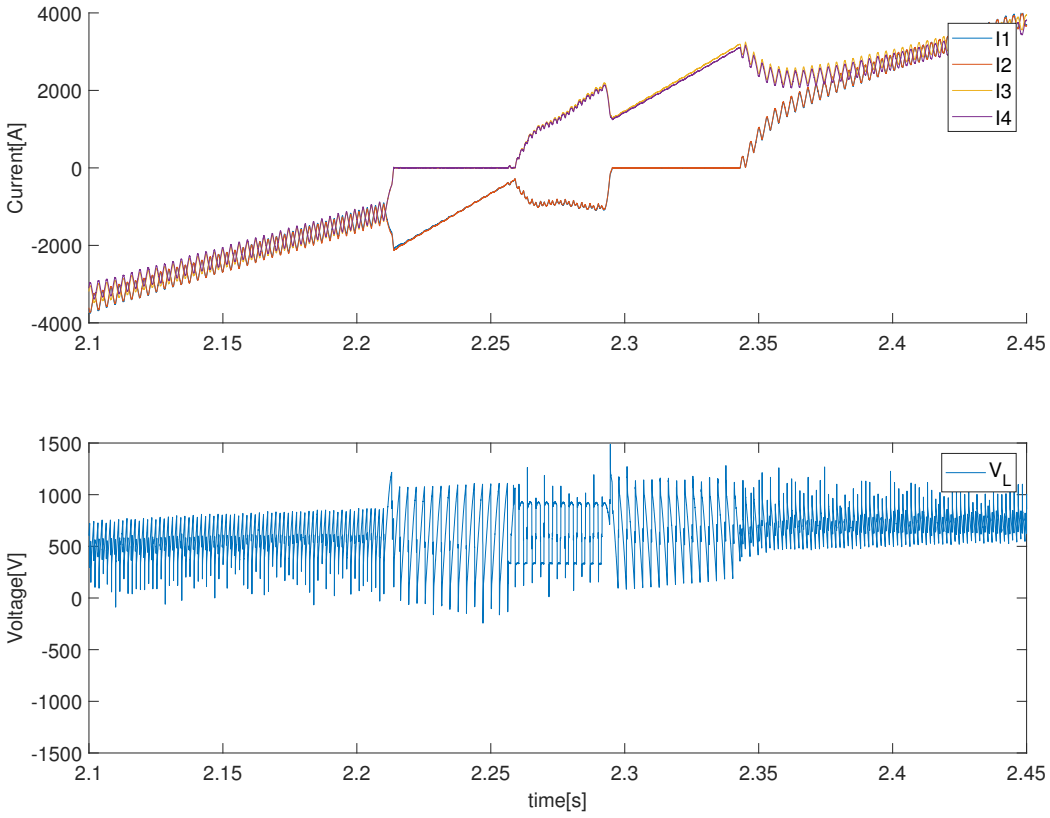

FIGURE 3.16: Real waveforms from the simulation of the PF PS with dummy load. The current reference signal is a bipolar triangular waveform.

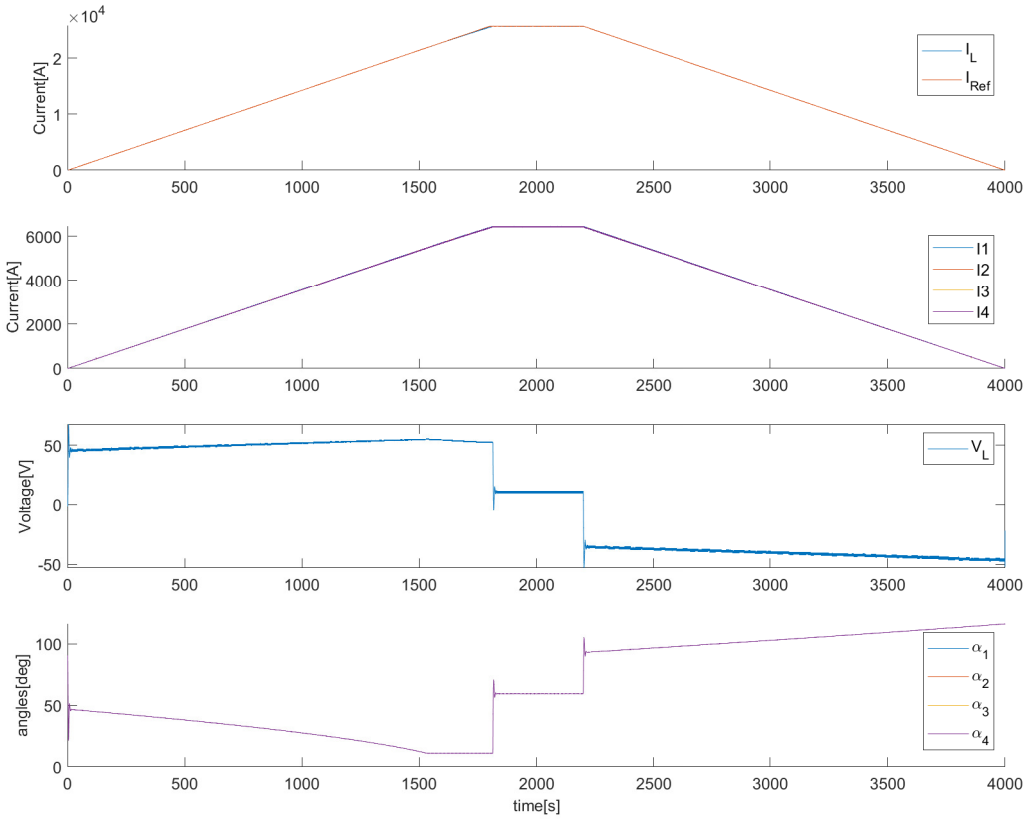

FIGURE 3.17: Waveforms from the simulation of the TF PS charge and discharge during an experimental session

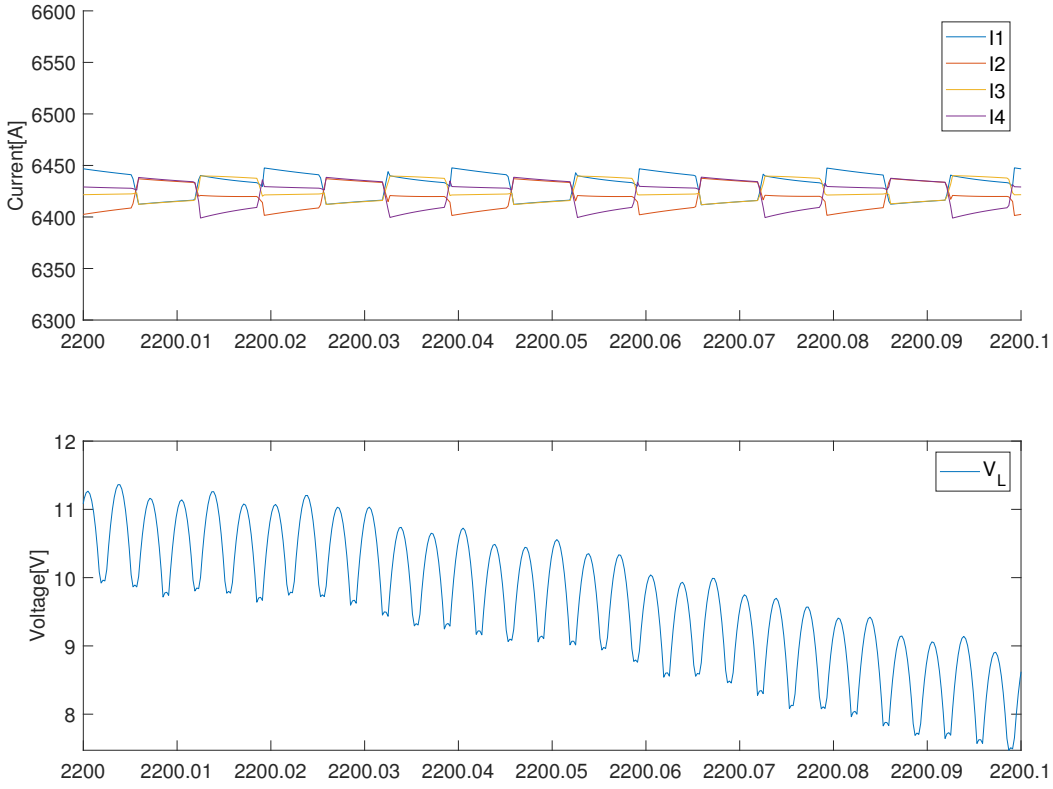

FIGURE 3.18: Zoom on the waveforms from the simulation of the TF PS charge and discharge during an experimental session.

3.2 Using multilayer perceptron to avoid false-positive quench signals in Toroidal Field quench detection voltage system

A quench in a superconducting magnet occurs when it transitions to a resistive state, producing voltage, heat, mechanical stress, and increased cryogenic pressure [52]. This phenomenon converts both the stored magnetic energy and the power supply input into thermal energy, which can range from partial to complete dissipation. Given the high energy density and current levels in such magnets, the risk of thermal damage is significant, making fast and reliable quench detection essential. Protection strategies must therefore be integrated into the magnet design to ensure system safety and performance. Upon detection, the power supply is shut down and the magnet is discharged, either internally or through an external dump resistor.

In JT-60SA, all superconducting magnets are equipped with Quench Detection Voltage (QDV) systems that monitor resistive voltage signals to detect the onset of a quench [53]. The Toroidal Field QDV (TFQDV) system uses a Wheatstone bridge to compare voltages between adjacent TF coils, effectively suppressing inductive components and isolating resistive ones.

During the 2023 Integrated Commissioning (IC) campaigns, the TFQDV system registered spurious signals during PF coil current ramps, despite being designed to reject such transients. These false positives posed a risk of unnecessary TF system discharges, potentially causing multi-day interruptions to operations.

An initial mitigation method applied a static linear mapping of PF coil voltages onto the integrated TFQDV signal used by the detection logic [54]. The linear nature of this model supports efficient implementation within the interlock system, allowing for real-time correction. While this method showed promising results across several independent test runs, its performance remains inconsistent, highlighting the need for more robust modeling approaches.

To address this, a data-driven approach using artificial neural networks (ANNs) was proposed. ANNs can model nonlinear dynamic systems effectively, improving the accuracy of quench signal prediction. Their ability to learn complex input-output relationships from data makes them well-suited for this task [26, 55], and they have been successfully applied in fusion-related modeling and control [56, 57].

3.2.1 A quick overview of JT-60SA quench detection system issue

The interlock system utilizes the integral of the TFQDV signal, computed over successive time periods of approximately one second, as the essential determinant for quench detection [54]. At times, spurious transients have caused this integral to approach the predetermined activation threshold, which is defined as:

$$\mathrm{Int}V(t) = \int_{t-t_0}^{t} V(u)\,\mathrm{d}u > 100\,\mathrm{mV}\cdot\mathrm{s}, \tag{3.2}$$

where Int$V$ represents the integrated TFQDV voltage sampled nearly every second, $V(t)$ is the instantaneous TFQDV measurement for a specific coil

pair, and t_0 is conventionally set to 1 s, or is shortened when the threshold condition is fulfilled.

Fig. 3.19 displays nine Int$V$ signals derived from shot E100738 of the IC database, which was captured during a plasma operation, alongside their corresponding ten PF current and voltage signals. A potential origin for these anomalous signals lies in unintentional magnetic coupling between the PF and TF coil assemblies. Ideally, the TF and PF coil winding planes should be orthogonal, thereby ensuring negligible mutual inductance. However, some magnetic flux can still couple into the TF coils. Regarding data acquisition, the nine TF QDV signals are sampled at a frequency of 100 Hz. The nine Int$V$ signals, representing the time-integrated TF QDV signals used by the detection logic, therefore preserve this sampling rate. Conversely, the PF coil voltage signals are initially captured at a higher rate of 4 kHz and are consequently stored in a separate database.

Stemming from these physical considerations, the initial mitigation strategy was grounded in linearly correlating the Int$V$ with a controllable variable that can be tracked and estimated promptly. In this context, the ten PF PS coil voltages, V_{PF}, were chosen as the pertinent variables.

This methodology yields a 9×10 transfer matrix T that correlates V_{PF} with the estimated Int$V(t)$. Crucially, employing a linear transfer matrix is also justified by its capacity for uncomplicated real-time implementation in the interlock system, enabling immediate correction features.

The application of this proposed method across several uncorrelated experimental runs has demonstrated certain positive outcomes. Nonetheless, its efficacy is not yet consistently reliable across every scenario examined.

3.2.2 A novel ANN-based approach

The goal of this work is to introduce an alternative scheme for mitigating the issue mentioned above by modeling the TF QDV–PF PS disturbance as a dynamic process, thereby eliminating the necessity for upstream parameter extraction routines that rely on dedicated experimental runs.

This aim is initially pursued utilizing a simple yet effective architecture, the multilayer perceptron (MLP) with a unipolar sigmoidal activation function and the gradient descent algorithm, which, as a universal function approximator, is capable of mapping any nonlinear dynamic behavior [58], and is furthermore employed in a wide array of similar disturbance mitigation strategies [59], [60].

Let us consider a multilayer perceptron composed of n_0 input units, l layers, and n neurons in total. The synaptic weights are represented by the matrix $\mathbf{W}$, structured as:

$$\mathbf{W} = \begin{pmatrix} \mathbf{W^1} & 0 & \dots & 0 \\ 0 & \mathbf{W^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{W^l} \end{pmatrix}$$

where $\mathbf{W^i}$ indicates the weight matrix associated with the ith layer.

The corresponding bias vector is:

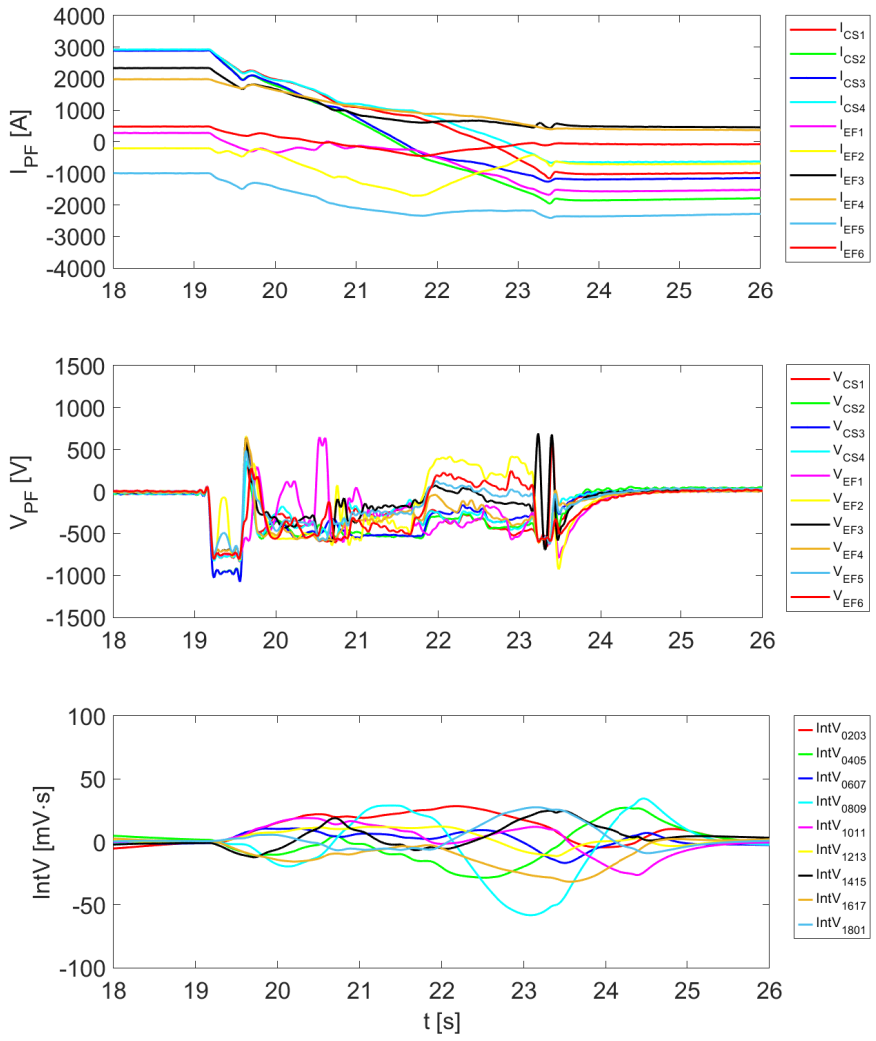

FIGURE 3.19: Signals from shot E100738 of the IC database. Top panel: ten PF current signals I_{PF}. Middle panel: ten PF voltage signals V_{PF}. Bottom panel: nine Int$V$ signals.

$$\mathbf{B} = \begin{pmatrix} \mathbf{B^1} \\ \mathbf{B^2} \\ \vdots \\ \mathbf{B^l} \end{pmatrix}$$

with $\mathbf{B^i}$ being the bias vector for the ith layer.

The output y_j of the jth neuron is computed by:

$$y_j = f\left[\sum_i (w_{ij} \cdot u_i) + b_j\right] \tag{3.3}$$

where u_i is the output from neuron i, w_{ij} is the synaptic weight from neuron i to neuron j, and b_j is the corresponding bias. The selected activation function $f(\cdot)$ is the standard sigmoidal function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

The training phase is carried out via the backpropagation algorithm, which propagates the error signal from the output layer to the input layer. Let us define the error for a given input/output pattern p as:

$$E_p = \frac{1}{2}\sum_i (t_{pi} - o_{pi})^2 \tag{3.5}$$

where t_{pi} and o_{pi} represent, respectively, the desired and actual outputs of the ith neuron for the pth pattern.

The total error over all P training patterns is:

$$E = \sum_{p=1}^{P} E_p \tag{3.6}$$

By performing gradient descent on E, the weights are updated according to the rule:

$$\Delta_p w_{ij} = \eta\, \delta_{pj}\, o_{pi} \tag{3.7}$$

where η is the learning rate, o_{pi} is the output of neuron i when pattern p is presented, and δ_{pj} is the local gradient (or delta) associated with neuron j.

To enhance convergence and reduce oscillations, a momentum term α is introduced, yielding the updated rule:

$$\Delta_p w_{ij}(k) = \eta\, \delta_{pj}(k)\, o_{pi}(k) + \alpha\, \Delta_p w_{ij}(k-1) \tag{3.8}$$

Finally, the weights are updated iteratively across epochs k according to:

$$w_{ij}(k) = w_{ij}(k-1) + \Delta_p w_{ij}(k) \tag{3.9}$$

This formulation enables the real-valued multilayer perceptron to minimize the total error $E(\mathbf{w})$ across training samples and converge toward an optimal set of weights and biases.

3.2.3 Preprocessing and choice of the signals to process

The input/output mapping structure utilized in the first mitigation method, where the voltage of the Poloidal Field coils, V_{PF}, is used to derive the integrated voltage signal, $\mathrm{Int}V(t)$, is partially retained, while discarding the premise that the system is static.

A preliminary, critical step involved the removal of noise and ripple components from the V_{PF} signals. Signals were first processed with a 100 Hz low-pass filter to suppress high-frequency ripples originating from the thyristor bridge rectifier commutations. Subsequently, the filtered V_{PF} signal was downsampled to 100 Hz to match the sampling rate of the $\mathrm{Int}V(t)$ signal, thus ensuring precise temporal alignment for ensuing data processing and analysis.

The codomain of the unipolar sigmoidal activation function spans the interval $[0, 1]$, which directly impacts the pre- and post-processing steps for the input and output data. Consequently, all input and target signals must be normalized to this range before being processed by the network and subsequently denormalized back to their original scale. The first step is therefore to establish the amplitude ranges of the raw signals to be adopted for the normalization and denormalization of the processed signals:

- The PF voltage, V_{PF}, spans $[-5.5\,\mathrm{kV},\ 5.5\,\mathrm{kV}]$, which corresponds to the maximum attainable output voltage when considering the contribution of a booster power supply [51].
- The integration signal, $\mathrm{Int}V$, is allocated a working range of $[-500\,\mathrm{mV}\cdot\mathrm{s},\ 500\,\mathrm{mV}\cdot\mathrm{s}]$.

Let us begin with the definition of the target as the 9 $\mathrm{Int}V(t)$ signals at the discrete time instant t:

$$\mathbb{T} = \left[\mathrm{Int}V(t)\right] = \begin{bmatrix} \mathrm{Int}V_{0203}(t) \\ \mathrm{Int}V_{0405}(t) \\ \vdots \\ \mathrm{Int}V_{0203}(t) \\ \mathrm{Int}V_{0203}(t) \end{bmatrix} \tag{3.10}$$

In order to select the model's input, a key observation concerns the system's dynamic behavior. Firstly, the value of $\mathrm{Int}V(t)$ is estimated over consecutive integration windows of approximately 1 s, based on the assumption that the system dynamics depend directly on the temporal history of the $\mathrm{Int}V(t)$ signal. At this initial phase, this system may be interpreted as a nonlinear finite impulse response (NFIR) **wang1991**, where the regressors of V_{PF} serve as the model inputs. Secondly, given the preceding assumption, it is vital to pinpoint the appropriate temporal regressors of V_{PF} for inclusion in the modeling process. A natural starting point arises from the fact that $\mathrm{Int}V(t)$ represents a fixed-window integral of the TF QDV signal over a 1-second interval. Consequently, this suggested employing 100 temporal regressors for each of the 10 PF PS voltages within the identical time window.

However, this initial choice was ultimately rejected, primarily due to the substantial volume of data required for processing at each prediction step, which presents a potential risk to achieving the real-time performance requirements of the method. A supplementary observation is that the dynamics of the $\mathrm{Int}V$ signal tend to persist for more than one second following the

conclusion of the V_{PF} excitation, suggesting a longer system response time compared to the excitation itself. For example, in shot E100941 (Fig. 3.20), the V_{PF} excitation ceases at approximately 24.5 s, whereas the Int$V$ dynamics continue until beyond 26 s. As a result, a reduced subset of Int$V$ regressors was chosen, while simultaneously expanding the time window of the delayed regressors to cover up to 2 seconds.

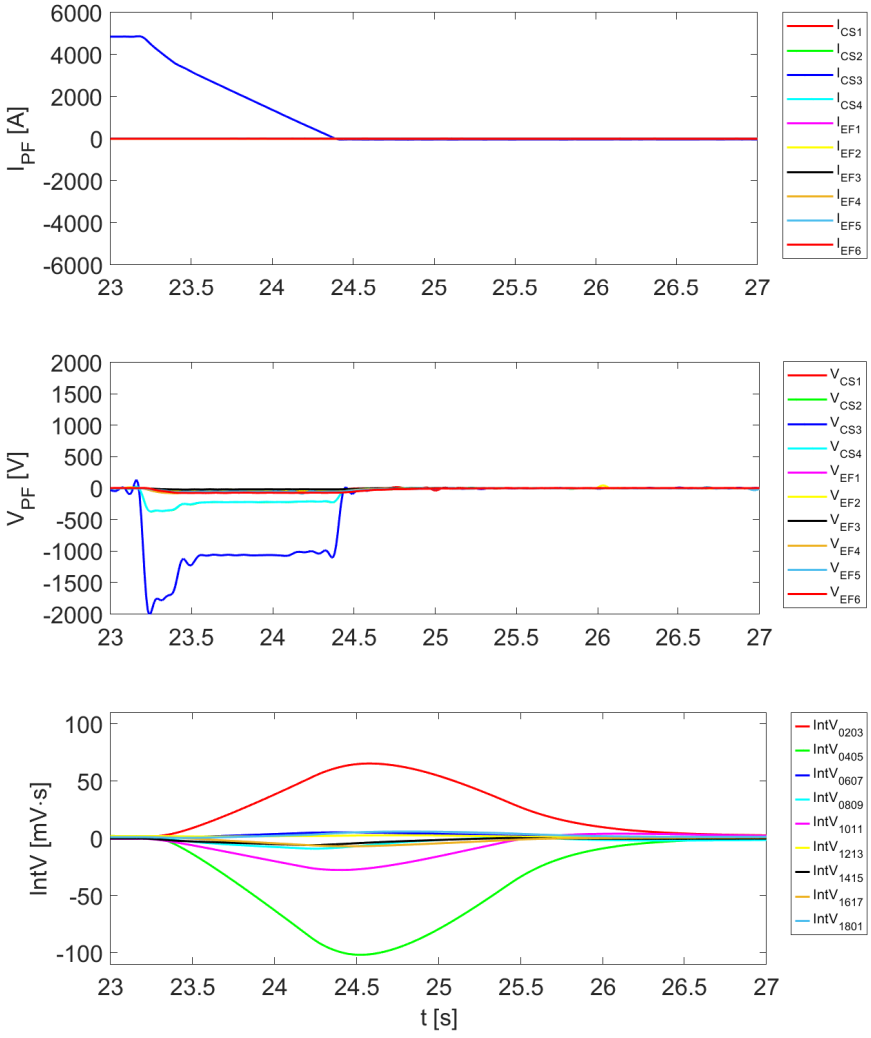

FIGURE 3.20: Signals from shot E100941 of the IC database. Top panel: ten PF current signals I_{PF}. Middle panel: ten PF voltage signals V_{PF}. Bottom panel: nine Int$V$ signals. The V_{PF} excitation ends at approximately 24.5 s, whereas the dynamics of Int$V$ persist until over 26 s.

In a first phase, a peak-to-peak correlation assessment between V_{PF} and Int$V$ signals was performed to ascertain the optimal number of regressors and their temporal spacing. Yet, this analysis failed to identify consistently shared features across all the shots evaluated. Consequently, the decision was made to proceed empirically.

After thorough evaluation, the optimal balance between model efficacy

and architectural simplicity was achieved by selecting 20 equally spaced time-delayed samples of V_{PF}, separated by intervals of 100 ms:

$$\mathbb{I} = \begin{bmatrix} V_{\mathrm{PF}}(t - 0.1\,\mathrm{s}) \\ V_{\mathrm{PF}}(t - 0.2\,\mathrm{s}) \\ \vdots \\ V_{\mathrm{PF}}(t - 2.0\,\mathrm{s}) \end{bmatrix} \tag{3.11}$$

where t is the same discrete time variable referenced in eq. 3.10 and:

$$V_{\mathrm{PF}}(t) = \begin{bmatrix} V_{\mathrm{CS1}}(t) \\ V_{\mathrm{CS2}}(t) \\ \vdots \\ V_{\mathrm{CS4}}(t) \\ V_{\mathrm{EF1}}(t) \\ V_{\mathrm{EF2}}(t) \\ \vdots \\ V_{\mathrm{EF6}}(t) \end{bmatrix} \tag{3.12}$$

Accordingly, the neural network's input layer at each iteration comprises 20 time-delayed regressors for each of the 10 PF voltage channels, yielding a total of $20 \times 10 = 200$ input neurons. The output layer consists of 9 neurons, each corresponding to one of the Int$V$ signals.

To validate the proposed ANN-based methodology, a segment of the dataset from the Integrated Commissioning (IC) benchmark was partitioned into two distinct subsets. In the training stage, the first subset was utilized to train the neural network and included shots characterized by varied PF coil energization current patterns. This heterogeneity was deliberately chosen to expose the network to a broad range of features and trends, thereby enhancing its capacity to generalize across different operational scenarios. Specifically:

- Figs. 3.19 and 3.20 exhibit representative examples of plasma scenario shots, where the PF coils are energized concurrently, and the coil currents I_{PF} do not display a specific trend. These scenarios closely mirror actual plasma operating conditions.
- Fig. 3.21 provides data from shot E100640, which features trapezoidal excitation patterns, as it is evident the coil currents I_{PF} follow a specific trapezoidal trend. In this instance, each PF coil is energized individually to isolate its respective contribution.

The summary scheme detailing the training process is presented in Fig. 3.22.

Once the training phase was finalized, a separate set of shots, distinct from those used for training, was employed to assess the neural network's generalization capability. This testing stage, subsequently termed the feedforward phase, sought to evaluate the trained model's proficiency in replicating the target behavior under operating conditions not previously encountered. During this phase, the ANN outputs were initially denormalized back to the original Int$V$ domain, corresponding to the range [−500 mV·s, 500 mV·s].

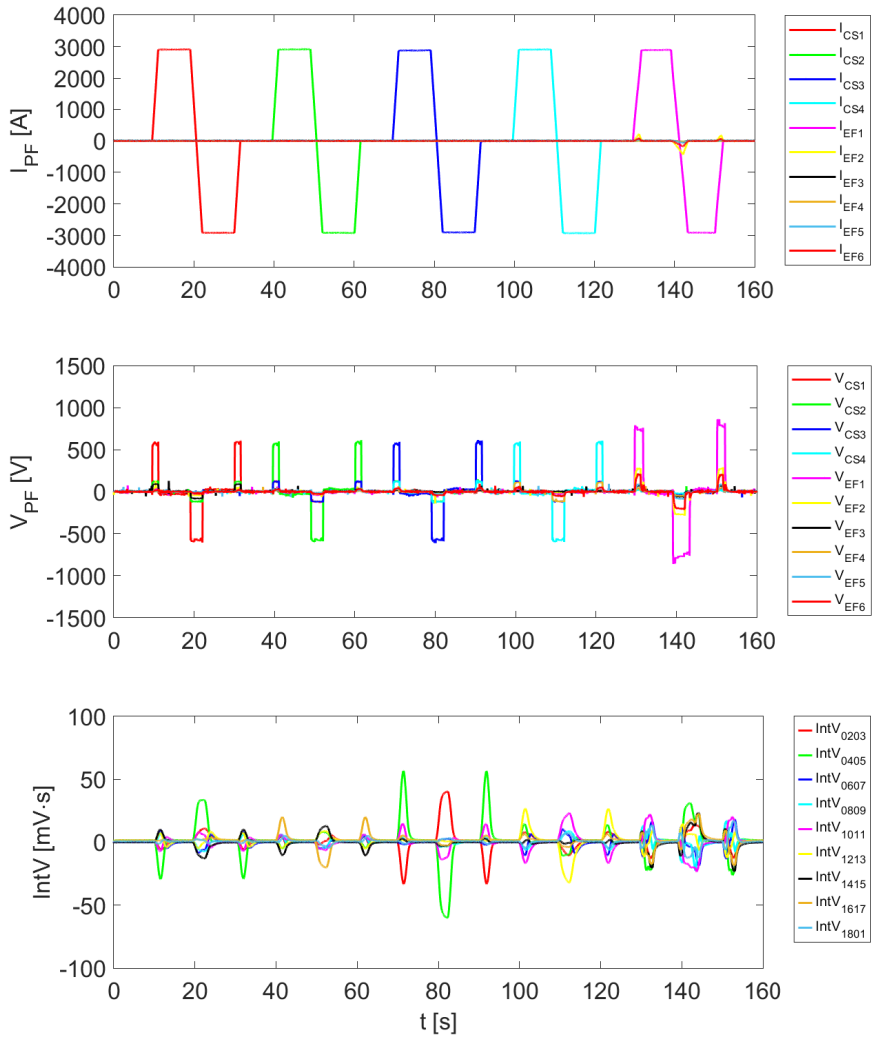

FIGURE 3.21: Signals from shot E100640 of the IC database. Top panel: ten PF current signals I_{PF}. Middle panel: ten PF voltage signals V_{PF}. Bottom panel: nine Int$V$ signals.

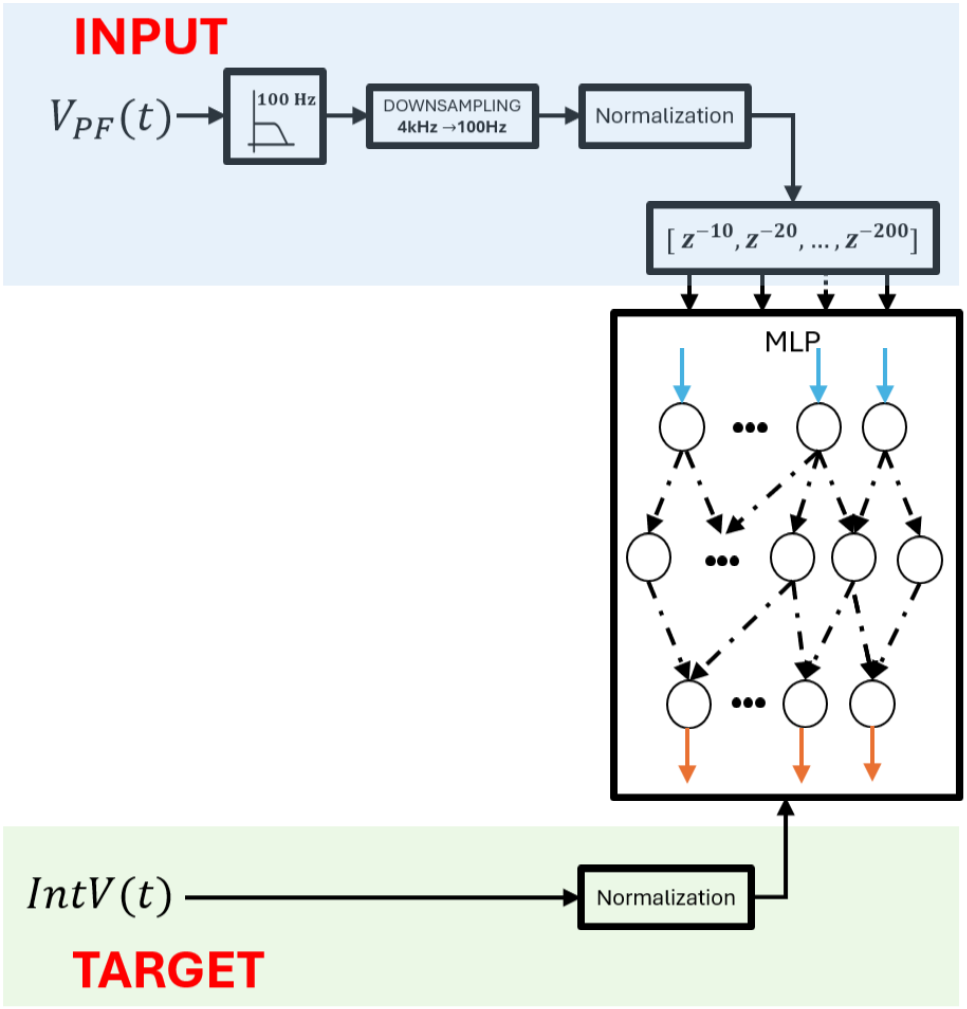

FIGURE 3.22: Resume scheme for the training phase. Inputs $V_{\mathrm{PF}}(t)$ are filtered at 10 Hz, downsampled to 100 Hz and normalized in interval $[0, 1]$. The block $[z^{-10}, z^{-20}, \ldots, z^{-200}]$ is a vector of delay operators that multiply the input in order to have the 20 previously indicated regressors. Outputs are filtered at 10 Hz and normalized in interval $[0, 1]$.

3.2.4 Performance index

To properly evaluate the efficacy achieved during the feedforward phase, a performance index has been introduced. Let us begin with the definition of the corrected signals $\mathrm{corrInt}V_j(t)$, computed as:

$$\mathrm{corrInt}V_j(t) = \widetilde{\mathrm{Int}V_j}(t) - \mathrm{Int}V_j(t) \tag{3.13}$$

where $\widetilde{\mathrm{Int}V_j}(t)$ represents the output of the neural network, thus being the ANN-predicted spurious component to be subsequently subtracted from the actual $\mathrm{Int}V_j(t)$ signal, with $j = 1, \ldots, 9$.

Next, we define the correction factor $C_{\%}$ as:

$$C_{\%} = 100 - \frac{|\mathrm{corrInt}V(t)|_{\max}}{|\mathrm{Int}V(t)|_{\max}} \times 100$$

where $|\mathrm{corrInt}V(t)|_{\max}$ is the maximum absolute value of the corrected signal, and $|\mathrm{Int}V(t)|_{\max}$ is the maximum absolute value of the original measured signal.

The scheme summarizing the feedforward and correction process is illustrated in Fig. 3.23.

3.2.5 Training parameters and Structure of the MLP

The training process was rigorously monitored to ensure adequate convergence. The number of training epochs, which denotes a single learning cycle in which all training patterns are presented to the neural network, was selected to minimize the error between the network outputs and their corresponding targets, while simultaneously preventing overfitting [61]. This phenomenon occurs when the network becomes overly specialized in reproducing the training data, capturing noise or specific patterns that fail to generalize effectively to unseen cases. In the present context, this would imply that the network simply memorizes the specific Int$V$ behavior of the limited training shots, rather than acquiring the underlying dynamical relationship between the inputs $\mathbb{I}$ and the targets $\mathbb{T}$, which is essential for accurately reproducing the original system's behavior.

Following a detailed analysis of performance trends and model hyperparameters, the number of training epochs was established at 40,000. This decision was supported by two main observations. First, after approximately 35,000 epochs, the average weight gradient decreased to the order of 10^{-8}, signifying almost complete convergence of the network weights. Consequently, the model's performance on the feedforward phase dataset remained essentially constant when the number of epochs was increased from 40,000 to 50,000, confirming that additional training offered no significant improvements. However, performance degradation on the testing dataset was observed beyond 50,000 epochs, indicating the beginning of overfitting.

Table 3.5 reports the variation in the order of the weight gradient and the mean correction factor $C\%$ across all testing phase datasets relative to the number of training epochs. To reduce the risk of overfitting and enhance training efficiency, parameters beyond the number of epochs were also carefully considered. These included the number of hidden neurons, n_h, which lacks a fixed rule in the literature. If n_h is too small, the network may lack

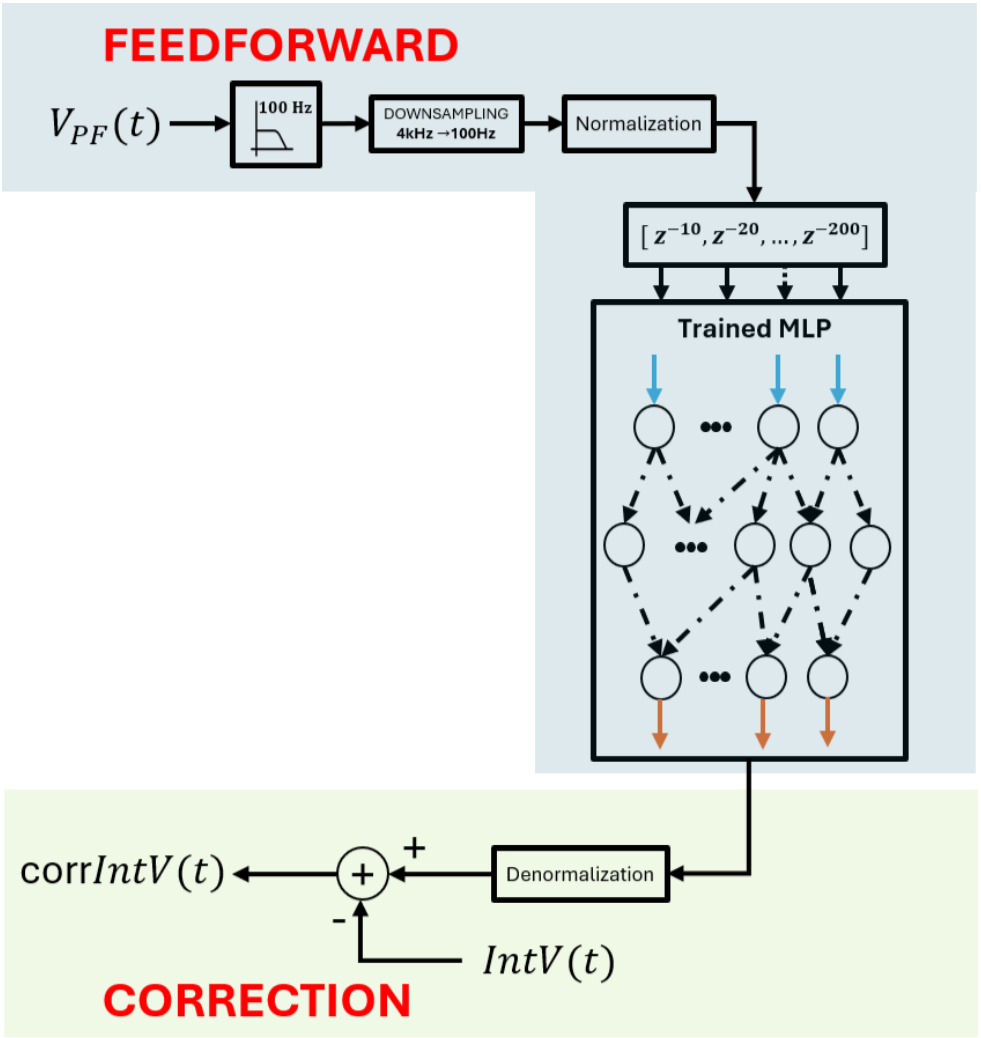

FIGURE 3.23: Resume scheme for the feedforward and correction phases. Inputs $V_{\mathrm{PF}}(t)$ are filtered at 10 Hz, downsampled to 100 Hz and normalized in interval $[0, 1]$. The block $[z^{-10}, z^{-20}, \ldots, z^{-200}]$ is a vector of delay operators that multiply the input in order to have the 20 previously indicated regressors. Outputs are then denormalized in interval $[0, 1]$ and used as corrector of the real measured disturbance.

the necessary capacity to learn from the available data; if excessively large, the risk of overfitting increases. After evaluating different configurations, n_h was chosen to be 75% of the number of input neurons n_i due to the optimal performance level achieved.

Furthermore, the learning rate η and the momentum coefficient α were also fine-tuned. While higher values for these parameters, typically around 0.5, can speed up the learning process, excessively high values can also induce persistent error oscillations and impede stable convergence.

Given the preceding considerations, the structure of the MLP and the training parameters were configured as follows:

- Number of input neurons: $n_i = 200$,
- Number of output neurons: $n_o = 9$,
- Number of hidden neurons: $n_h = 150$,
- Learning rate: $\eta = 0.01$,
- Momentum coefficient: $\alpha = 0.1$.

Epochs	$\log_{10}(\nabla W)$	$\boldsymbol{C_{\%}}$
10000	-4.1	45.8%
20000	-5.7	50.5%
30000	-7.9	53.4%
40000	-8.3	56.3%
50000	-8.3	56.1%
60000	-8.3	55.8%
70000	-8.3	55.7%

TABLE 3.5: Variation of the order of weight gradient and the mean correction factor $C_{\%}$ on the feedforward phase dataset with respect to the number of training epochs.

—

3.3 Feedforward phase results

Figures 3.24, 3.25 and 3.26 present a comparison between the measured signals $\mathrm{Int}V_j(t)$, denoted by 'o' markers, and the corresponding feedforward phase results $\widetilde{\mathrm{Int}V_j}(t)$ generated by the network, indicated with bold lines, for various unseen shots. The lower panels of each figure show the corrected signals $\mathrm{corrInt}V_j(t)$, computed as defined in Eq. (3.13). Furthermore, to highlight the achieved correction capability, each figure panel includes straight gray lines indicating the absolute maximum values of the peaks. These lines emphasize the actual range of the measured signals $\mathrm{Int}V(t)$ and the corresponding range of the corrected signals $\widetilde{\mathrm{Int}V_j}(t)$.

As evident in Figures 3.24 and 3.25, the proposed methodology substantially reduces the peaks of the $\mathrm{Int}V(t)$ signals. Notably, shots exhibiting high peak values are predicted with superior relative accuracy. For example, in scenario E100940, Fig. 3.24, the correction factor achieves $C_{\%} = 85\%$, signaling a significant reduction of the original peak amplitude. This outcome

underscores the network's ability to effectively mitigate the Int$V(t)$ signal, even in critical regions. In the simpler case of the trapezoidal test E100641, Fig. 3.25, the correction factor reaches $C_{\%} = 87\%$.

Considering Table 3.6, which reports the correction factors over a wider set of shots, it is observable that the correction task is more challenging for shots E100655, E100740, and E100874, with correction factors $C_{\%}$ around 60%. Although these values are lower than those observed in the cases previously analyzed, they still represent a noteworthy improvement over the initial condition.

The poorest outcome among all tested shots is represented by E100734, where the improvement only reaches $C_{\%} = 10\%$. A distinguishing characteristic of this shot is its relatively low peak amplitude compared to the others, as illustrated in Fig. 3.26. This limited improvement partially supports the previous finding that the network tends to be more proficient at identifying and correcting high-amplitude peaks, as demonstrated in the case of E100940.

There remains substantial scope for refinement of the presented methodology:

- Specific attention should be directed towards particular scenario shots like E100734. Although their peaks are less critical, remaining well below the 100 mV·s threshold, exploring strategies for further mitigating these residual peaks is still important.
- Only a restricted number of case studies have been investigated thus far, highlighting the need for broader validation across a more diverse range of scenarios.
- It has been noted that, for certain shots, the optimal correction factor is attained at a different number of training epochs compared to others, suggesting variability in convergence behavior.
- The number of hidden neurons could potentially be reduced to decrease the computational complexity of the algorithm, while sustaining comparable performance.

Although this constitutes an initial step towards applying ANNs to the problem under consideration, the proposed machine learning approach has yielded results that surpassed expectations. The MLP proved to be a viable candidate for tackling the mitigation task, delivering highly encouraging outcomes. The model demonstrated a remarkable capability to generalize across diverse PF coil energization strategies, confirmed by its robust performance on previously unseen experimental shots during the feedforward phase. Moreover, as detailed in Table 3.6, the ANN-based approach successfully lowered the amplitude of spurious Int$V$ signals in the majority of cases, achieving correction factors exceeding 50% in most evaluated shots.

Although the generalization capability of the ANN-based approach is currently limited by the relatively small number of case studies, highlighting the need for validation across a wider range of operational conditions, the method has proven effective in modeling and subtracting the observed disturbances. This confirms that the underlying mechanism is not due to random noise or hardware-induced artifacts, but rather to a deterministic response of the system to the PF voltage patterns.

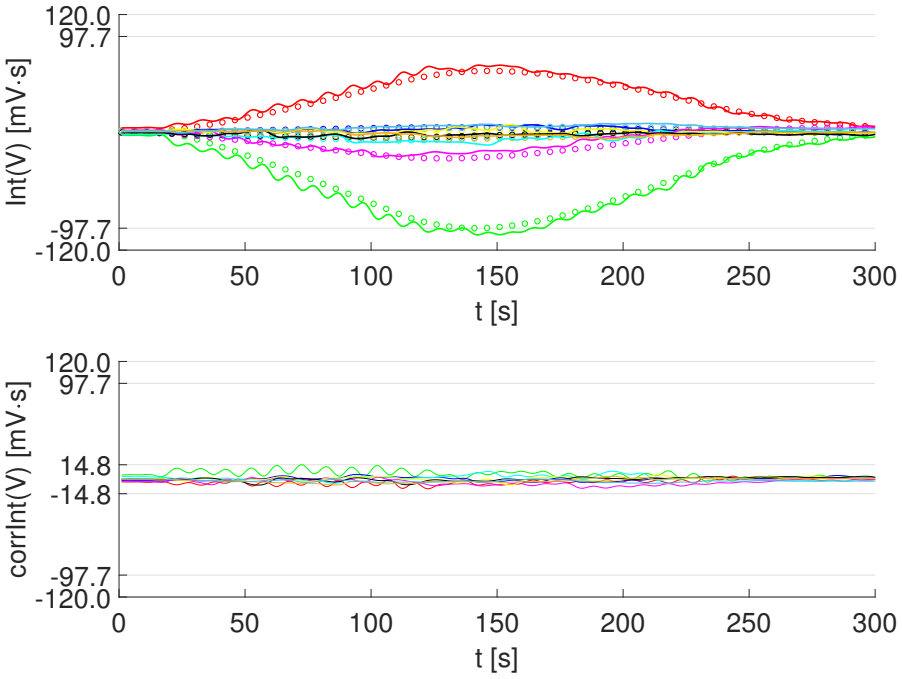

FIGURE 3.24: Top: comparison between actual Int$V$ signals ("o" markeded lines) and predicted simInt$V$ signals (bold lines) for shot E100940 of the IC database; Bottom: corresponding corrected corrInt$V$ signals.

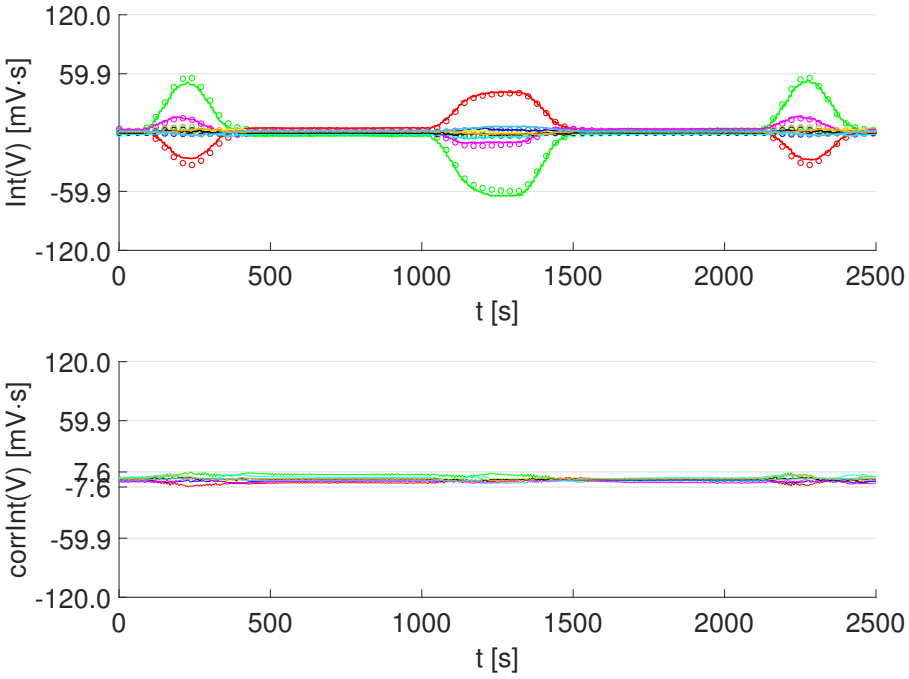

FIGURE 3.25: Top: comparison between actual Int$V$ signals ("o" markeded lines) and predicted simInt$V$ signals (bold lines) for shot E100641 of the IC database; Bottom: corresponding corrected corrInt$V$ signals.

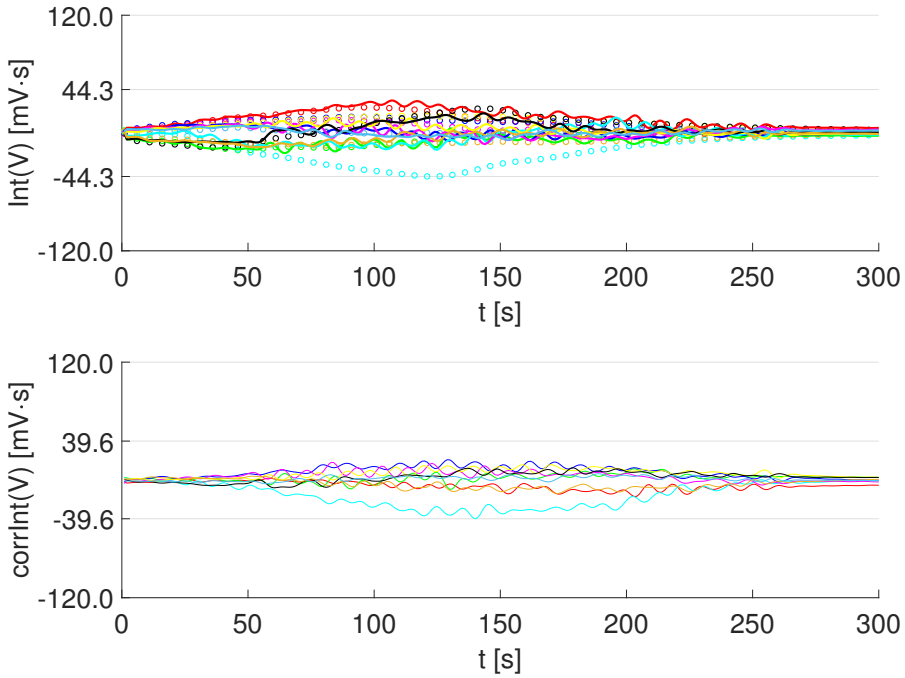

FIGURE 3.26: Top: comparison between actual Int$V$ signals ("o" markeded lines) and predicted simInt$V$ signals (bold lines) for shot E100734 of the IC database; Bottom: corresponding corrected corrInt$V$ signals.

Shot	$C_{\%}$
E100641	87%
E100655	60%
E100734	10%
E100737	12%
E100740	61%
E100874	58%
E100875	75%
E100940	85%

TABLE 3.6: Correction factors reached for different shots using the ANN approach.

These findings strongly support the hypothesis that spurious TFQDV signals can be accurately predicted and mitigated using data-driven techniques that exploit temporal dependencies. In particular, the experimental results demonstrate the ANN's ability to capture the system's nonlinear dynamics.

Overall, the results confirm the potential of ANN-based strategies as scalable and effective solutions for real-time disturbance mitigation in fusion devices.

Chapter 4

Hypercomplex multilayer perceptrons and Learning-on-Learning

Hypercomplex neural networks are crucial tools for processing high-dimensional data in various engineering and scientific applications. Since quaternion product is a noncommutative extension of complex domain product [62], hypercomplex neural networks can process multidimensional data more compactly and efficiently with a reduced risk to converge to local minima with respect to the real-valued counterpart.

Despite this advantages, their widespread adoption has been hindered by computational challenges, particularly in training and optimization processes [63]. Traditional optimization techniques, or specifically learning algorithms, are often not directly applicable to quaternion-valued networks, necessitating the development of specialized algorithms and methodologies.

In literature, different quaternion learning algorithms have been proposed to train hypercomplex neural networks effectively. These algorithms enable the training of neural networks while preserving the relationships between features through the utilization of quaternion algebra [64] (Appendix A). To address the computational challenges associated with Quaternion-valued Neural Networks, novel algorithmic approaches have been explored to reduce arithmetic complexity and accelerate computations. In order to minimize computational complexity, hypercomplex linear convolution algorithms have been developed, utilizing matrix-vector products and advantageous decompositions [65].

In this chapter, Hypercomplex Multilayer Perceptrons (HMLPs) are discussed and novel learning and optimization techniques are introduced. Two different learning algorithms are revisited:

- a first order method, the quaternion-valued gradient descend (GD) algorithm,

- and, a second order method, the quaternion-valued Levenberg-Marquardt (LML) algorithm, proposed in [66].

Moreover, the learning-on-learning strategy introduced in [67] has demonstrated significant improvements in the training phase of real-valued MLPs. In particular, the use of auxiliary networks to predict weights has been shown to enhance both the efficiency of the training process and the convergence properties of the models. This methodology can be systematically extended to HMLPs in order to address the stringent requirements of real-time learning for the image processing in plasma fusion plants. A relevant application is the detection of hot spots in thermal images acquired by infrared thermocameras monitoring the internal surfaces of reactors [68], application that can quickly prevent the occurrence of significant demage in the Tokamak wall.

4.1 Hypercomplex multilayer perceptrons: architecture and routines

Let n_o, n_h, and n_i be the number of output, hidden, and input neurons, respectively. The notation convention used is:

- Superscript O refers to quantities associated with the output layer;
- Superscript H refers to quantities related to the hidden layer;
- Superscript I refers to quantities concerning the input layer.

Weights and biases are defined as follows:

- $\mathbf{W}_{nm}^{HO}$ denotes the quaternion weight linking the mth hidden neuron to the nth output neuron:

$$\mathbf{W}_{nm}^{HO} = \mathbf{w}_{0nm}^{HO} + \imath\, \mathbf{w}_{1nm}^{HO} + \jmath\, \mathbf{w}_{2nm}^{HO} + \kappa\, \mathbf{w}_{3nm}^{HO} \tag{4.1}$$

- $\mathbf{W}_{nm}^{IH}$ represents the weight connecting the mth input neuron to the nth hidden neuron:

$$\mathbf{W}_{nm}^{IH} = \mathbf{w}_{0nm}^{IH} + \imath\, \mathbf{w}_{1nm}^{IH} + \jmath\, \mathbf{w}_{2nm}^{IH} + \kappa\, \mathbf{w}_{3nm}^{IH} \tag{4.2}$$

- $\mathbf{\Theta}_{n}^{H}$ indicates the bias term of the nth hidden neuron:

$$\mathbf{\Theta}_{n}^{H} = \theta_{0n}^{H} + \imath\, \theta_{1n}^{H} + \jmath\, \theta_{2n}^{H} + \kappa\, \theta_{3n}^{H} \tag{4.3}$$

The selected activation function is a quaternion-extended sigmoid function, defined as:

$$\mathbf{X}_n = \boldsymbol{\sigma}(\mathbf{S}_n) = \sigma(S_{0n}) + \imath\, \sigma(S_{1n}) + \jmath\, \sigma(S_{2n}) + \kappa\, \sigma(S_{3n}) \tag{4.4}$$

where:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.5}$$

is the standard real-valued sigmoid. Its derivative is expressed as:

$$\dot{\boldsymbol{\sigma}}(\cdot) = \dot{\sigma}(\cdot) + \imath\,\dot{\sigma}(\cdot) + \jmath\,\dot{\sigma}(\cdot) + \kappa\,\dot{\sigma}(\cdot) \tag{4.6}$$

with:

$$\dot{\sigma}(\cdot) = \sigma(\cdot)[1 - \sigma(\cdot)] \tag{4.7}$$

4.1.1 Feedforward propagation

The feedforward phase is computed as follows:

- The activation of the nth hidden neuron is obtained via:

$$\mathbf{S}_n^H = \sum_{m=1}^{n_i} \left(\mathbf{W}_{nm}^{IH} \otimes \mathbf{I}_m\right) + \boldsymbol{\Theta}_n^H \tag{4.8}$$

- The output from the hidden neuron is then:

$$\mathbf{X}_n^H = \boldsymbol{\sigma}(\mathbf{S}_n^H) = \sigma(S_{0n}^H) + \imath\,\sigma(S_{1n}^H) + \jmath\,\sigma(S_{2n}^H) + \kappa\,\sigma(S_{3n}^H) \tag{4.9}$$

- The activation of the output neuron is computed by:

$$\mathbf{S}_n^O = \sum_{m=1}^{n_h} \left(\mathbf{W}_{nm}^{HO} \otimes \mathbf{X}_m^H\right) + \boldsymbol{\Theta}_n^O \tag{4.10}$$

- Finally, the network's output is:

$$\mathbf{Y}_n = \mathbf{X}_n^O = \boldsymbol{\sigma}(\mathbf{S}_n^O) = \sigma(S_{0n}^O) + \imath\,\sigma(S_{1n}^O) + \jmath\,\sigma(S_{2n}^O) + \kappa\,\sigma(S_{3n}^O) \tag{4.11}$$

4.1.2 Network error computation

The error function is defined as the mean squared difference between predicted and desired outputs:

$$\mathbf{Y}_n = Y_{0n} + \imath\,Y_{1n} + \jmath\,Y_{2n} + \kappa\,Y_{3n} \tag{4.12}$$

$$\mathbf{t}_n = t_{0n} + \imath\,t_{1n} + \jmath\,t_{2n} + \kappa\,t_{3n} \tag{4.13}$$

$$E = \frac{1}{2}\sum_{n=1}^{n_o} \left[(t_{0n} - Y_{0n})^2 + (t_{1n} - Y_{1n})^2 + (t_{2n} - Y_{2n})^2 + (t_{3n} - Y_{3n})^2\right] \tag{4.14}$$

4.2 Learning algorithms

The backpropagation process enables a neural network to iteratively update its weights and biases, based on the error between the actual output and the expected target. During training phase, the network processes multiple input instances, commonly referred to as *patterns*, and the resulting error is propagated backward through the layers to refine the model's parameters. Numerous strategies for MLPs have been introduced to leverage the information encoded in the error dynamics. Among them, the GD which utilizes the first derivative of the error function, and the LM algorithm, that incorporates second-order derivative information to enhance convergence.

In the context of HMLPs, these optimization approaches have been adapted from their real-valued counterparts by extending the underlying mathematical framework.

4.2.1 Gradient descent algorithm

Let us define the propagation of the error relative to the nth neuron of the output layer as [32]:

$$\delta_h^{OH} = e_{an}^{OH}\dot{\sigma}(S_{an}^O) + \imath e_{bn}^{OH}\dot{\sigma}(S_{bn}^O) + \jmath e_{cn}^{OH}\dot{\sigma}(S_{cn}^O) + \kappa e_{dn}^{OH}\dot{\sigma}(S_{dn}^O) \tag{4.15}$$

where: $h = 1, ..., n_o$ is the number of the neuron considered in the output layer, and

$$\mathbf{e}_n^{OH} = e_{an}^{OH} + \imath e_{bn}^{OH} + \jmath e_{cn}^{OH} + \kappa e_{dn}^{OH} \tag{4.16}$$

is the quaternion defined as:

$$\mathbf{e}_n^{OH} = \mathbf{t}_n - \mathbf{Y}_n \tag{4.17}$$

providing the error value calculated with respect to the nth pattern, with $n = 1, ..., M$.

The corrections of the weights and biases (respectively $\Delta\mathbf{W}$ and $\Delta\mathbf{\Theta}$) of the hidden layer neurons are computed with the following formulas:

$$\Delta\mathbf{W}_{hn}^{HO} = \eta\delta_n^{OH} \otimes (\mathbf{X}_h^H)^* \tag{4.18}$$

$$\Delta\mathbf{\Theta}_n^{OH} = \eta\delta_n^{OH} \tag{4.19}$$

were η is the learning rate.

We get the hidden layer error from:

$$\mathbf{e}_n^H = \sum_{h=1}^{N_O} (\mathbf{W}_{hn}^{HO})^* \otimes \delta_h^{OH} \tag{4.20}$$

and then we can compute the propagation of the error from the hidden layer to the input one δ_h^{HI}:

$$\delta_h^{HI} = e_{an}^H\dot{\sigma}(S_{an}^H) + \imath e_{bn}^H\dot{\sigma}(S_{bn}^H) + \jmath\jmath e_{cn}^H\dot{\sigma}(S_{cn}^H) + \kappa\kappa e_{dn}^H\dot{\sigma}(S_{dn}^H) \tag{4.21}$$

where $h = 1, ..., n_h$ is the number of neurons considered in the hidden layer.

The corrections of the weights and biases of the input layer neurons are computed with the following formulas:

$$\Delta\mathbf{W}_{hn}^{IH} = \eta\delta_n^{HI} \otimes (\mathbf{X}_h^I)^* \tag{4.22}$$

$$\Delta\mathbf{\Theta}_n^I = \eta\delta_n^{HI} \tag{4.23}$$

Therefore, the GD learning algorithm for HMLPs can be summarized indicating with $\mathbf{W} \in \mathbb{H}^N$ the set of all weights and biases of the network, as follows:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \Delta\mathbf{W}(k) \tag{4.24}$$

where N is the total number of weights and biases of the network, k represent the epoch and $\Delta\mathbf{W}(k)$ is selected according to either (4.18) - (4.19) or (4.22) - (4.23) on the basis of the specific weights and biases.

This method ensures that the weights and biases, which guarantee the optimal performance is term of minimizing $E(\mathbf{w})$, are obtained after a sufficient number of learning epochs.

4.2.2 Levenberg-Marquardt algorithm

The LM method relies on the second-order derivative of the error, thus previously a deeper knowledge on error dynamics at the cost of a computationally intensive calculation.

The error is a function from the matrix $\overset{\mathbb{R}}{\mathbf{W}}$ that contains all the weights and biases of the network. Recalling the formulation of the function reported in (4.14), it can be rewritten as:

$$E(\overset{\mathbb{H}}{\mathbf{w}}) = \frac{1}{2}\sum_{m=1}^{M}(\mathbf{e}^m)^\dagger \mathbf{e}^m \tag{4.25}$$

where $\overset{\mathbb{H}}{\mathbf{w}} \in \mathbb{H}^N$ is the N-dimensional weights and biases augmented quaternion vector (derived from (A.3)), $\mathbf{e}^m = [\mathrm{e}_1^m, \mathrm{e}_2^m, ..., \mathrm{e}_{n0}^m]$ is the error for the m-th training pattern and $(\mathbf{e}^m)^\dagger$ denotes the Hermitian transpose of $\mathbf{e}^m$.
Then, we can express $E(\overset{\mathbb{H}}{\mathbf{w}})$ as:

$$E(\overset{\mathbb{R}}{\mathbf{w}}) = \frac{1}{2}\sum_{m=1}^{M}(\overset{\mathbb{R}}{\mathbf{e}}^m)^T \overset{\mathbb{R}}{\mathbf{e}}^m \tag{4.26}$$

Now the Hessian matrix of the error function $E(\overset{\mathbb{R}}{\mathbf{w}})$ can be computed as:

$$\overset{\mathbb{R}}{\mathbf{H}} \coloneqq \frac{\partial}{\partial \overset{\mathbb{R}}{\mathbf{w}}}\left(\frac{\partial E(\overset{\mathbb{R}}{\mathbf{w}})}{\partial \overset{\mathbb{R}}{\mathbf{w}}}\right)^T \tag{4.27}$$

From the Hessian matrix, applying Newton's method, the update rule for the weights and biases can be derived as:

$$\overset{\mathbb{R}}{\mathbf{W}}(k+1) = \overset{\mathbb{R}}{\mathbf{W}}(k) - \overset{\mathbb{H}}{\mathbf{H}}^{-1}(k)\nabla E(\overset{\mathbb{R}}{\mathbf{w}}(k)) \tag{4.28}$$

where $\nabla E(\overset{\mathbb{R}}{\mathbf{w}}(k))$ is the gradient of the error function calculated as:

$$\begin{aligned}\nabla E(\overset{\mathbb{R}}{\mathbf{w}}(k)) &= \frac{1}{2}\sum_{m=1}^{M}\left(\frac{\partial \overset{\mathbb{R}}{\mathbf{e}}^m(k)^T}{\partial}\overset{\mathbb{R}}{\mathbf{w}}(k)\overset{\mathbb{R}}{\mathbf{e}}^m(k) + \overset{\mathbb{R}}{\mathbf{e}}^m(k)^T\frac{\partial \overset{\mathbb{R}}{\mathbf{e}}^m(k)}{\partial \overset{\mathbb{R}}{\mathbf{w}}(k)}\right)^T = \\ &= \sum_{m=1}^{M}\frac{\partial \overset{\mathbb{R}}{\mathbf{e}}^m(k)^T}{\partial \overset{\mathbb{R}}{\mathbf{w}}(k)^T}\overset{\mathbb{R}}{\mathbf{e}}^m(k) = \overset{\mathbb{R}}{\mathbf{J}}\overset{\mathbb{R}}{\mathbf{e}}(k)\end{aligned} \tag{4.29}$$

where $\overset{\mathbb{R}}{\mathbf{J}}$ is the Jacobian matrix containing the partial derivatives of each component of the error vector $\mathbf{e}^m$ (calculated for the m-th pattern) with respect to each component of $\overset{\mathbb{R}}{\mathbf{W}}$:

$$\overset{\mathbb{R}}{\mathbf{J}} = \begin{bmatrix} \frac{\partial e_a^1}{\partial w_a^1} & \frac{\partial e_a^1}{\partial w_b^1} & \frac{\partial e_a^1}{\partial w_c^1} & \frac{\partial e_a^1}{\partial w_d^1} & \frac{\partial e_a^1}{\partial w_a^2} & \cdots & \frac{\partial e_a^1}{\partial w_d^N} \\ \frac{\partial e_b^1}{\partial w_a^1} & \frac{\partial e_b^1}{\partial w_b^1} & \frac{\partial e_b^1}{\partial w_c^1} & \frac{\partial e_b^1}{\partial w_d^1} & \frac{\partial e_b^1}{\partial w_a^2} & \cdots & \frac{\partial e_b^1}{\partial w_d^N} \\ \frac{\partial e_c^1}{\partial w_a^1} & \frac{\partial e_c^1}{\partial w_b^1} & \frac{\partial e_c^1}{\partial w_c^1} & \frac{\partial e_c^1}{\partial w_d^1} & \frac{\partial e_c^1}{\partial w_a^2} & \cdots & \frac{\partial e_c^1}{\partial w_d^N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_d^M}{\partial w_a^1} & \frac{\partial e_d^M}{\partial w_b^1} & \frac{\partial e_d^M}{\partial w_c^1} & \frac{\partial e_d^M}{\partial w_d^1} & \frac{\partial e_d^M}{\partial w_a^2} & \cdots & \frac{\partial e_d^M}{\partial w_a^1} \end{bmatrix} \tag{4.30}$$

and $\overset{\mathbb{R}}{\mathbf{e}}$ is the vector:

$$\overset{\mathbb{R}}{\mathbf{e}} = \begin{bmatrix} \overset{\mathbb{R}}{\mathbf{e}}^1 \\ \overset{\mathbb{R}}{\mathbf{e}}^2 \\ \vdots \\ \overset{\mathbb{R}}{\mathbf{e}}^M \end{bmatrix} \in \mathbb{R}^{4Mn_0}$$

Newton's method can be computationally inefficient for two main reasons. First, the update rule involves the inverse of the Hessian matrix, which is known to potentially introduce numerical instabilities, especially when the matrix is not positive definite. Second, the calculation of second-order derivatives, which populate the Hessian, can be computationally demanding. To mitigate this drawback, a slight modification of the update rule can be adopted, as follows:

$$\overset{\mathbb{R}}{\mathbf{w}}(k+1) = \overset{\mathbb{R}}{\mathbf{w}}(k) - (\overset{\mathbb{R}}{\mathbf{H}}(k) + \lambda_k \mathbb{I}_{4N})^{-1} \nabla E(\overset{\mathbb{R}}{\mathbf{w}}(k)) \tag{4.31}$$

The introduction of the scaled identity matrix $\lambda_k \mathbb{I}_{4N}$, allows to improve the numerical stability of the inverse operation. Moreover, the scaling must be adopted at each iteration. As concern the complexity of computing second-order derivatives, a suitable approximation can be introduced, as:

$$\begin{aligned} \overset{\mathbb{R}}{\mathbf{H}} &\coloneqq \frac{\partial}{\partial \overset{\mathbb{R}}{\mathbf{w}}} \left(\frac{\partial E}{\partial \overset{\mathbb{R}}{\mathbf{w}}} \right)^T = \\ &= \sum_{m=1}^{M} \frac{\partial}{\partial \overset{\mathbb{R}}{\mathbf{w}}} \left(\frac{\partial \overset{\mathbb{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathbb{R}}{\mathbf{w}}^T} \overset{\mathbb{R}}{\mathbf{e}^m} \right) \approx \sum_{m=1}^{M} \frac{\partial \overset{\mathbb{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathbb{R}}{\mathbf{w}}^T} \frac{\partial \overset{\mathbb{R}}{\mathbf{e}^m}}{\partial \overset{\mathbb{R}}{\mathbf{w}}} = \overset{\mathbb{R}}{\mathbf{J}}^T \overset{\mathbb{R}}{\mathbf{J}} \end{aligned} \tag{4.32}$$

where the second-order derivatives of the Hessian with respect to each training pattern have been neglected.

Thus, combining (4.31) and (4.32), the LM algorithm update rule is obtained:

$$\overset{\mathbb{R}}{\mathbf{w}}(k+1) = \overset{\mathbb{R}}{\mathbf{w}}(k) - (\overset{\mathbb{R}}{\mathbf{J}}^T \overset{\mathbb{R}}{\mathbf{J}} + \lambda_k \mathbb{I}_{4N})^{-1} \overset{\mathbb{R}}{\mathbf{J}}^T \overset{\mathbb{R}}{\mathbf{e}} \tag{4.33}$$

The value of λ_k is, therefore, appropriately modified each interaction, according with the following rule:

$$\lambda_{k+1} = \begin{cases} \lambda_k \cdot \vartheta, & \text{if } E(\overset{\mathbb{R}}{\mathbf{w}}(k)) > E_{\text{best}} \text{ and } \lambda_k < \lambda_{\max}, \\ \lambda_k / \vartheta, & \text{if } E(\overset{\mathbb{R}}{\mathbf{w}}(k)) < E_{\text{best}} \text{ and } \lambda_k > \lambda_{\max}. \end{cases} \tag{4.34}$$

where ϑ is a arbitrary chosen update factor for λ_k, E_{best} is the minimum error already valued during learning, and , $\lambda_{\max}$ is threshold over which the update rule is almost negligible and weights are not changing. If the threshold is achieved and $E(\overset{\mathbb{R}}{\mathbf{w}}(k))$ remains higher than E_{best}, learning is stopped as the algorithm converges to a minimum.

Rule (4.34) makes the LM method a sort of compromise among the GD and the Newton method algorithm. In fact, when λ_k is very small, (4.33) approaches the solutions obtained by the Newton algorithm. On the contrary, when λ_k is very louse equation tends to the GD algorithm.

In order to efficiently implement this method in the context of quaternion algebra, we will adopt the $\mathbb{HR}$ calculus outlined in Appendix A.

The $\mathbb{HR}$ calculus, in fact, allows to recast the backpropagation in HMLPs in terms of augmented quaternion algebra, hence providing a more compact and tractable formulation of the weigths update rule.

Recalling equations (A.9) and (A.10) with $f = \overset{\mathbb{R}}{\mathbf{e}}^m$, it is possible to establish a relation between $\nabla E(\overset{\mathbb{R}}{\mathbf{w}})$ and $\nabla E(\overset{\mathbb{H}}{\mathbf{w}})$ as:

$$\begin{aligned}\nabla E(\overset{\mathbb{R}}{\mathbf{w}})^T &= \frac{\partial \overset{\mathbb{R}}{\mathbf{e}}^m}{\partial \overset{\mathbb{R}}{\mathbf{w}}} = \frac{\partial \overset{\mathbb{R}}{\mathbf{e}}^m}{\partial \overset{\mathbb{H}}{\mathbf{w}}} \mathbf{J}_N \\ &= \frac{\partial \left(\frac{1}{4} \mathbf{J}^{\dagger}_{n_0} \overset{\mathbb{H}}{\mathbf{e}}^m \right)}{\partial \overset{\mathbb{H}}{\mathbf{w}}} \mathbf{J}_N = \frac{1}{4} \mathbf{J}^{\dagger}_{n_0} \frac{\partial \overset{\mathbb{H}}{\mathbf{e}}^m}{\partial \overset{\mathbb{H}}{\mathbf{w}}} \mathbf{J}_N \\ &= \frac{1}{4} \mathbf{J}^{\dagger}_{n_0} \nabla E(\overset{\mathbb{H}}{\mathbf{w}}) \mathbf{J}_N\end{aligned} \tag{4.35}$$

so that:

$$\overset{\mathbb{R}}{\mathbf{J}} = \frac{1}{4} \mathbf{J}^{\dagger}_{n_0} \overset{\mathbb{H}}{\mathbf{J}} \mathbf{J}_N \tag{4.36}$$

where $\overset{\mathbb{H}}{\mathbf{J}}$ is the Jacobian matrix in the Hamiltonian domain:

$$\overset{\mathbb{H}}{\mathbf{J}} = \begin{bmatrix} \frac{\partial \overset{\mathbb{H}}{\mathbf{e}}_1}{\partial \overset{\mathbb{H}}{\mathbf{w}}} \\ \frac{\partial \overset{\mathbb{H}}{\mathbf{e}}_2}{\partial \overset{\mathbb{H}}{\mathbf{w}}} \\ \vdots \\ \frac{\partial \overset{\mathbb{H}}{\mathbf{e}}_M}{\partial \overset{\mathbb{H}}{\mathbf{w}}} \end{bmatrix} \tag{4.37}$$

It follows that:

$$\nabla E(\overset{\mathbb{R}}{\mathbf{w}}) = \overset{\mathbb{R}}{\mathbf{J}}^T \overset{\mathbb{R}}{\mathbf{e}} = \overset{\mathbb{R}}{\mathbf{J}}^{\dagger} \overset{\mathbb{R}}{\mathbf{e}} = \frac{1}{4} \mathbf{J}^{\dagger}_N \overset{\mathbb{H}}{\mathbf{J}}^{\dagger} \mathbf{J}_{n_0 M} \overset{\mathbb{R}}{\mathbf{e}} \tag{4.38}$$

Since:

$$\overset{\mathbb{R}}{\mathbf{e}} = \frac{1}{4} \mathbf{J}^{\dagger}_{n_0 M} \overset{\mathbb{H}}{\mathbf{e}} \tag{4.39}$$

then recalling:

$$\nabla E(\overset{\mathbb{R}}{\mathbf{w}}) = \left(\frac{1}{4} \mathbf{J}^{\dagger}_N \overset{\mathbb{H}}{\mathbf{J}}^{\dagger} \mathbf{J}_{n_0 M} \right) \left(\frac{1}{4} {\mathbf{J}^{\dagger}}_{n_0 M} \overset{\mathbb{H}}{\mathbf{e}} \right) = \frac{1}{4} \mathbf{J}^{\dagger}_N \overset{\mathbb{H}}{\mathbf{J}}^{\dagger} \overset{\mathbb{H}}{\mathbf{e}} \tag{4.40}$$

and then:

$$\begin{aligned}\overset{\mathbb{R}}{\mathbf{H}} \approx \overset{\mathbb{R}}{\mathbf{J}}^T \overset{\mathbb{R}}{\mathbf{J}} &= \overset{\mathbb{R}}{\mathbf{J}}^{\dagger} \overset{\mathbb{R}}{\mathbf{J}} = \frac{1}{4} \mathbf{J}^{\dagger}_N \overset{\mathbb{H}}{\mathbf{J}}^{\dagger} \overset{\mathbb{H}}{\mathbf{e}} \\ &= \left(\frac{1}{4} \mathbf{J}^{\dagger}_N \overset{\mathbb{H}}{\mathbf{J}}^{\dagger} \overset{\mathbb{H}}{\mathbf{J}}_{n_0 M} \right) \left(\frac{1}{4} \mathbf{J}^{\dagger}_{n_0 M} \overset{\mathbb{H}}{\mathbf{J}} \mathbf{J}_N \right) \\ &= \frac{1}{4} \mathbf{J}^{\dagger}_N \overset{\mathbb{H}}{\mathbf{J}}^{\dagger} \overset{\mathbb{H}}{\mathbf{J}} \mathbf{J}_N\end{aligned} \tag{4.41}$$

Now it is possible to recast the update rule (4.33) in the $\mathbb{H}$ domain as:

$$\begin{aligned}\frac{1}{4}\overset{\mathbb{H}^\dagger}{\mathbf{J}}_N\overset{\mathbb{H}}{\mathbf{W}}(k+1) &= \frac{1}{4}\overset{\mathbb{H}^\dagger}{\mathbf{J}}_N\overset{\mathbb{H}}{\mathbf{W}}(k)\\ &\quad-\left(\frac{1}{4}\mathbf{J}_N^\dagger\overset{\mathbb{H}^\dagger}{\mathbf{J}}\overset{\mathbb{H}}{\mathbf{J}}\mathbf{J}_N+\lambda_k\mathbf{I}_{4N}\right)^{-1}\frac{1}{4}\mathbf{J}_N^\dagger\overset{\mathbb{H}^\dagger}{\mathbf{J}}\overset{\mathbb{H}}{\mathbf{e}}_k\end{aligned} \tag{4.42}$$

Let us focus on the term

$$\left(\frac{1}{4}\mathbf{J}_N^\dagger\overset{\mathbb{H}^\dagger}{\mathbf{J}}\overset{\mathbb{H}}{\mathbf{J}}\mathbf{J}_N+\lambda_k\mathbf{I}_{4N}\right)^{-1}\frac{1}{4}\mathbf{J}_N^\dagger$$

which, recalling that $I_{4N}=\frac{1}{4}\mathbf{J}^\dagger{}_N\mathbf{J}_N$, we can rewrite as:

$$\left(\frac{1}{4}\mathbf{J}_N^\dagger\left[\overset{\mathbb{H}}{\mathbf{J}^\dagger}\overset{\mathbb{H}}{\mathbf{J}}+\lambda_k\mathbf{I}_{4N}\right]\mathbf{J}_N\right)^{-1}\frac{1}{4}\mathbf{J}_N^\dagger$$

thus,

$$\begin{aligned}&4\,\mathbf{J}_N^{-1}\left(\overset{\mathbb{H}}{\mathbf{J}^\dagger}\overset{\mathbb{H}}{\mathbf{J}}+\lambda_k\mathbf{I}_{4N}\right)^{-1}\mathbf{J}_N^{\dagger-1}\frac{1}{4}\mathbf{J}_N^\dagger\\ &\quad=\mathbf{J}_N^{-1}\left(\overset{\mathbb{H}}{\mathbf{J}^\dagger}\overset{\mathbb{H}}{\mathbf{J}}+\lambda_k\mathbf{I}_{4N}\right)^{-1}\end{aligned} \tag{4.43}$$

Now, inverting eq. (A.5), we get:

$$(\mathbf{J}\dagger_N\mathbf{J}_N)^{-1}=(4\,\mathbf{I}_{4N})^{-1}$$

from which:

$$\mathbf{J}_N^{-1}\,\mathbf{J}\dagger_N^{-1}=\frac{1}{4}\,\mathbf{I}_{4N}$$

hence:

$$\mathbf{J}_N^{-1}=\frac{1}{4}\,\mathbf{J}^\dagger{}_{4N} \tag{4.44}$$

Substituting 4.44 into 4.42, we obtain the update rule for the Levenberg–Marquardt learning algorithm in the $\mathbb{H}$ algebra:

$$\overset{\mathbb{H}}{\mathbf{W}}(k+1)=\overset{\mathbb{H}}{\mathbf{W}}(k)-\left(\overset{\mathbb{H}}{\mathbf{J}^\dagger}(k)\overset{\mathbb{H}}{\mathbf{J}}(k)\;+\;\lambda_k\mathbf{I}_{4N}\right)^{-1}\overset{\mathbb{H}}{\mathbf{J}^\dagger}(k)\,\overset{\mathbb{H}}{\mathbf{e}}_k$$

Indicating:

$$\overset{\mathbb{H}}{\mathbf{H}}_k=\overset{\mathbb{H}}{\mathbf{J}^\dagger}\overset{\mathbb{H}}{\mathbf{J}}=\begin{bmatrix}\mathbf{H}_k^1 & \mathbf{H}_k^2 & \mathbf{H}_k^3 & \mathbf{H}_k^4\\ \imath\mathbf{H}_k^2\imath^{-1} & \imath\mathbf{H}_k^1\imath^{-1} & \imath\mathbf{H}_k^4\imath^{-1} & \imath\mathbf{H}_k^3\imath^{-1}\\ \jmath\mathbf{H}_k^3\jmath^{-1} & \jmath\mathbf{H}_k^4\jmath^{-1} & \jmath\mathbf{H}_k^1\jmath^{-1} & \jmath\mathbf{H}_k^2\jmath^{-1}\\ \kappa\mathbf{H}_k^4\kappa^{-1} & \kappa\mathbf{H}_k^3\kappa^{-1} & \kappa\mathbf{H}_k^2\kappa^{-1} & \kappa\mathbf{H}_k^1\kappa^{-1}\end{bmatrix} \tag{4.45}$$

and

$$\nabla E(\overset{\mathbb{H}}{\mathbf{W}}(k))=\overset{\mathbb{H}}{\mathbf{J}^\dagger}\overset{\mathbb{H}}{\mathbf{e}}_k=\begin{bmatrix}\nabla E(\overset{\mathbb{R}}{\mathbf{W}}(k))\\ \imath\nabla E(\overset{\mathbb{R}}{\mathbf{W}}(k))\imath^{-1}\\ \jmath\nabla E(\overset{\mathbb{R}}{\mathbf{W}}(k))\jmath^{-1}\\ \kappa\nabla E(\overset{\mathbb{R}}{\mathbf{W}}(k))\kappa^{-1}\end{bmatrix} \tag{4.46}$$

it follows

$$\overset{\mathbb{H}}{\mathbf{W}}(k+1) = \overset{\mathbb{H}}{\mathbf{W}}(k) - \left(\overset{\mathbb{H}}{\mathbf{H}}_k + \lambda_k \mathbf{I}_{4N}\right)^{-1} \nabla E(\overset{\mathbb{H}}{\mathbf{W}}(k)) \tag{4.47}$$

4.3 Learning-on-Learning approach

The main idea of this manuscript hinges on the fact that, independently from the specific learning algorithm adopted, the update rules (4.24) or (4.47) determine an evolution which follows a trajectory in the weights space that depends on the values assumed in earlier iterations. Therefore, this evolution can be considered as the evolution of a discrete-time dynamical systems in the quaternion domain. Under this perspective, the causal relationship established by the learning process can be itself predicted by using a neural network approach [67]. Based on this assumption, a strategy to improve the convergence time of the learning process can be to instantiate a number of smaller-size auxiliary networks learning in parallel the trend of $\overset{\mathbb{H}}{\mathbf{W}}(k)$. We will call this approach Learning-on-Learning.

This means that, after a number k_{min} of epochs for the main network, suitable patterns are created from each weight evolution with the aim of predicting its convergence value. In order to do this, $\overset{\mathbb{H}}{\mathbf{W}}(k)$ dynamics must be associated with as ansatz as:

$$\overset{\mathbb{H}}{\mathbf{W}}(k) = f\left(\overset{\mathbb{H}}{\mathbf{W}}(k-1), \overset{\mathbb{H}}{\mathbf{W}}(k-2), \ldots, \overset{\mathbb{H}}{\mathbf{W}}(k-n_w)\right) \tag{4.48}$$

where n_w is the number of regressors included in the ansatz.

The computational efficiency of the auxiliary networks during the learning phase is crucial to outpace the primary network learning process. The determination of the number of auxiliary networks to deploy involves considerations related to the quantity of weights to predict and the capacity of the available hardware. Optimally, each computing core should be associated with a dedicated auxiliary neural network.

During learning, some weights will rapidly converge and only slight or negligible variations can be observed. This occurs when $\Delta \mathbf{W}_{ij} < \epsilon$. Therfore, after $k_{\min}$ epochs only those weights not satisfing this condition have to be predicted. For these quantities, training are created according to the ansatz 4.48.

These networks will immediately begin the learning phase on the available patterns. Once that the weights are predicted, they are sent to the main network, substituted into the actual weight matrix, and finally the error is computed. The new weight matrix is accepted only if the following condition is satisfied:

$$E(\mathbf{W}_{pred}) < E(\mathbf{W}_{prec}) \tag{4.49}$$

where $E(\mathbf{W}_{pred})$ represents the error associated to the predicted weight matrix and $E(\mathbf{W}_{prec})$ is the error associated with the weight matrix before the substitution.

It has been observed that when the estimated weights are sent to the main network, the error, in some cases, can assume large values, so that it

is necessary to perform a certain number of learning epochs after which the error value reaches the steady one.

A summarizing scheme going through the various step of learning on learning is showed in fig 4.1.

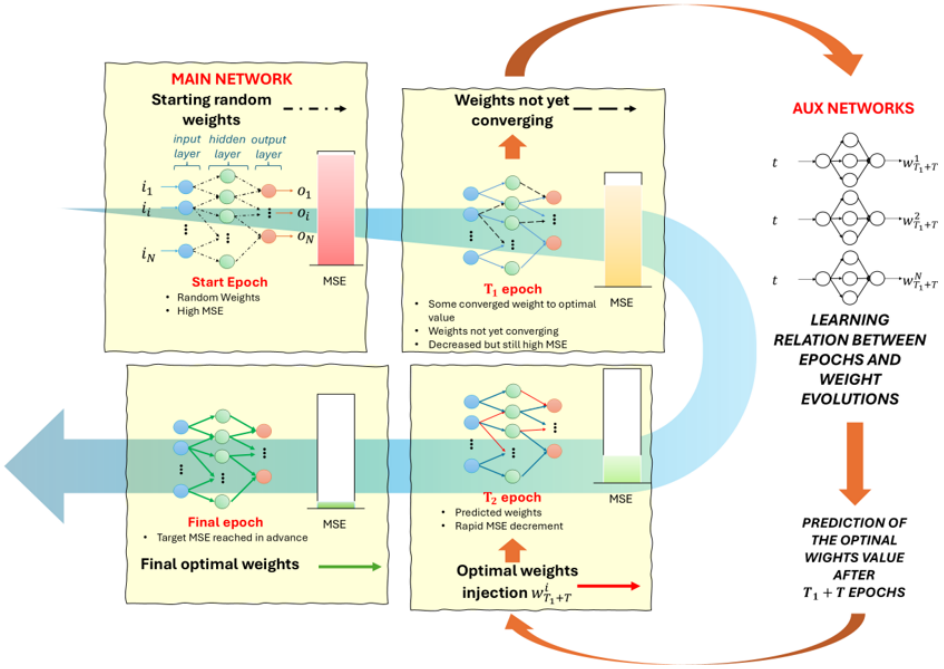

FIGURE 4.1: Scheme representing the step-by-step evolution of the learning-on-learning approach.

4.3.1 A first demonstrative example

The following example is intended to illustrate the functioning of the Learning-on-Learning method during the training phase. A HMLP is trained to predict a chaotic time series generated by the Chua system, whose governing equations are presented in (4.50).

$$\begin{cases} \dot{x} = \alpha(y - x - h(x)) \\ \dot{y} = x - y + z \\ \dot{z} = -\beta y \\ h(x) = m_1 * x + 0.5(m_0 - m_1)(|x + 1| - |x - 1|) \end{cases} \tag{4.50}$$

With the parameter configuration $\alpha = 9.0$, $\beta = 14.28$, $\gamma = -\frac{8}{7}$, $m_0 = 0.03$, and $m_1 = -\frac{5}{7}$, the Chua system exhibits a double-scroll attractor, as shown in Figures 4.2 and 4.3. The objective is to enable the network to perform one-step prediction, i.e., to estimate the three state variables at time step $k + 1$ based on the input at time k. The data-to-neuron mapping for this example is structured as follows:

- One input neuron, whose imaginary components $\{\imath, \jmath, \kappa\}$ represent the three signal values at time k;
- One output neuron, whose imaginary components represent the corresponding signal values at time $k + 1$;

- In both cases, the real component is fixed at zero.

The dataset used for training, highlighted in red, consists of 500 samples extracted as the learning model for the network.

The number of hidden neurons n_h of the used HMLP is 2. Then, the network has 4 quaternion weights (16 components), and 3 quaternion bias (12 components).

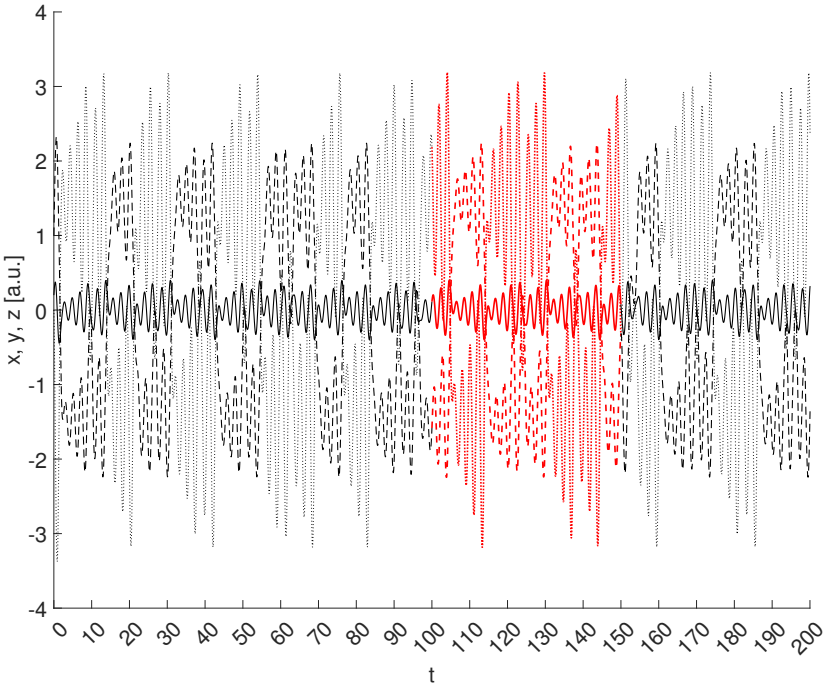

FIGURE 4.2: Three signals of Chua system state variables. The red part is the one used for the learning phase.

Figures 4.4 and 4.5 illustrate the evolution of the quaternion weight components during the training phase. In Fig. 4.4, the weight trajectories obtained using the classical gradient descent algorithm (GD, blue dashed line) are compared with those obtained using the gradient descent algorithm augmented by the learning-on-learning method (GDL, orange line). Similarly, Fig. 4.5 presents a comparison between the Levenberg–Marquardt algorithm (LM, blue dashed line) and its learning-on-learning variant (LML, orange line).

In both cases, the networks start from identical initial weight configurations. As expected, the evolution of weights initially follows the same path for both standard and learning-on-learning-enhanced algorithms, due to the deterministic nature of the backpropagation process. A clear divergence point can be observed, marking the moment when the learning-on-learning mechanism becomes active: the orange line abruptly separates from the blue dashed one. This corresponds to the instant when the weight values predicted by the auxiliary networks are injected into the main network.

Then, a certain number of epochs is performed in order that the error value reaches the expected value. In that time interval, it is notable that the weights have some value oscillation before reaching the optimal value.

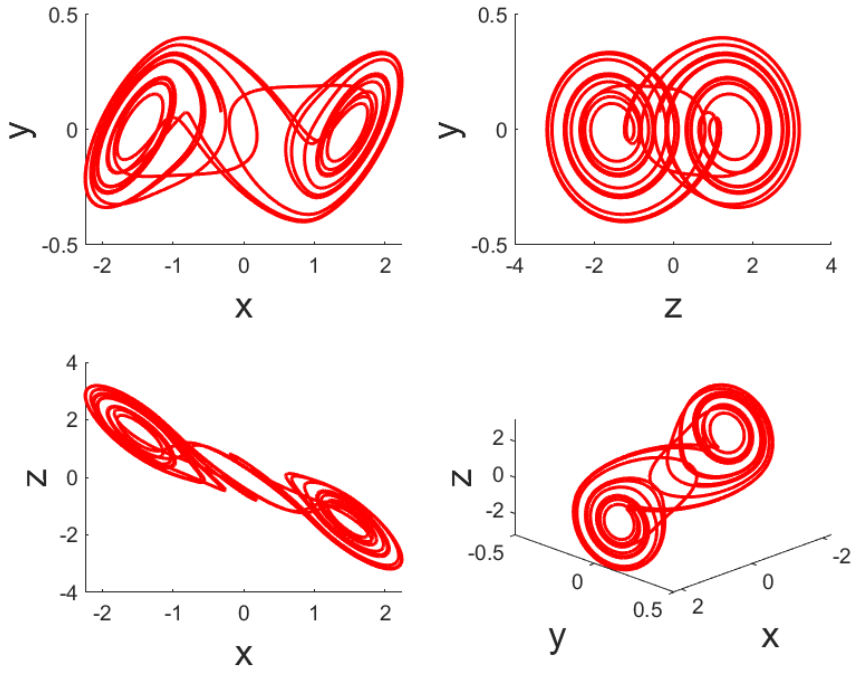

FIGURE 4.3: Four projections of the Chua system double scroll actractor.

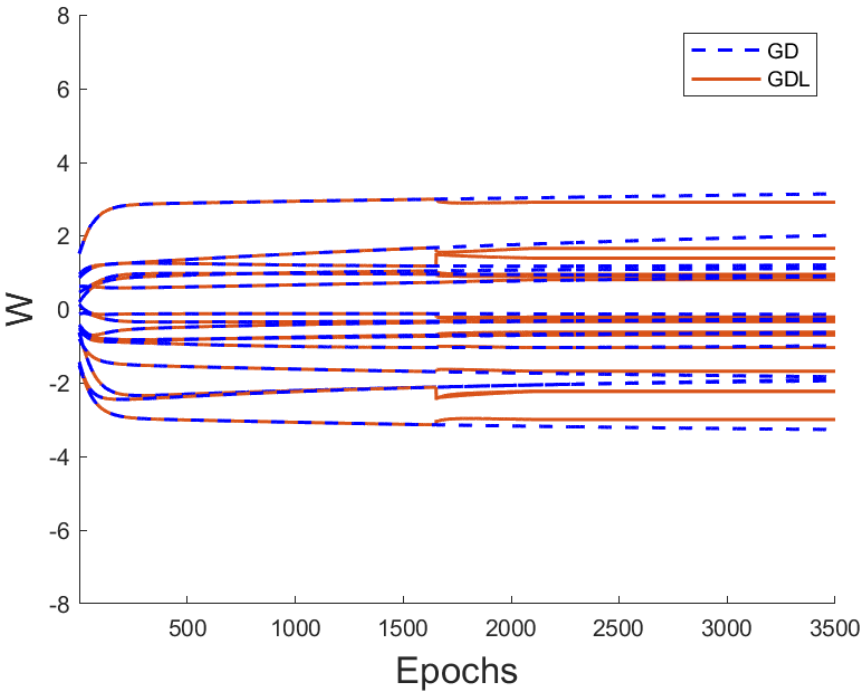

FIGURE 4.4: Evolution of the four components of the weights in GD method in the first 350 epochs in Chua system learning. GD is represented in blue, GDL is represented in orange.

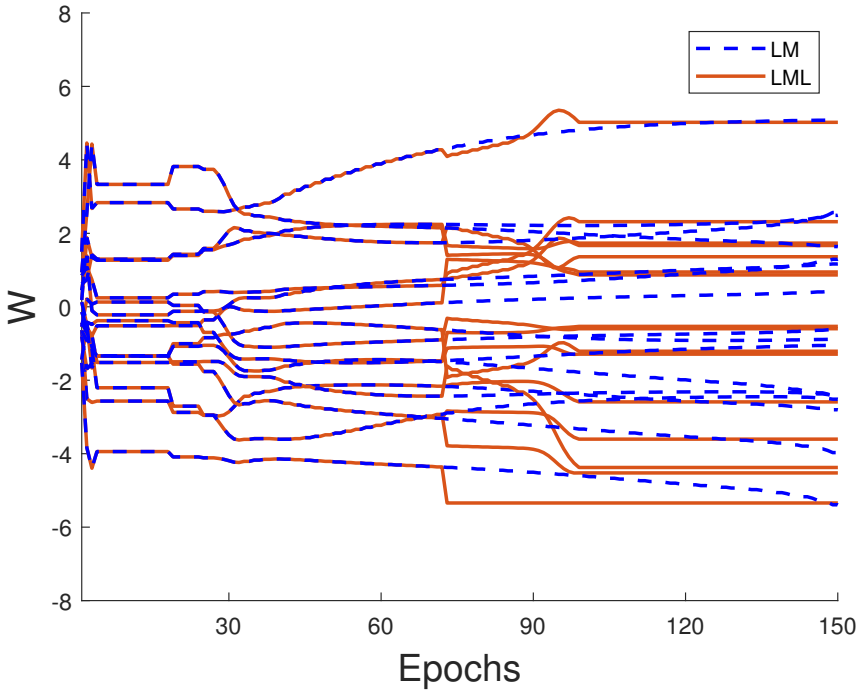

FIGURE 4.5: Evolution of the four components of the weights in LM method in the first 150 epochs in Chua system learning. LM is represented in blue, LML is represented in orange.

4.4 Results and performance comparison between different networks

In this section, a series of comparative evaluations are carried out to highlight the pros and cons of the proposed approaches, suggesting possible improvements for future developments.

Initially, the performance of the HMLP and a real-valued MLP are compared, both employing the GD algorithm with the Learning-on-Learning strategy. The application of Learning-on-Learning to MLPs was introduced and compared to standard GD algorithm in [69], where it was highlighted the superiority of such a tool in learning convergence speed.

After assessing the benefits of using quaternion-valued neural networks for specific tasks as opposed to real-valued ones, a second comparison campaign is conducted involving HMLP trained with GD and LM algorithms against their Learning-on-Learning counterparts across various problem scenarios. These tests aim to assess the potential benefits in terms of convergence speed and model efficiency offered by the combination of hypercomplex representations and Learning-on-Learning approach.

4.4.1 HMLP for Planetary Orbits Prediction

The objective of this introductory study is to emphasize the significance of HMLPs as effective tools for processing high-dimensional data across various scientific and engineering fields. Given that trajectories are inherently three-dimensional, quaternion representation is particularly suitable for accurately modeling their spatial paths.

In this context, real-valued MLPs and quaternion-valued HMLPs employing gradient descent algorithm with Learning-on-Learning optimization are compared regarding their ability to predict complex orbital trajectories using a data-driven approach [70, 71].

To begin with, we will present the study task of this experiment. Due to the vast distances involved, often spanning thousands of light years, current sensors exhibit limitations in precisely tracking these orbits [72]. Although traditional observational data remain valuable, they frequently lack the precision necessary to characterize complex trajectories.

Training patterns and performance evaluations for the two HMLPs were generated using the SPICE system provided by the Navigation and Ancillary Information Facility (NAIF). SPICE offers tools that facilitate the planning, modeling, and execution of planetary exploration missions. The SPICE data sets contain navigation and ancillary information that define precise observation geometries for planetary science and engineering purposes. These kernels are produced by authoritative sources and include metadata essential for interpretation.

Using the MATLAB implementation of SPICE, planetary trajectorie, also referred to as epicycloids, were extracted with Earth as the reference observer (more precisely, the barycenter of each planet-satellite system) within a three-dimensional xyz coordinate system. Specifically, the trajectory of Mars from June 1, 2000, to June 1, 2010, was analyzed. The resulting curve was sampled at 3000 points, corresponding to a sampling interval of approximately 29 hours and 15 minutes, as illustrated in Fig. 4.6.

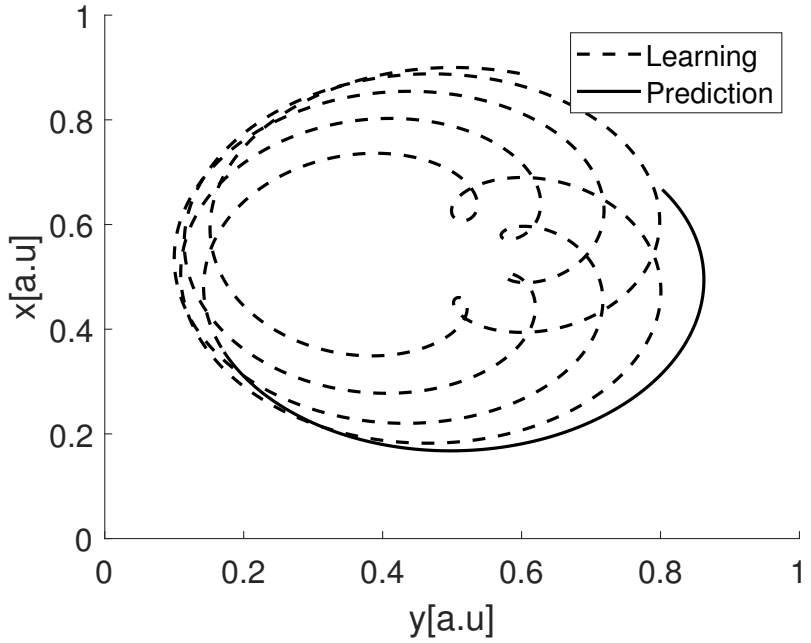

FIGURE 4.6: Normalized x,y-coordinates of Mars trajectory from June 1, 2000, to June 1, 2010.

Various tests were conducted on the curve shown in Fig. 4.6, with coordinates normalized within the range $[0.1, 0.9]$, using the two previously mentioned network models. The first 2500 samples of the curve were employed

for training the networks, while the subsequent 200 samples were reserved for evaluating the prediction accuracy.

The training parameter of the networks are:

- 15-20-10-3 architecture for the MLP, 5-15-10-1 for the HMPL;
- learning rate 0.1 for the MLP, while learning rate 0.01 for the HMLP;
- momentum 0.5 for the MLP, while momentum 0.1 for the HMLP;
- both the networks use the Learning-On-Learning algorithm in learning phase.

The inputs to the networks consist of the three spatial coordinates of five consecutive points, $[p_{k-5}, p_{k-4}, \ldots, p_{k-1}]$, along the trajectory, where k denotes the index of the current sample. The targets correspond to the three spatial coordinates of the point p_k. Since each neuron in the HMLP processes quaternion inputs and outputs, the network employs 5 input neurons, where the imaginary components of each input quaternion represent the spatial coordinates of the points, and the real part is fixed at zero.

Network evaluation was performed using an n-step prediction with $n = 30$. Specifically, starting from samples indexed $k-5$ to $k-1$ as initial conditions, the networks predicted the sample at index k. This predicted point was then fed back as input for subsequent predictions. To prevent divergence caused by noise accumulation during iterative prediction, the actual trajectory points were reintroduced every 30 steps to correct the prediction trend.

The performance of the networks is illustrated in Fig. 4.7, which compares the 30-step predictions with the original trajectory, and in Fig. 4.8, which presents the pointwise mean square error for each coordinate.

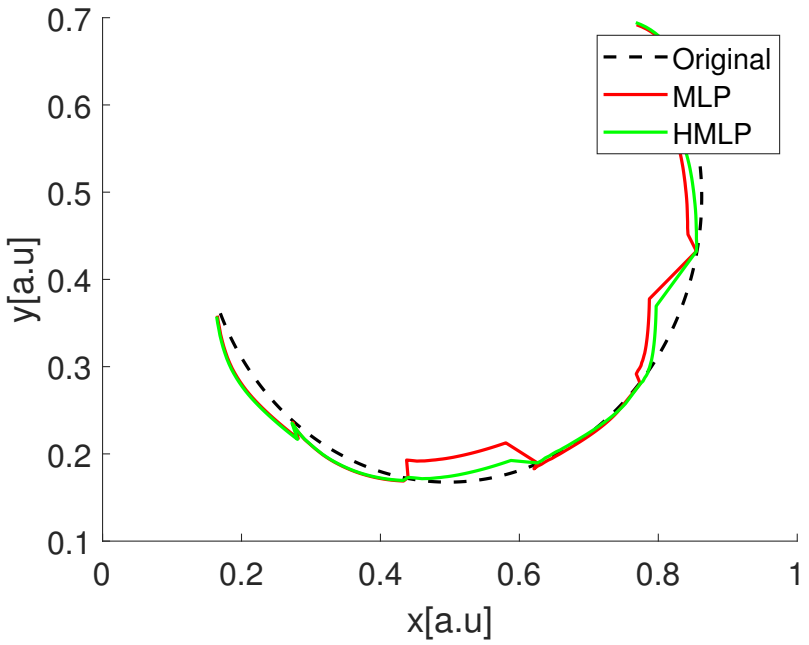

FIGURE 4.7: Comparison between the original trajectory and the predicted ones.

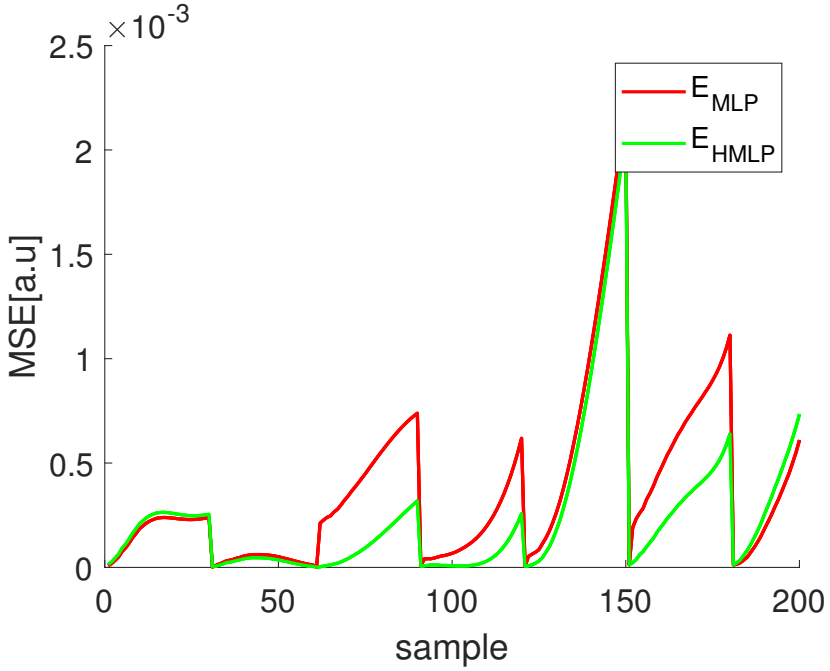

FIGURE 4.8: The point-to-point mean square error for each coordinate.

Considering the first 60 samples, the prediction errors of both MLP models are comparable. Beyond the 60^{th} sample, however, the HMLP consistently outperforms the real-valued MLP. Notably, the HMLP achieves this with approximately 10,000 training epochs compared to around 14,000 epochs for the MLP. This reduction in training cycles, combined with the smaller number of parameters in the HMLP architecture, highlights its superior learning efficiency.

As shown in the graphs, prediction accuracy degrades over time, yet the HMLP maintains a high level of precision, especially considering that each sample corresponds to a sampling interval exceeding 29 hours. The mean squared error (MSE) remains below 5×10^{-4} up to the 120^{th} sample, corresponding to roughly 145 days.

It is important to note that accuracy could be further improved by increasing the number of training epochs with a lower learning rate and using more samples. The current results were obtained with relatively few training cycles and samples, considering the complexity of the approximation task.

4.4.2 Prediction of chaotic time series: Hyperchaotic Lorenz system

After establishing the superiority of HMLP with GD and Learning-on-Learning algorithms in prediction problems, another case study illustrates the performance differences between the Learning-on-Learning approach and its standard counterparts by applying the network algorithms to predict time series generated by the hyperchaotic Lorenz system, described by the following state-space equations:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - \gamma y + w \\ \dot{z} = xy - \beta z \\ \dot{w} = -ky - rw \end{cases} \tag{4.51}$$

with $\sigma = 10$, $\rho = 15$, $\gamma = 1$, $\beta = 1$, $k = 30$, $r = 5$ and initial conditions [1 1 1 1]. The system has been integrated using a time step $\Delta t = 0.01$.

The two types of networks (or four, when including their learning-on-learning counterparts) are tasked with learning a segment of the system's time evolution using the minimum number of training epochs, with the goal of reproducing another segment of the trajectory while minimizing the mean squared error (MSE). The signals used for training and testing are shown in Fig. 4.9.

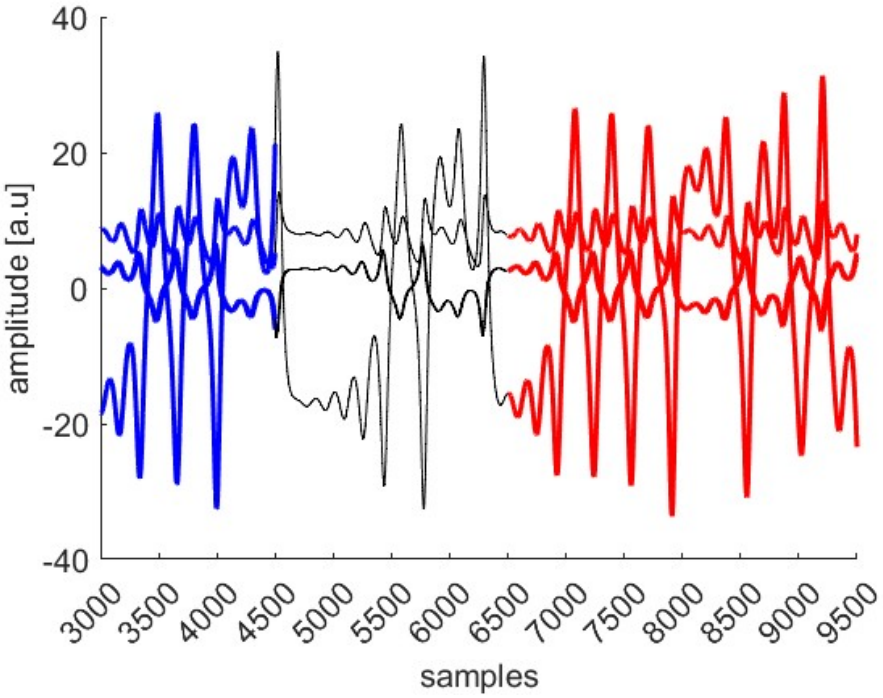

FIGURE 4.9: Four signals of hyperchaotic Lorenz system state variables. The blue part is the one used for the learning phase, while the red part is the one used for test.

The portion of the signals used for the training phase consists of 1500 samples and serves as input patterns for the networks, while the testing phase is based on an additional 3000 samples. Each network features a single input neuron, whose quaternion components represent the four signal values at time step k, and a single output neuron, whose components correspond to the signal values at time step $k+1$. In summary, the input to the network comprises the four-dimensional signal at index k, and the output corresponds to the signal at index $k+1$, enabling a one-step-ahead prediction of the chaotic time series.

All four networks adopt the same architecture, consisting of a single hidden layer with a variable number of neurons, $n_h \in \{2, \ldots, 10\}$.

As previously noted, a large number of training samples was intentionally

used to highlight the differences in convergence speed between the conventional learning algorithms and their learning-on-learning counterparts. Each configuration, defined by a specific value of n_h, was evaluated over 10 independent trials to ensure statistical robustness.

Performance analysis

Figures 4.10 and 4.11 report the mean squared error (MSE) between the original and predicted signals, along with the number of training epochs, as a function of the number of hidden neurons. Figure 4.10 compares the performance of GD and GDL, while Figure 4.11 presents the same comparison for the LM and LML.

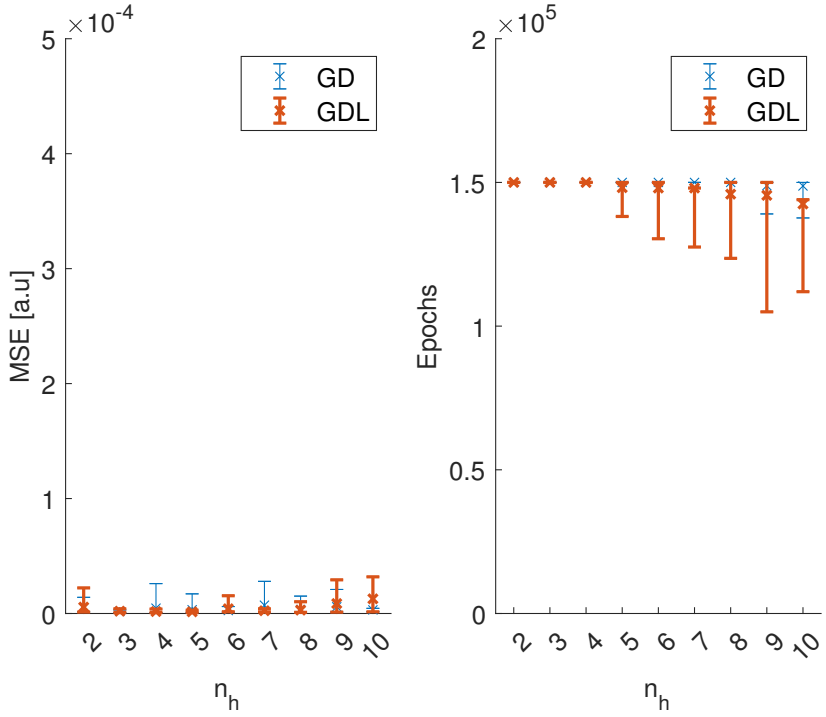

FIGURE 4.10: Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Lorenz system. N_{hn} is the number of adopted hidden neurons. GD is represented in blue, GDL is represented in orange.

The most representative results from the extensive set of tests are reported. Figure 4.12 presents a comparison of the attractors corresponding to the state variables (x, y, z) as reconstructed by the four different training methods.

Figure 4.13 shows the attractors corresponding to the error signals obtained during the test phase for the state variables of the Lorenz system, defined as $(x - x_o, y - y_o, z - z_o)$, where (x, y, z) represent the exact values of the state variables at each time step and (x_o, y_o, z_o) denote the corresponding outputs predicted by the networks.

In Fig.4.14 it is represented the MSE per sample on the four state variables of the Lorenz system.

In Figures 4.10 and 4.11, we can observe that the error bars corresponding to the mean squared error are nearly identical for both methods, as well it is evident in 4.12 and 4.13. This indicates that the improvement carried on

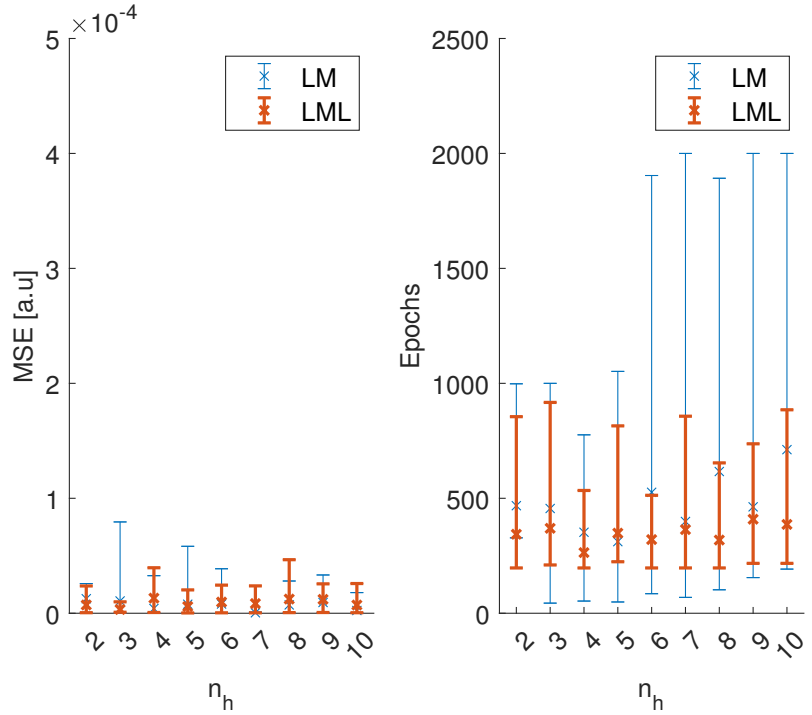

FIGURE 4.11: Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Lorenz system. N_{hn} is the number of adopted hidden neurons. LM is represented in blue, LML is represented in orange.

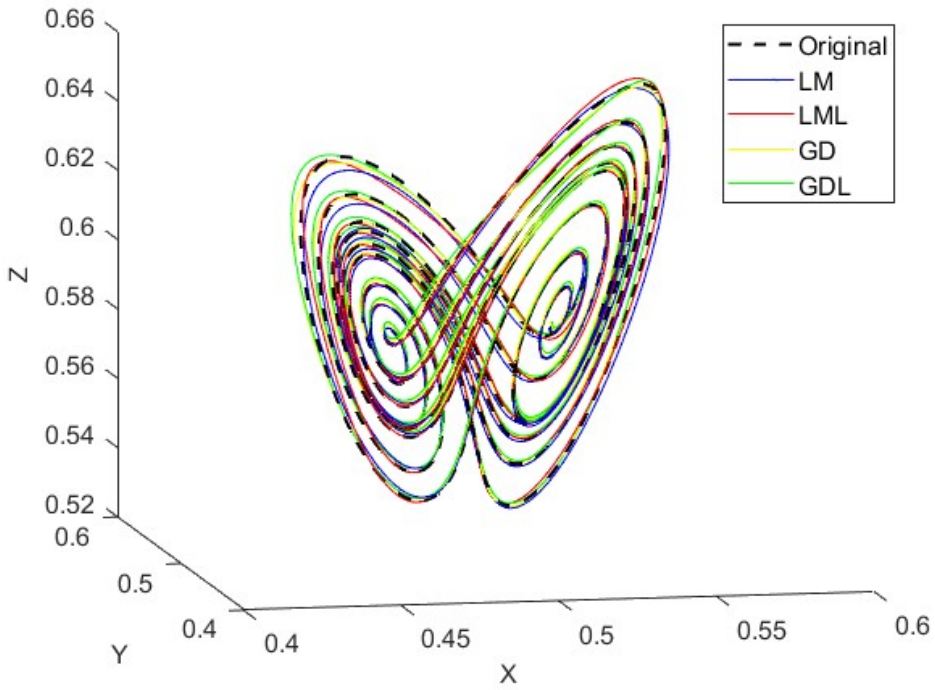

FIGURE 4.12: Actractors of the state variables (x, y, z) obtained by the four different methods. The original actractor is represented by black dashed line, LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green.

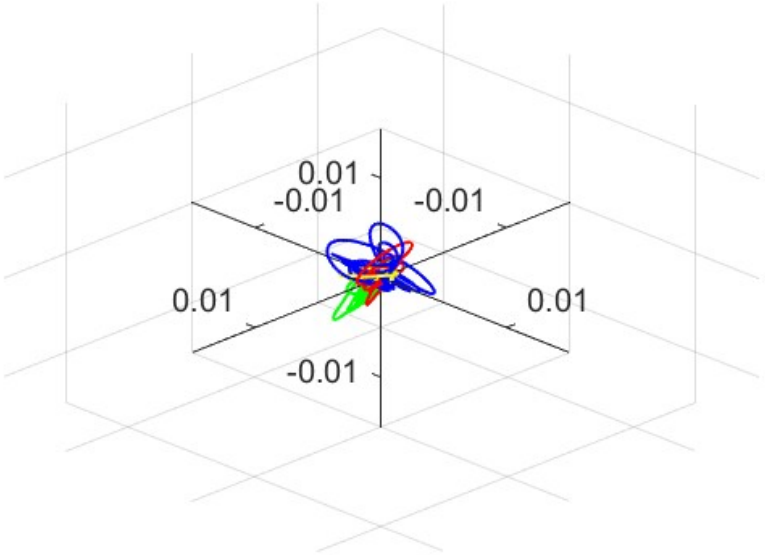

FIGURE 4.13: Actractor of the error between target and output of the networks in test phase on Lorenz system. LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green.

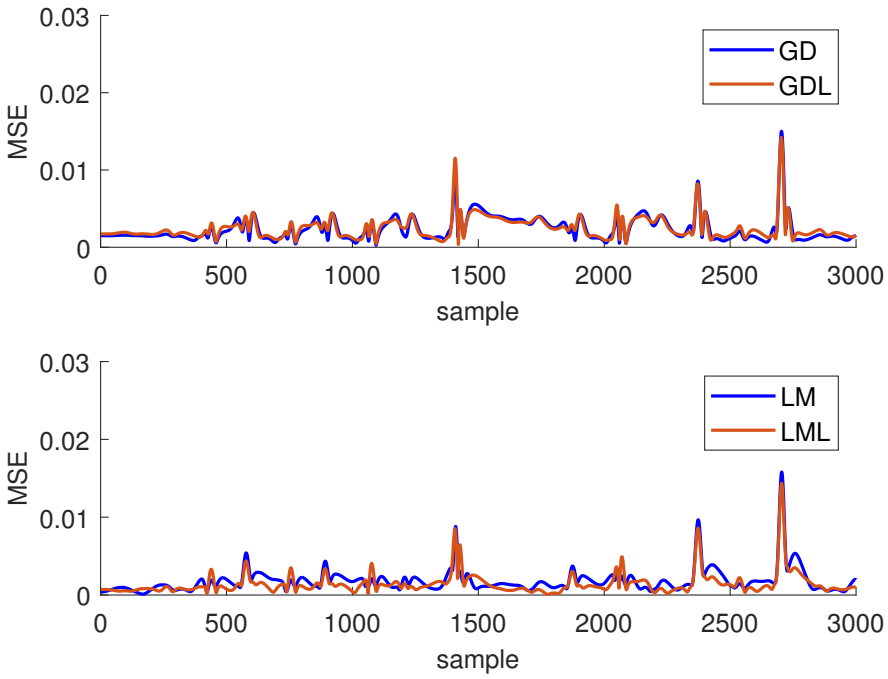

FIGURE 4.14: MSE per sample on the four state variables (x, y, z, ω) of the Lorenz system. On the top, the comparison between GD (blue) and GDL (orange), on the bottom the comparison between LM (blue) and LML (orange).

by the learning-on-learning strategy is not in terms of performance accuracy. However, the real advantage lies in the reduction of the number of epochs required, which is particularly significant in networks with a large number of neurons, as we already said at the beginning of this discussion. Indeed, the differences between the mean value of each bar, represented by the "x" marker, encrease with n_h.

Weight trend analysis

As demonstrated in previous results, the advantage of the learning-on-learning strategy becomes more pronounced as the number of neurons, n_h, increases. Figures 4.10 and 4.11 show that, while both methods achieve similar performance, likely converging to the same local minima, the difference in required epochs grows with larger network sizes.

Two key aspects characterize the weight dynamics at the start of training. First, networks with fewer neurons ($n_h = 2, 3, 4$) exhibit faster transient weight updates compared to larger architectures ($n_h = 8, 9, 10$). Second, during GD learning, weight evolution progresses relatively slowly, as shown in Fig. 4.15. Incorporating learning-on-learning exploits this by delaying auxiliary network pattern formation until the mean gradient of weight corrections exceeds a threshold. This approach avoids the initial transient phase and enables clearer pattern identification, improving the auxiliary networks' prediction of the weights' final values.

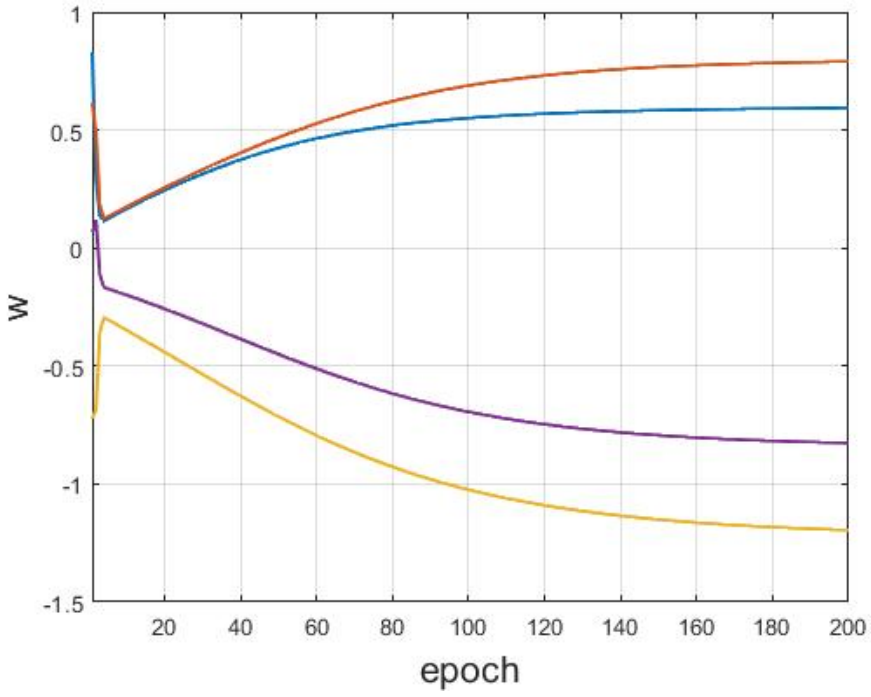

FIGURE 4.15: Evolution of the four components of the weights in GD method in the first 200 epochs in Hyperchaotic Lorenz system learning.

In contrast, the LM method exhibits a rapid evolution of the weights during the training phase, as illustrated in Fig. 4.16. This behavior precludes the exclusion of any portion of the weight dynamics, thereby requiring the inclusion of the entire learning trajectory, including the initial transient phase.

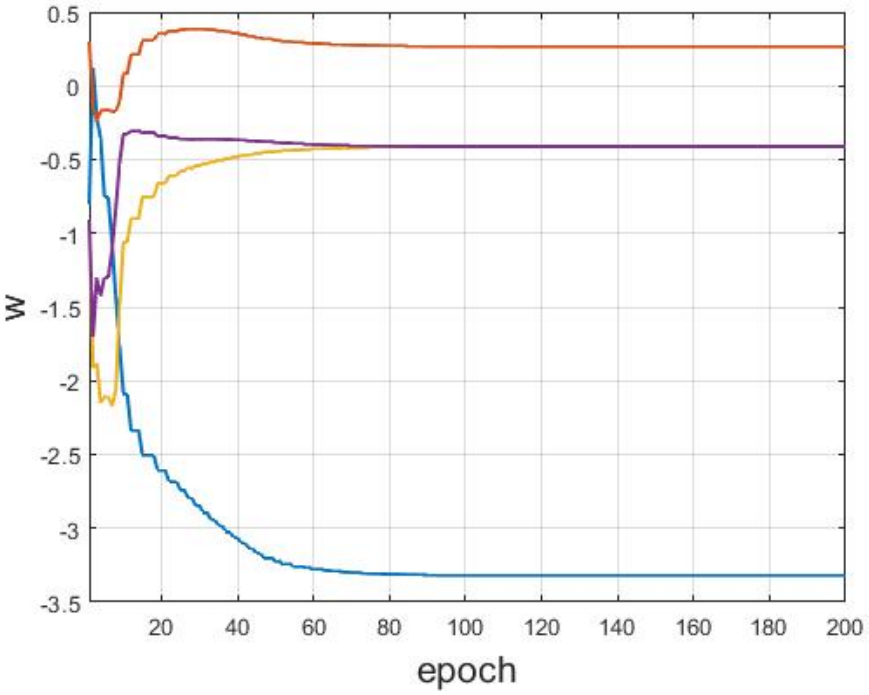

FIGURE 4.16: Evolution four components of the weights in LM method in the first 200 epochs in Hyperchaotic Lorenz system learning.

While this does not present a major obstacle for the auxiliary networks, it complicates the accurate estimation of the final weight values on the first attempt.

The main challenge arises when applying the LM algorithm to small networks with few hidden neurons and a limited number of training patterns. Under such conditions, the LM method already performs efficiently, and the introduction of the learning-on-learning mechanism may not yield significant performance gains. For this reason, the proposed approach is more suitable for networks characterized by a large number of neurons and extensive training sets.

These considerations can be observed in Figs. 4.17 through 4.20, which show the evolution of a selected weight component and the corresponding mean squared error (MSE, in logarithmic scale) as a function of training epochs for the different configurations. Comparisons are presented for extreme cases, $n_h = 2$ and $n_h = 10$, with identical initial weight values assigned to each network pair.

In the case of $n_h = 2$ (Figs. 4.17 and 4.19), minimal differences are observed between the classical and learning-on-learning approaches, both in the convergence behavior of the weights and in the MSE trends. Conversely, for $n_h = 10$ (Figs. 4.18 and 4.20), the benefits of the learning-on-learning strategy become more evident. Notably, both GDL and LML exhibit more pronounced improvements in error reduction—especially after the injection of the predicted optimal weights—compared to their classical counterparts, which tend to converge prematurely to local minima.

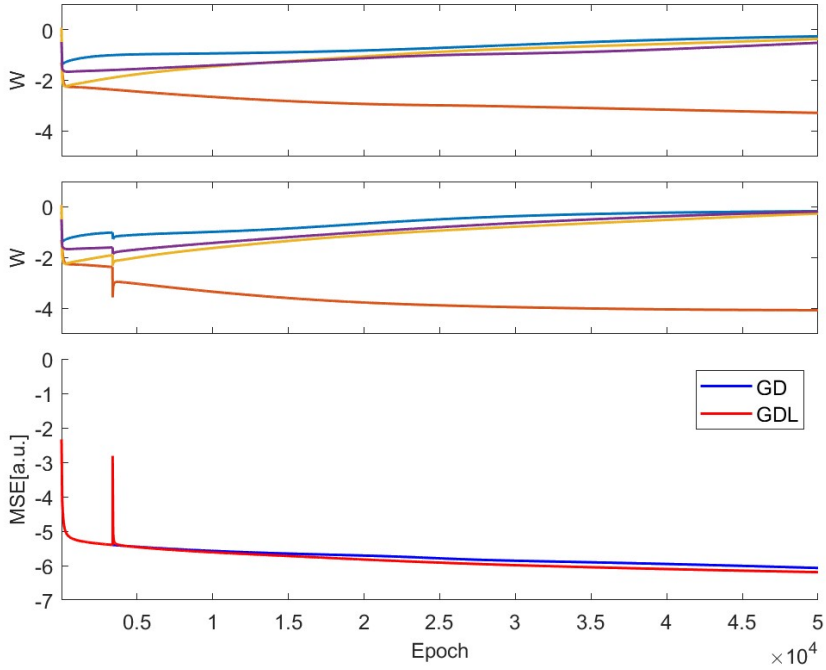

FIGURE 4.17: Comparison of the evolution by epoch of the weights components and of the error in GD and GDL for the case with number of neurons in the hidden layer $n_h = 2$. From the top, the weight profile of GD, the weight profile of GDL and the errors evolution (logarithmic scale) of both the method.

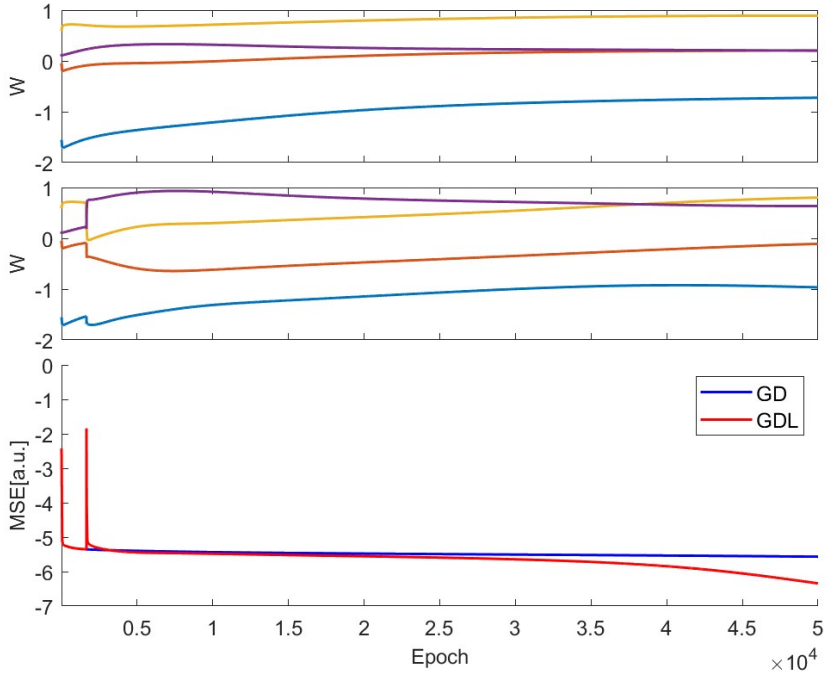

FIGURE 4.18: Comparison of the evolution by epoch of the weights components and of the error in GD and GDL for the case with number of neurons in the hidden layer $n_h = 10$. From the top, the weight profile of GD, the weight profile of GDL and the errors evolution(logarithmic scale) of both the method.

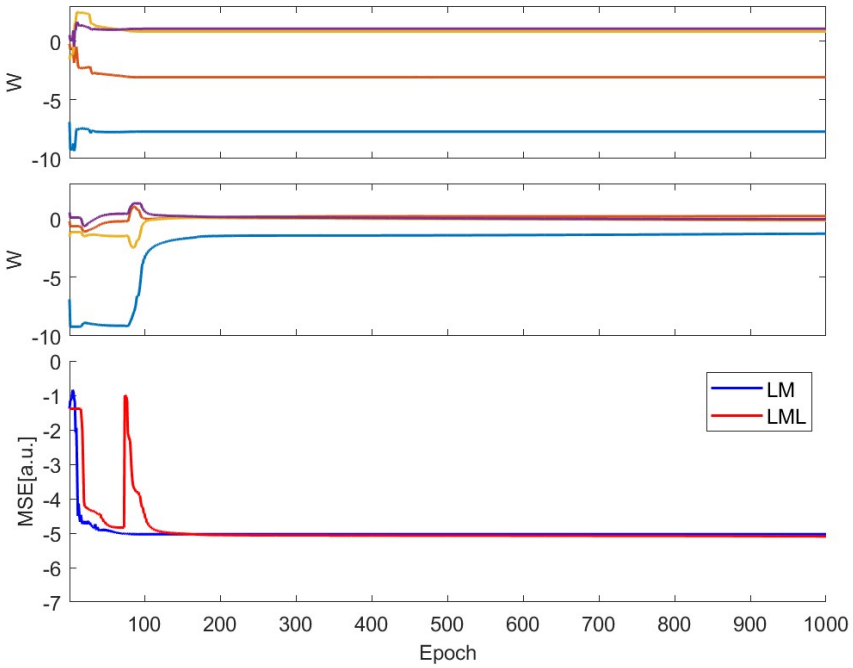

FIGURE 4.19: Comparison of the evolution by epoch of the weights components and of the error in LM and LML for the case with number of neurons in the hidden layer $n_h = 2$. From the top, the weight profile of LM, the weight profile of LML and the errors evolution (logarithmic scale) of both the method.

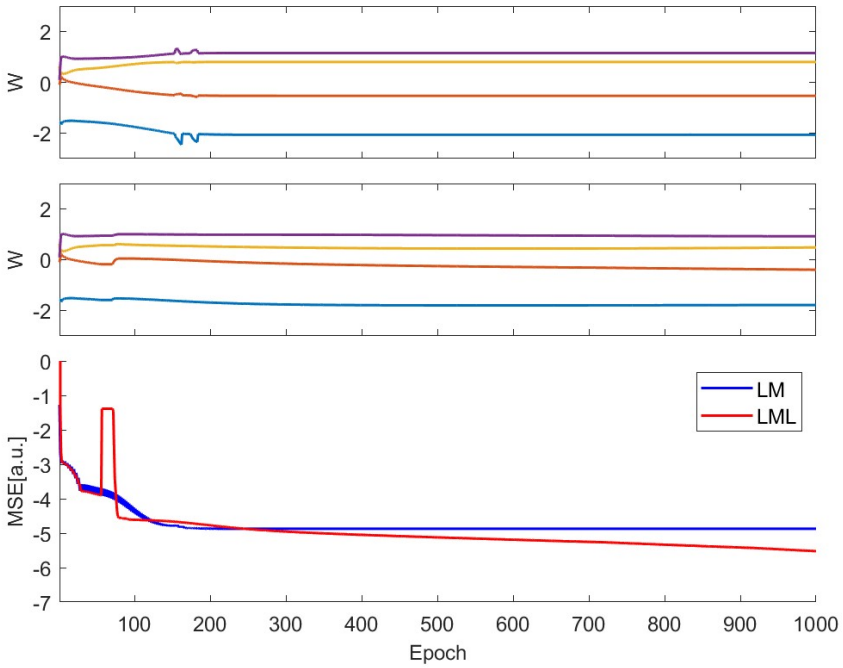

FIGURE 4.20: Comparison of the evolution by epoch of the weights components and of the error in LM and LML for the case with number of neurons in the hidden layer $n_h = 10$. From the top, the weight profile of LM, the weight profile of LML and the errors evolution (logarithmic scale) of both the method.

4.5 Spacecraft rotatory dynamics

This section presents a comparative analysis of the performance achieved by the various methods in a practical scenario involving the prediction of a system that models the attitude dynamics of a rigid body. A representative case is considered, namely a spacecraft subjected to periodic perturbations [73]. The dynamics of the spacecraft are described by the following equations:

$$\begin{cases} \dot{q_0} = -\frac{1}{2}\omega^T \vec{q} \\ \dot{\vec{q}} = \frac{1}{2}\dot{q_0}\omega + \vec{q} \times \omega \\ I\dot{\omega} = \tau - \omega \times I\omega \end{cases} \tag{4.52}$$

where:

- $\mathbf{q} = q_0 + \vec{q}$ is a quaternion representing the orientation of the body with respect to an inertial frame;
- I is the matrix of principal moments of inertia of the body, that is reported in (4.53)
- ω is a vector representing the angular speed
- τ is a vector representing the externally applied torque (a periodic signal on all the three components)

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.63 & 0 \\ 0 & 0 & 0.85 \end{bmatrix} \tag{4.53}$$

The system has been integrated using a time step $\Delta t = 0.01$. The generated signals for $\mathbf{q}$ and ω are reported in Fig. 4.21.

The portion of the signals employed for the learning phase consists of 1500 samples, which are used as training patterns for the networks, while the segment allocated to the testing phase also comprises 1500 samples. As in the previous case, the networks are tasked with performing a one-step-ahead prediction of the time series.

We used the same topology for all the four networks:

- two input neuron which quaternion components represent each one of the seven signal sample at time k;
- one hidden layer with variable number of neurons $[2, \ldots, 10]$;
- two output neuron which quaternion components represent each one of the seven signal sample at time $k + 1$;

In this case study, the number of signals is seven, while the total number of input and output neuron components is eight. To accommodate this structure, only the vector part of the second quaternion neuron is used to represent the three components of the angular velocity vector $\boldsymbol{\omega}$, with the corresponding real part set to zero.

Each network configuration, corresponding to a different number of hidden neurons, has been tested 10 times to ensure statistical significance.

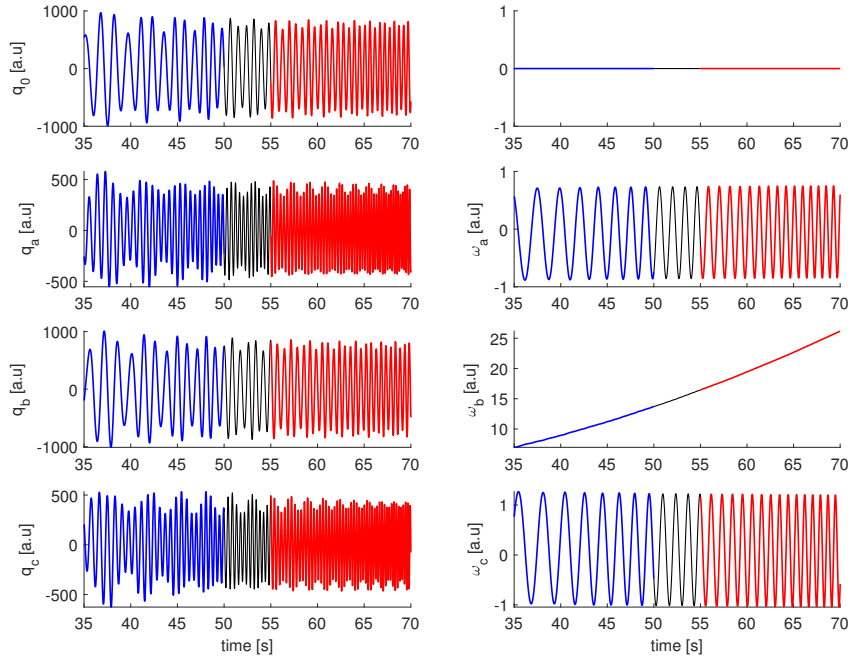

FIGURE 4.21: Signals of the $\mathbf{q}$ and ω vectors. The blue part is the one used for the learning phase, while the red part is the one used for test.

The mean squared error (MSE) between the original and predicted signals, along with the number of epochs required for convergence, are presented in Figures 4.22 and 4.23. These figures provide a comparison between the GD and GDL methods, and between the LM and LML methods, respectively.

Also in this case of study, the best results from the different tests are taken.

The comparison between the actractors of the vectorial part of the attitude quaternion (q_1, q_2, q_3) obtained by the four different methods is shown in Fig.4.24.

In Fig.4.25 it is represented the actractors of the error the networks got from the test phase on the vectorial part of the attitude quaternion $(q_1 - q_{1o}, q_2 - q_{2o}, q_3 - q_{3o})$ of the lorenz system, where (q_1, q_2, q_3) are the exact values of the quaternion components at each time step and (q_{1o}, q_{2o}, q_{3o}) are the output values of the networks.

The MSE per sample on the four part of the attitude quaternion of the spacecraft system is shown in Fig.4.24.

As anticipated, the results demonstrate that, while the final performance levels of all four methods are nearly equivalent, the LM algorithm significantly outperforms GD in terms of training speed, achieving convergence approximately three orders of magnitude faster. Specifically, the maximum number of epochs was limited to 2000 for LM and 150 000 for GD.

Once again, the similarity of the attractors shown in Figures 4.12 and 4.24, as well as the comparison of the error attractor dimensions in 3D space (Figures 4.13 and 4.25), confirms the performance equivalence. Notably, the error attractors are almost entirely contained within a sphere of radius $r = 0.005$

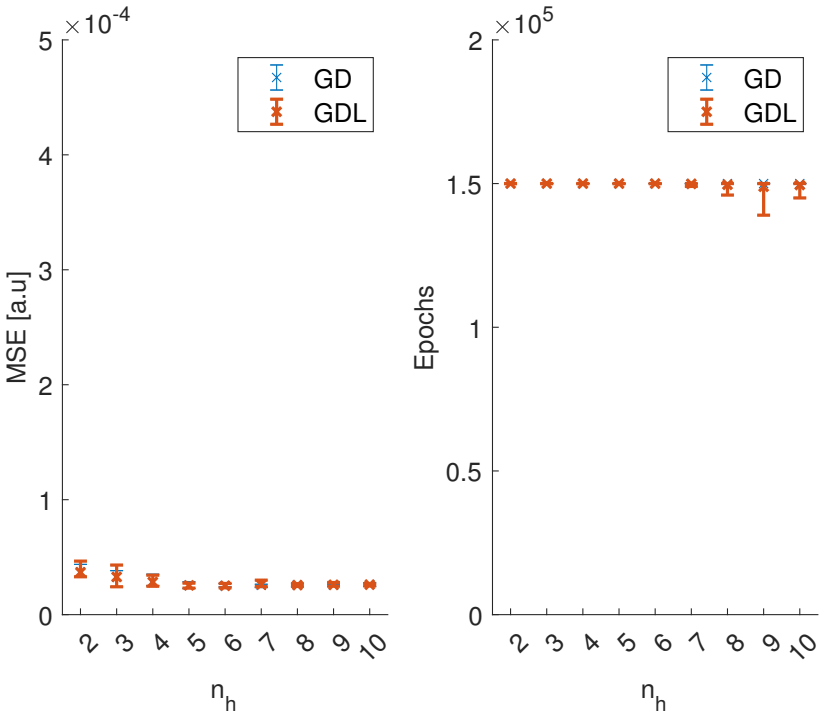

FIGURE 4.22: Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Spacecraft dynamics. N_{hn} is the number of adopted hidden neurons. GD is represented in blue, GDL is represented in orange.

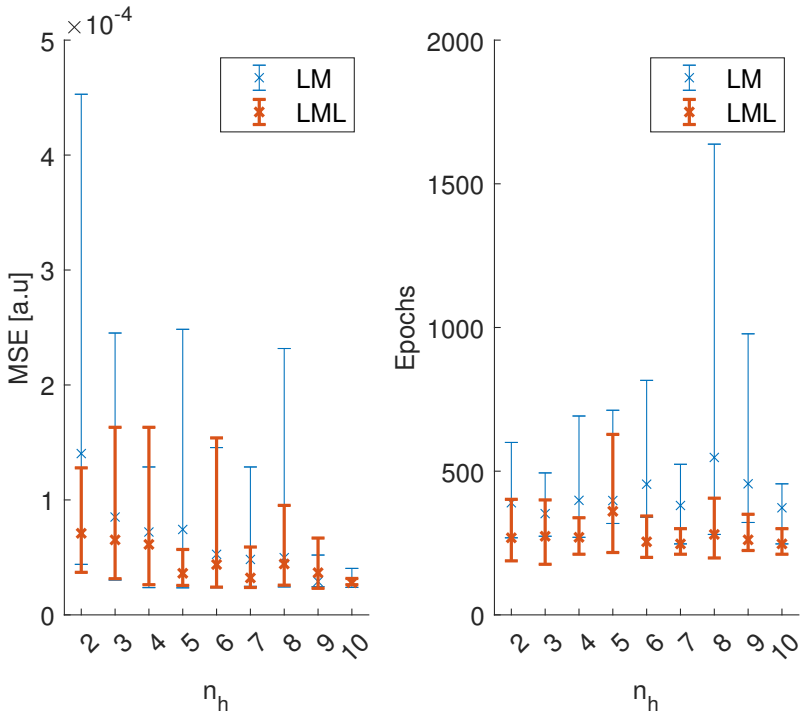

FIGURE 4.23: Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Spacecraft dynamics. N_{hn} is the number of adopted hidden neurons. LM is represented in blue, LML is represented in orange.

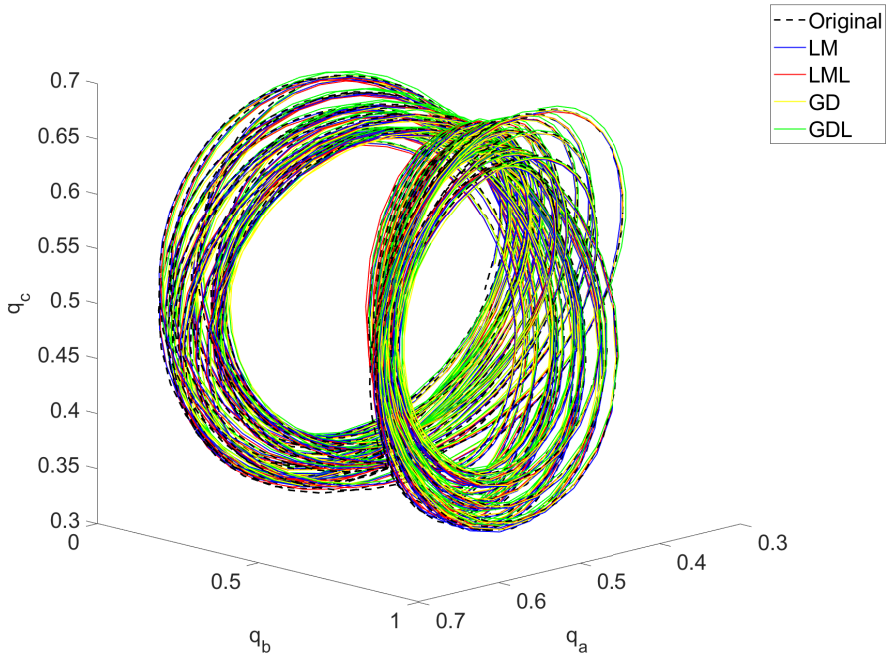

FIGURE 4.24: Actractors of the vectorial part of the attitude quaternion (q_1, q_2, q_3) obtained by the four different methods from w-axis label. The original actractor is represented by black dashed line, LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green.

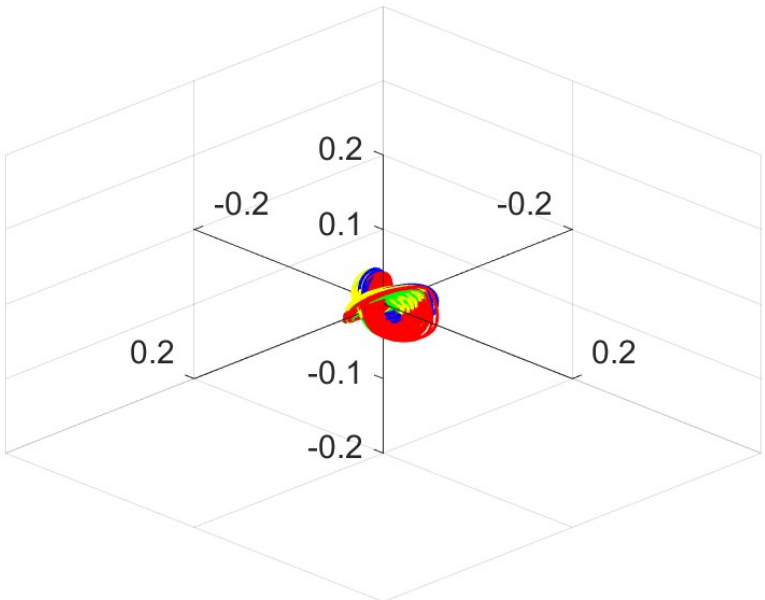

FIGURE 4.25: Actractor of the error between target and output of the networks in test phase on spacecraft system from w-axis label. LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green.

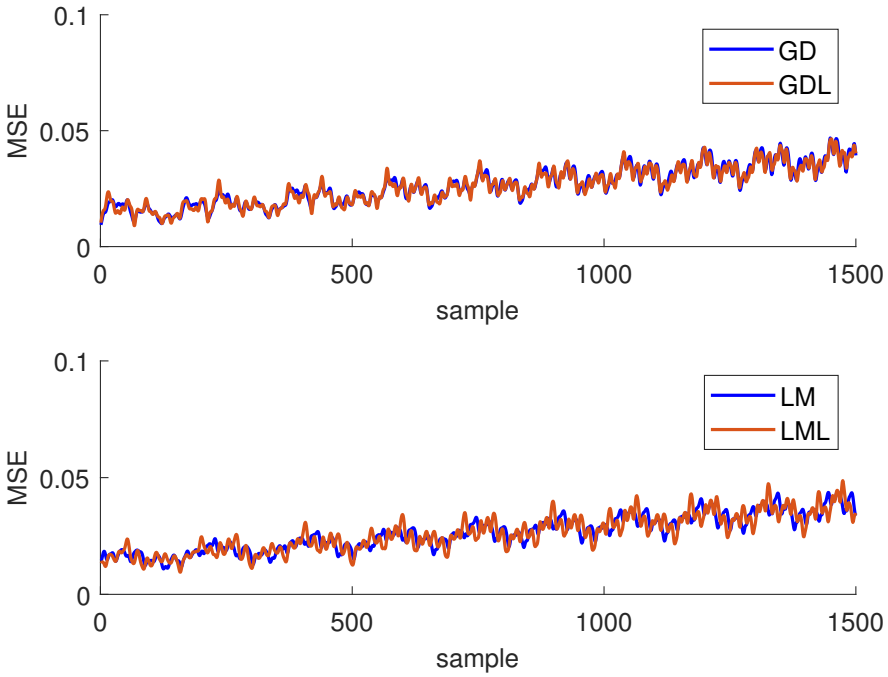

FIGURE 4.26: MSE per sample on the four part of the attitude quaternion (q_0, q_1, q_2, q_3) of the spacecraft system. On the top, the comparison between GD (blue) and GDL (orange), on the bottom the comparison between LM (blue) and LML (orange).

centered at the origin. Focusing on the contribution provided by the learning-on-learning approach, a modest improvement in accuracy is observed, although the gain in learning speed is significantly more evident. This is clearly illustrated by the vertical orange bars in Figures 4.22 and 4.23, which represent the distribution of epochs required as a function of the number of hidden neurons: these bars are not only narrower but also consistently lower compared to those representing the standard methods.

The proposed learning-on-learning strategy, when integrated with quaternion representations in hypercomplex neural networks, provides a clear advantage in terms of convergence efficiency. Despite achieving comparable prediction accuracy across all methods, the learning-on-learning approach significantly reduces the average number of epochs required for training.

In particular, in the case of the Levenberg–Marquardt (LM) algorithm, the benefits of auxiliary network-based predictions become more pronounced as the number of hidden neurons increases. This highlights that the proposed method is especially effective for larger network configurations.

In conclusion, these results demonstrate the potential of the Learnign-on-Learning strategy to reduce training time in HMLPs without increasing model complexity. The proposed method enhances convergence speed, thereby facilitating the application of these models to nonlinear system identification tasks. This makes them especially suitable for deployment in Real-Time experimental contexts, such as modern multicore embedded systems.

Chapter 5

Conclusions

This work contributes to the development of control systems for ITER, JT-60SA, DTT and future Tokamaks by examining diverse methodological approaches and offering insights for future advancements in control and data acquisition for magnetic confinement fusion plants. The thesis progresses through the selection of suitable hardware–software combinations for real-time control applications, the study of complex power supply control systems for coil energization, and the exploration of innovative techniques involving neural networks for real-time feedforward prediction.

The initial results aimed to evaluate the performance impact of different hardware–software configurations in real-time control and data acquisition loops. The findings clearly demonstrate that all tested platforms are capable of sustaining real-time control loops of approximately 10 kHz, even when operating under a real-time operating system, provided that appropriate software tuning is applied. Such solutions remain suitable for soft-to-firm real-time tasks, while simultaneously benefiting from enhanced computational resources.

Secondly, the collaboration with the Fusion for Energy Broader Approach Unit enabled the investigation of modeling and control strategies for poloidal and toroidal field coil power supply systems, which play a fundamental role in plasma shaping and stability. Building upon the DSP control code provided by manufacturers, a new simulation tool was developed to support the validation of novel control strategies and parameter configurations. This activity not only facilitated the acquisition of advanced expertise in the electrical domain but also fostered a deeper understanding of control methodologies directly applicable to the studied systems.

In parallel, standard multi-layer perceptrons were employed in a data-driven framework to model the multidimensional dependencies between the JT-60SA toroidal field quench detection system and the poloidal field coils.

This approach effectively mitigated spurious effects and enabled reliable signal prediction in real-time operation. On the other hand, the thesis also explored innovative machine learning methodologies, such as hypercomplex neural networks and learning-on-learning algorithms, aimed at advancing predictive control and diagnostic enhancement. These strategies were investigated demonstrating their potential to strengthen system performance, reliability, and robustness in complex real-time environments. Taken together, these two lines of investigation highlight how classical neural network architectures can successfully address specific prediction challenges in operational systems, while more advanced paradigms offer promising perspectives for future developments in real-time learning for plasma fusion control.

Overall, the results achieved in this work contribute significantly to advancing research in this complex domain, and can be considered as potential candidates for future improvements in the control and diagnostic systems of Tokamaks.

Appendix A

Fondamentals on Quaternion algebra

A quaternion $\mathbf{q}$ is an extension of complex numbers that consists of four real-valued components (q_a, q_b, q_c, q_d) and three imaginary units $(\imath, \jmath, \kappa)$, which serve as the basis vectors of a three-dimensional orthonormal reference frame.

$$\mathbf{q} = \mathrm{q}_a + \mathrm{q}_b\imath + \mathrm{q}_c\jmath + \mathrm{q}_d\kappa \tag{A.1}$$

The role of the imaginary part in quaternion algebra is the same as the unit $i = \sqrt{-1}$ in complex algebra, and the quaternion imaginary components satisfy the following expressions:

$$\imath^2 = \jmath^2 = \kappa^2 = -1$$

$$\jmath\kappa = -\kappa\jmath = \imath$$

$$\kappa\imath = -\imath\kappa = \jmath$$

$$\imath\jmath = -\jmath\imath = \kappa$$

A quaternion can be seen also as the sum of a real number q_a and a vector $\vec{\mathbf{q}}$:

$$\mathbf{q} = \mathrm{q}_a + \vec{\mathbf{q}}$$

In matrix form it is expressed as:

$$\mathrm{q} = [\mathrm{q}_a, \mathrm{q}_b, \mathrm{q}_c, \mathrm{q}_d]^T$$

The fundamental operators for the quaternion algebra are:

- Conjugate:
$$\mathbf{q}^* = \mathrm{q}_a - \vec{\mathbf{q}} = \mathrm{q}_a - \mathrm{q}_b\imath - \mathrm{q}_c\jmath - \mathrm{q}_d\kappa$$
- Modulus:
$$|\mathbf{q}^*| = \sqrt{\mathrm{q}_a^2 + \mathrm{q}_b^2 + \mathrm{q}_c^2 + \mathrm{q}_d^2}$$

- Sum of quaternions:

$$\mathbf{q} + \mathbf{p} = (\mathrm{q}_a + \mathrm{p}_a) + (\mathrm{q}_b + \mathrm{p}_b)\imath + (\mathrm{q}_c + \mathrm{p}_c)\jmath + (\mathrm{q}_d + \mathrm{p}_d)\kappa$$

- Product of quaternions:

$$\mathbf{q} \otimes \mathbf{p} = (\mathrm{q}_a + \vec{\mathbf{q}}) \otimes (\mathrm{p}_a + \vec{\mathbf{p}}) = \mathrm{q}_a\mathrm{p}_a - \vec{\mathbf{q}} \cdot \vec{\mathbf{p}} + \mathrm{q}_a\vec{\mathbf{p}} + \mathrm{p}_a\vec{\mathbf{q}} + \vec{\mathbf{q}} \times \vec{\mathbf{p}}$$

  where "$\cdot$" is the scalar product and "$\times$" represents the vector product, as defined in vector algebra. The quaternion product is only associative and distributive but not commutative.

- The inverse of a quaternion is derived from the following formula:

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\mathbf{q}^* \otimes \mathbf{q}}$$

- The quaternion involution [74] geometrically describes a 3-D rotation of the vector part of $\mathbf{q}$ by an angle 2θ about the pure unit vectors $\imath$, $\jmath$ and κ, and it is expressed by the following equations:

$$\begin{aligned} \mathbf{q}^{\imath} &= -\imath\mathbf{q}\imath = \mathrm{q}_a \ \ + \imath\mathrm{q}_b - \jmath\mathrm{q}_c - \kappa\mathrm{q}_d, \\ \mathbf{q}^{\jmath} &= -\jmath\mathbf{q}\jmath = \mathrm{q}_a - \imath\mathrm{q}_b + \jmath\mathrm{q}_c - \kappa\mathrm{q}_d, \\ \mathbf{q}^{\kappa} &= -\kappa\mathbf{q}\kappa = \mathrm{q}_a - \imath\mathrm{q}_b - \jmath\mathrm{q}_c + \kappa\mathrm{q}_d, \end{aligned} \tag{A.2}$$

A.0.1 Hamiltonian-Real Calculus

We will now outline the fundamentals of the so-called Hamiltonian-Real (HR) calculus, which establishes a formal link between compact hyper-complex algebra and the algorithmically non tractable real algebra. This allows us to implement the quaternion-valued operators involved in the learning algorithms in terms of real-valued vectors [75].

Consider a vector q of quaternions:

$$\mathrm{q} = [\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_N]^T \in \mathbb{H}^N.$$

It can be written as:

$$\mathrm{q} = \mathrm{q}_a + \imath\mathrm{q}_b + \jmath\mathrm{q}_c + \kappa\mathrm{q}_d \in \mathbb{H}^N,$$

where $\mathrm{q}_a, \mathrm{q}_b, \mathrm{q}_c, \mathrm{q}_d$ are real-valued vectors in $\mathbb{R}^N$.

From Eqs. (A.2), we can define the relation between the augmented quaternion vector:

$$\overset{\mathbb{H}}{\mathrm{q}} \equiv \begin{bmatrix} \mathrm{q} \\ \mathrm{q}^{\imath} \\ \mathrm{q}^{\jmath} \\ \mathrm{q}^{\kappa} \end{bmatrix} \in \mathbb{H}^{4N} \tag{A.3}$$

and the dual-quadrivariate real vector:

$$\overset{\mathbb{R}}{\mathrm{q}} \equiv \begin{bmatrix} \mathrm{q}_a \\ \mathrm{q}_b \\ \mathrm{q}_c \\ \mathrm{q}_d \end{bmatrix} \in \mathbb{R}^{4N}$$

that is:

$$\overset{\mathbb{H}}{\mathrm{q}} = \mathbb{J}_N \overset{\mathbb{R}}{\mathrm{q}} \tag{A.4}$$

as:

$$\begin{bmatrix} \mathrm{q} \\ \mathrm{q}^{\imath} \\ \mathrm{q}^{\jmath} \\ \mathrm{q}^{\kappa} \end{bmatrix} = \begin{bmatrix} \mathbb{I}_N & \imath\mathbb{I}_N & \jmath\mathbb{I}_N & \kappa\mathbb{I}_N \\ \mathbb{I}_N & \imath\mathbb{I}_N & -\jmath\mathbb{I}_N & -\kappa\mathbb{I}_N \\ \mathbb{I}_N & -\imath\mathbb{I}_N & \jmath\mathbb{I}_N & -\kappa\mathbb{I}_N \\ \mathbb{I}_N & -\imath\mathbb{I}_N & -\jmath\mathbb{I}_N & \kappa\mathbb{I}_N \end{bmatrix} \begin{bmatrix} \mathrm{q}_a \\ \mathrm{q}_b \\ \mathrm{q}_c \\ \mathrm{q}_d \end{bmatrix}$$

with:

$$\mathbb{J}_N \equiv \begin{bmatrix} \mathbb{I}_N & \imath\mathbb{I}_N & \jmath\mathbb{I}_N & \kappa\mathbb{I}_N \\ \mathbb{I}_N & \imath\mathbb{I}_N & -\jmath\mathbb{I}_N & -\kappa\mathbb{I}_N \\ \mathbb{I}_N & -\imath\mathbb{I}_N & \jmath\mathbb{I}_N & -\kappa\mathbb{I}_N \\ \mathbb{I}_N & -\imath\mathbb{I}_N & -\jmath\mathbb{I}_N & \kappa\mathbb{I}_N \end{bmatrix}$$

where the subscript of $\mathbb{J}$ indicates the size of each $\mathbb{I}_N$ that are the $N \times N$ identity matrix.

Let us consider the following rule:

$$\mathbb{J}_N^{\dagger}\mathbb{J}_N = \mathbb{J}_N\mathbb{J}_N^{\dagger} = 4\mathbb{I}_{4N} \tag{A.5}$$

from which we can derive the inverse of (A.4):

$$\overset{\mathbb{R}}{\mathrm{q}} = \frac{1}{4}\mathbb{J}_N^{\dagger}\overset{\mathbb{H}}{\mathrm{q}} \tag{A.6}$$

where $\mathbb{J}_N^{\dagger}$ denotes the Hermitian transpose of $\mathbb{J}_N$.

Let us define a function $f : \mathbb{H}^N \to \mathbb{R}$ which can be formalized in three equivalent forms [75]:

$$f(\mathrm{q}) \Leftrightarrow f(\overset{\mathbb{H}}{\mathrm{q}}) f(\mathrm{q}, \mathrm{q}^i, \mathrm{q}^j, \mathrm{q}^k) \Leftrightarrow f(\overset{\mathbb{R}}{\mathrm{q}}) \coloneqq f(\mathrm{q}_a, \mathrm{q}_b, \mathrm{q}_c, \mathrm{q}_d) \tag{A.7}$$

If we define the following derivatives:

$$\begin{aligned} \frac{\partial f}{\partial \mathrm{q}} &\coloneqq \left[\frac{\partial f}{\partial \mathrm{q}_1}, \frac{\partial f}{\partial \mathrm{q}_2}, \ldots, \frac{\partial f}{\partial \mathrm{q}_N}\right], \\ \frac{\partial f}{\partial \overset{\mathbb{H}}{\mathrm{q}}} &\coloneqq \left[\frac{\partial f}{\partial \mathrm{q}}, \frac{\partial f}{\partial \mathrm{q}^i}, \frac{\partial f}{\partial \mathrm{q}^j}, \frac{\partial f}{\partial \mathrm{q}^k}\right], \\ \frac{\partial f}{\partial \overset{\mathbb{R}}{\mathrm{q}}} &\coloneqq \left[\frac{\partial f}{\partial \mathrm{q}_a}, \frac{\partial f}{\partial \mathrm{q}_b}, \frac{\partial f}{\partial \mathrm{q}_c}, \frac{\partial f}{\partial \mathrm{q}_d}\right], \end{aligned} \tag{A.8}$$

we obtain, from the chain rule, that:

$$\frac{\partial f}{\partial \overset{\mathbb{H}}{\mathrm{q}}} = \frac{1}{4}\frac{\partial f}{\partial \overset{\mathbb{R}}{\mathrm{q}}}\mathbb{J}_N^{\dagger} \Leftrightarrow \frac{\partial f}{\partial \overset{\mathbb{R}}{\mathrm{q}}} = \frac{\partial f}{\partial \overset{\mathbb{H}}{\mathrm{q}}}\mathbb{J}_N \tag{A.9}$$

Let us consider the real gradient and the augmented quaternion gradient [75], respectively defined as:

$$\nabla_{\overset{\mathbb{R}}{\mathrm{q}}} f \coloneqq \left(\frac{\partial f}{\partial \overset{\mathbb{R}}{\mathrm{q}}}\right)^T, \quad \nabla_{\overset{\mathbb{H}}{\mathrm{q}}} f \coloneqq \left(\frac{\partial f}{\partial \overset{\mathbb{H}}{\mathrm{q}}}\right)^{\dagger} \tag{A.10}$$

The relations in (A.9) can be rewritten as:

$$\underset{\mathrm{q}}{\nabla_{\mathbb{H}}} f = \frac{1}{4}\mathbb{J}_N \underset{\mathrm{q}}{\nabla_{\mathbb{R}}} f \Leftrightarrow \underset{\mathrm{q}}{\nabla_{\mathbb{R}}} f = \mathbb{J}_N^{\dagger} \underset{\mathrm{q}}{\nabla_{\mathbb{H}}} f \tag{A.11}$$

This shows that $\underset{\mathrm{q}}{\nabla_{\mathbb{R}}} f$ and $\underset{\mathrm{q}}{\nabla_{\mathbb{H}}} f$ are related by a simple invertible linear transformation.

Bibliography

[1] F. Chen, *Introduction to Plasma Physics and Controlled Fusion*, 3rd. Springer, 2016.

[2] J. K. Shultis and R. E. Faw, *Fundamentals of Nuclear Science and Engineering*. New York - Basel: Marcel Dekker Inc., 2002.

[3] M. Ariola and A. Pironti, *Magnetic Control of Tokamak Plasmas* (Advances in Industrial Control), 2nd ed. Cham, Switzerland: Springer, Cham, 2016, ISBN: 978-3-319-29888-7. DOI: 10.1007/978-3-319-29890-0. [Online]. Available: https://link.springer.com/book/10.1007/978-3-319-29890-0.

[4] J. Connor and O. Pogutse, "Influence of an x-point on the lh transition power threshold," *Plasma physics and controlled fusion*, vol. 43, no. 3, p. 281, 2001.

[5] M. Kikuchi, K. Lackner, and M. Q. Tran, *Fusion Physics*. Vienna: IAEA, 2012.

[6] Argonne Leadership Computing Facility, "Global simulation of plasma microturbulence at the petascale & beyond (optimizing the gtc code for blue gene/q)," Argonne National Laboratory, Tech. Rep. ANL/ALCF/ESP-13/15, 2013, ALCF-2 Early Science Program Technical Report. DOI: 10.2172/1079772.

[7] ITER Organization, *Iter technical basis*, https://www.iter.org/mach/TechnicalBasis, Accessed 2025.

[8] W. Horton, "Drift waves and transport," *Reviews of Modern Physics*, vol. 71, no. 3, p. 735, 1999.

[9] I. P. E. Groups, "Chapter 1: Overview and summary," *Nuclear Fusion*, vol. 39, no. 12, p. 2137, 1999.

[10] A. e. a. Sips, "Overview of progress in european medium-sized tokamaks towards an integrated plasma-edge/wall solution," *Nuclear Fusion*, vol. 53, no. 10, p. 104 015, 2013.

[11] J. e. a. Ferron, "Real time equilibrium reconstruction for tokamak discharge control," *Nuclear Fusion*, vol. 38, no. 8, p. 1055, 1998.

[12] A. E. Costley, J. Hugill, and P. F. Buxton, "On the power and size of tokamak fusion pilot plants and reactors," *Nuclear Fusion*, vol. 55, no. 3, p. 033 001, 2015.

[13] I. H. Hutchinson, *Principles of Plasma Diagnostics*. Cambridge University Press, 2005.

[14] J. Journeaux and A. Wallander, "Plant control design handbook," ITER Organization, Technical Report, 2020.

[15] ITER Organization, *In a few lines*, `https://www.iter.org/few-lines`, Accessed July 2025, 2025.

[16] EUROfusion, *Jt-60sa*, `https://euro-fusion.org/devices/jt-60sa/`, Accessed July 2025, 2025.

[17] ENEA / DTT Project, *Dtt – divertor tokamak test facility*, `https://www.dtt-project.it/`, Accessed July 2025, 2025.

[18] Fusion for Energy (F4E), *Marte2 v1.0.0 documentation: Overview*, `https://vcis.f4e.europa.eu/marte2-docs/master/html/overview.html`, Accessed 2025, 2018.

[19] A. Neto, F. Sartori, F. Piccolo, A. Barbalace, R. Vitelli, and H. Fernandes, "Linux real time framework for fusion devices," vol. 84, pp. 1408,1411, 2009.

[20] A. Neto et al., "Marte: A multiplatform real-time framework," *IEEE Transactions on Nuclear Science*, vol. 57, no. 2, pp. 479,486, 2010.

[21] T. Bellizio, G. D. Tommasi, N. Risoli, R. Albanese, and A. Neto, "A marte based simulator for the jet vertical stabilization system," *Fusion Engineering and Design*, vol. 86, pp. 1026,1029, 2011.

[22] N. Ferron et al., "Implementation and review of the axisymmetric equilibrium system of rfx-mod2 within the marte2 framework," *Electronics*, vol. 11, no. 17, p. 2751, 2022.

[23] Fusion for Energy, *Marte2 datasource & brokers*, `https://vcis.f4e.europa.eu/marte2-docs/master/html/core/gams/datasource.html`, Accessed 15 August 2022.

[24] Fusion for Energy, *Marte2 gam*, `https://vcis.f4e.europa.eu/marte2-docs/master/html/core/gams/gam.html`, Accessed 15 August 2022.

[25] G. Avon, A. Buscarino, A. C. Neto, and F. Sartori, "Marte2 embedded signal processing unit for the iter magnetics diagnostics," in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, Toronto, ON, Canada, 2021, pp. 1,6. DOI: `10.1109/IECON48115.2021.9589325`.

[26] S. Haykin, *Neural Networks and Learning Machines*, 3rd. Prentice Hall, 2009, ISBN: 9780131471399.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: `10.1038/nature14539`.

[28] K. L. van de Plassche et al., "Fast modeling of turbulent transport in fusion plasmas using neural networks," *Physics of Plasmas*, vol. 27, no. 2, 2020.

[29] F. Felici, J. Citrin, A. A. Teplukhina, and O. Sauter, "Real-time feedback control of β_p based on deep reinforcement learning on east," *Physics of Plasmas*, vol. 25, no. 5, p. 052 503, 2018. DOI: 10.1063/1.5016842.

[30] A. Tenaglia et al., "An interpretable isoflux-based observer for plasma shape control errors in tokamaks," *Fusion Engineering and Design*, vol. 207, p. 114 618, 2024. DOI: 10.1016/j.fusengdes.2024.114618.

[31] S. Mangini, F. Tacchino, D. Gerace, C. Macchiavello, and D. Bajoni, "Quantum computing model of an artificial neuron with continuously valued input data," *Machine Learning: Science and Technology*, vol. 1, no. 4, 2020. DOI: 10.1088/2632-2153/abaf98. [Online]. Available: https://doi.org/10.1088/2632-2153/abaf98.

[32] P. Arena, L. Fortuna, G. Muscato, and M. G. Xibilia, *Neural Networks in Multidimensional Domains: Fundamentals and New Trends in Modelling and Control* (Lecture Notes in Control and Information Sciences), 1st. London, UK: Springer London, 1998, vol. 234, p. 169, ISBN: 978-1-85233-006-4. DOI: 10.1007/BFb0047683.

[33] A. Buscarino, C. Famoso, D. Comminiello, and L. Patané, "Quaternion neural networks for multidimensional applications: An overview," pp. 1–4, 2023. DOI: 10.1109/ICECS58634.2023.10382881.

[34] D. Comminiello, M. Lella, S. Scardapane, and A. Uncini, "Quaternion convolutional neural networks for detection and localization of 3d sound events," pp. 8533–8537, 2019. DOI: 10.1109/ICASSP.2019.8682711.

[35] P. Arena, L. Fortuna, G. Muscato, and M. G. Xibilia, "Multilayer perceptrons to approximate quaternion valued functions," *Neural Networks*, vol. 10, no. 2, pp. 335–342, 1997. DOI: 10.1016/S0893-6080(96)00048-2.

[36] C. Galperti et al., "Overview of the tcv digital real-time plasma control system and its applications," *Fusion Engineering and Design*, vol. 208, p. 114 640, 2024.

[37] R. Kumar et al., "Real-time feedback control system for aditya-u horizontal plasma position stabilisation," *Fusion Engineering and Design*, vol. 165, p. 112 218, 2021.

[38] J. A. Snipes et al., "Iter plasma control system final design and preparation for first plasma," *Nuclear Fusion*, vol. 61, no. 10, p. 106 036, 2021.

[39] G Manduchi, A Rigonii, T. Fredian, J. Stillerman, A Neto, and F Sartori, "Marte2 and mdsplus integration for a comprehensive fast control and data acquisition system," *Fusion Engineering and Design*, vol. 161, p. 111 892, 2020.

[40] A. Barbalace et al., "Performance comparison of epics ioc and marte in a hard real-time control application," *IEEE Transactions on Nuclear Science*, vol. 58, no. 6, pp. 3162–3166, 2011.

[41] STMicroelectronics, *Stm32h723xg/stm32h733xg advanced arm®-based 32-bit mcus*, Reference Manual, [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32h723zg.html, 2023.

[42] A. Inc., *Picozed: System-on-module based on xilinx zynq-7000 all programmable soc*, [Online]. Available: https://www.avnet.com/americas/products/avnet-boards/avnet-board-families/picozed/, [Accessed: Jul. 25, 2025].

[43] Xilinx, *Zynq-7000 soc data sheet: Overview*, Technical Report, [Online]. Available: https://docs.xilinx.com/, 2023.

[44] G. Ferro et al., "Embedded implementation of a real-time switching controller on a robotic arm," in *Proc. 20th IEEE-NPSS Real Time Conf.(RT)*, 2016.

[45] A. R. Garola et al., "A zynq-based flexible adc architecture combining real-time data streaming and transient recording," *IEEE Transactions on Nuclear Science*, vol. 68, no. 2, pp. 245–249, 2021. DOI: 10.1109/TNS.2020.3035146.

[46] D.-T. Solutions, *Intelligent data acquisition boards and systems*, Available at https://www.d-tacq.com/, accessed August 26, 2025.

[47] Y. Kamada et al., "Completion of jt-60sa construction and contribution to iter," *Nuclear Fusion*, vol. 62, no. 4, 2022, Art. no. 042002.

[48] H. Murakami et al., "Completion of central solenoid for jt-60sa," *IEEE Transactions on Applied Superconductivity*, vol. 31, no. 5, 2021, Art. no. 4201005.

[49] M. Wanner, "Plant integration document, version 4.5," pp. 207–224, 2022.

[50] K. Shimada et al., "A design study of stable coil current control method for back-to-back thyristor converter in jt-60sa," *J. Plasma Fusion Res. Ser.*, vol. 9, p. 163, 2010.

[51] L. Novello et al., "Overview of the new magnet power supply systems of jt-60sa procured by eu," *Fusion Engineering and Design*, vol. 98, pp. 1122–1126, 2015.

[52] L. Bottura, "Magnet quench 101," *arXiv preprint*, vol. arXiv:1401.3927, 2014. [Online]. Available: https://arxiv.org/pdf/1401.3927.

[53] B. Lacroix et al., "Quench propagation and quench detection in the tf system of jt-60sa," *Fusion Engineering and Design*, vol. 88, no. 6-8, pp. 559–562, 2013.

[54] L. Zani, Q. Le Coz, K. Hamada, M. Parody-Guzman, B. Lacroix, and L. Novello, "Exploring operation limits of jt-60sa cryomagnet operation: Integrated commissioning database use for modeling analysis," *IEEE Transactions on Applied Superconductivity*, vol. 35, no. 5, pp. 1–5, 2025, Art. no. 4202405.

[55] L. Ljung, "System identification," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, Wiley, 2007.

[56] A. G. Spinosa, A. Buscarino, L. Fortuna, M. Iafrati, and G. Mazzitelli, "Data-driven order reduction in hammerstein–wiener models of plasma dynamics," *Engineering Applications of Artificial Intelligence*, vol. 100, p. 104 180, 2021.

[57] L. Fortuna, V. Marchese, A. Rizzo, and M. G. Xibilia, "A neural networks based system for post pulse fault detection and disruption data validation in tokamak machines," in *1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, IEEE, 1999, pp. 563–566.

[58] P. Auer, H. Burgsteiner, and W. Maass, "A learning rule for very simple universal approximators consisting of a single layer of perceptrons," *Neural Networks*, vol. 21, no. 5, pp. 786–795, 2008.

[59] A. Elnady and M. M. A. Salama, "Mitigation of voltage disturbances using adaptive perceptron-based control algorithm," *IEEE Transactions on Power Delivery*, vol. 20, no. 1, pp. 309–318, 2005.

[60] X. Tang and et al., "Deep learning-assisted jamming mitigation with movable antenna array," *IEEE Transactions on Vehicular Technology*, 2025, Accepted for future publication.

[61] S. Lawrence and C. L. Giles, "Overfitting and neural networks: Conjugate gradient and backpropagation," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000*, vol. 1, IEEE, 2000, pp. 114–119.

[62] Z. Zhang et al., "Fixed-time pinning common synchronization and adaptive synchronization for delayed quaternion-valued neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 2, pp. 2276–2289, 2022.

[63] J. Pöppelbaum and A. Schwung, "Time series compression using quaternion valued neural networks and quaternion backpropagation," *arXiv preprint arXiv:2403.11722*, 2024. arXiv: 2403.11722 [cs.LG].

[64] E. Bayro-Corrochano, "A survey on quaternion algebra and geometric algebra applications in engineering and computer science 1995–2020," *IEEE Access*, vol. 9, pp. 104 326–104 355, 2021. DOI: 10.1109/ACCESS.2021.3099677.

[65] A. Cariow and G. Cariowa, "Fast algorithms for quaternion-valued convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 457–462, Jan. 2021. DOI: 10.1109/TNNLS.2020.2979682.

[66] C.-A. Popa, "Levenberg-marquardt learning algorithm for quaternion-valued neural networks," pp. 272–278, 2016. DOI: 10.1109/SYNASC.2016.050.

[67] L. Fortuna, S. Graziani, M. Lo Presti, and G. Muscato, "Improving back-propagation learning using auxiliary neural networks," *International Journal of Control*, vol. 55, no. 4, pp. 793–807, 1992.

[68] M. D. Mauro, C. Famoso, G. Puglisi, L. Fortuna, and A. Buscarino, "Quaternion neural networks towards real-time image processing," pp. 1–5, 2025. DOI: 10.1109/ISCAS56072.2025.11044298.

[69] M. Bucolo, A. Buscarino, L. Fortuna, and G. Puglisi, "Learning-on-learning approach for modeling," pp. 1–6, 2022.

[70] A. Buscarino, C. Famoso, L. Fortuna, and G. Puglisi, "Hypercomplex multilayer perceptron for planetary orbits prediction," pp. 1427–1431, 2023.

[71] A. Buscarino, C. Famoso, L. Fortuna, and G. Puglisi, "Fast hypercomplex neural networks for modeling venus planetary orbit," pp. 1–4, 2023.

[72] E. Turan, S. Speretta, and E. Gill, "Autonomous navigation for deep space small satellites: Scientific and technological advances," *Acta Astronautica*, vol. 193, 2022.

[73] L. Fortuna, G. Muscato, and M. Xibilia, "A comparison between hmlp and hrbf for attitude control," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 318–328, 2001, ISSN: 1045-9227. DOI: 10.1109/72.914526.

[74] T. A. Ell and S. J. Sangwine, "Quaternion involutions and anti-involutions," *Computers & Mathematics with Applications*, vol. 53, no. 1, pp. 137–143, 2007, ISSN: 0898-1221. DOI: 10.1016/j.camwa.2006.10.011.

[75] D. Xu, Y. Xia, and D. P. Mandic, "Optimization in quaternion dynamic systems: Gradient, hessian, and learning algorithms," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 249–261, 2016. DOI: 10.1109/TNNLS.2015.2414942.

List of Figures

1.1 *The flight of Icaro to the sun. Author: unknown.* 1
1.2 Graphical representation of the deuterium-tritium (D-T) reaction. *Source:* [3] 3
1.3 Illustration of the Tokamak concept. The toroidal magnetic field is indicated with a blue arrow. Poloidal field (green arrows) are employed to control the plasma position and shape. *Source:* [6] 4
1.4 Poloidal D-shaped cross-section of a Tokamak machine, partitioned into regions occupied by the plasma (Ω_p) and the conducting structure (Ω_c). *Source:* [3] 5
1.5 JT-60SA Tokamak in QST Tokamak building. 8
1.6 Architectural graph of a multilayer perceptron with two hidden layers. *Source:* [26] 12
1.7 Scheme of a single neuron (or perceptron) with its inputs, weights and outputs. *Source:* [31] 12

2.1 Experimental setup for performance comparison between different devices. The Ethernet link is not point-to-point, but consists of a switch within an isolated network operating at 1 GHz bandwidth that cannot affect the given results. 16
2.2 STM32H723ZG Nucleo-board. 17
2.3 Zynq 7010 solution embedding a configurable FPGA. 19
2.4 Comparison of the jitter on 600,000 cycles for each device under test during the experiment. 20
2.5 Comparison between FreeRTOS with persistent execution of link status detection (bordeaux) and the fixed version (green) at 140 μs. 21

3.1 Schematic diagram of AC power supply for JT-60SA. 24
3.2 Circuital diagram of EF PS for JT-60SA. 25
3.3 Operating mode for each state and state transition of the PF PS state machines. 28
3.4 General schema of the PS interface with the JT-60SA control. . 29
3.5 PSIM electric scheme of the PF PS. 30

3.6 Current control closed-loop of the PF PS for JT-60SA. 31
3.7 Current imbalance compensation block diagram. 32
3.8 Circulating current compensation block diagram. 32
3.9 Typical operation of the TF coil power supply during an experiment day. 33
3.10 Circuital diagram of TF PS for JT-60SA. 34
3.11 PSIM electric scheme of the TF PS. 35
3.12 Waveforms from the simulation of the PF PS. The current reference signal is a bipolar triangular waveform. 38
3.13 Zoom of the waveforms from the simulation of the PF PS with dummy load. The current profile decreases crossing the zero . . 39
3.14 Real waveforms from the simulation of the PF PS with dummy load. The current profile decrease crossing the zero. 40
3.15 Zoom of the waveforms from the simulation of the PF PS with dummy load. The current profile increases crossing the zero. . . 41
3.16 Real waveforms from the simulation of the PF PS with dummy load. The current reference signal is a bipolar triangular waveform. 42
3.17 Waveforms from the simulation of the TF PS charge and discharge during an experimental session 43
3.18 Zoom on the waveforms from the simulation of the TF PS charge and discharge during an experimental session. 44
3.19 Signals from shot E100738 of the IC database. Top panel: ten PF current signals I_{PF}. Middle panel: ten PF voltage signals V_{PF}. Bottom panel: nine Int$V$ signals. 47
3.20 Signals from shot E100941 of the IC database. Top panel: ten PF current signals I_{PF}. Middle panel: ten PF voltage signals V_{PF}. Bottom panel: nine Int$V$ signals. The V_{PF} excitation ends at approximately 24.5 s, whereas the dynamics of Int$V$ persist until over 26 s. 50
3.21 Signals from shot E100640 of the IC database. Top panel: ten PF current signals I_{PF}. Middle panel: ten PF voltage signals V_{PF}. Bottom panel: nine Int$V$ signals. 52
3.22 Resume scheme for the training phase. Inputs $V_{\mathrm{PF}}(t)$ are filtered at 10 Hz, downsampled to 100 Hz and normalized in interval $[0, 1]$. The block $[z^{-10}, z^{-20}, \ldots, z^{-200}]$ is a vector of delay operators that multiply the input in order to have the 20 previously indicated regressors. Outputs are filtered at 10 Hz and normalized in interval $[0, 1]$. 53
3.23 Resume scheme for the feedforward and correction phases. Inputs $V_{\mathrm{PF}}(t)$ are filtered at 10 Hz, downsampled to 100 Hz and normalized in interval $[0, 1]$. The block $[z^{-10}, z^{-20}, \ldots, z^{-200}]$ is a vector of delay operators that multiply the input in order to have the 20 previously indicated regressors. Outputs are then denormalized in interval $[0, 1]$ and used as corrector of the real measured disturbance. 55
3.24 Top: comparison between actual Int$V$ signals ("o" markeded lines) and predicted simInt$V$ signals (bold lines) for shot E100940 of the IC database; Bottom: corresponding corrected corrInt$V$ signals. 58

3.25 Top: comparison between actual Int$V$ signals ("o" markeded lines) and predicted simInt$V$ signals (bold lines) for shot E100641 of the IC database; Bottom: corresponding corrected corrInt$V$ signals. 58
3.26 Top: comparison between actual Int$V$ signals ("o" markeded lines) and predicted simInt$V$ signals (bold lines) for shot E100734 of the IC database; Bottom: corresponding corrected corrInt$V$ signals. 59

4.1 Scheme representing the step-by-step evolution of the learning-on-learning approach. 70
4.2 Three signals of Chua system state variables. The red part is the one used for the learning phase. 71
4.3 Four projections of the Chua system double scroll actractor. . . 72
4.4 Evolution of the four components of the weights in GD method in the first 350 epochs in Chua system learning. GD is represented in blue, GDL is represented in orange. 72
4.5 Evolution of the four components of the weights in LM method in the first 150 epochs in Chua system learning. LM is represented in blue, LML is represented in orange. 73
4.6 Normalized x,y-coordinates of Mars trajectory from June 1, 2000, to June 1, 2010. 74
4.7 Comparison between the original trajectory and the predicted ones. 75
4.8 The point-to-point mean square error for each coordinate. . . . 76
4.9 Four signals of hyperchaotic Lorenz system state variables. The blue part is the one used for the learning phase, while the red part is the one used for test. 77
4.10 Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Lorenz system. N_{hn} is the number of adopted hidden neurons. GD is represented in blue, GDL is represented in orange. 78
4.11 Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Lorenz system. N_{hn} is the number of adopted hidden neurons. LM is represented in blue, LML is represented in orange. 79
4.12 Actractors of the state variables (x, y, z) obtained by the four different methods. The original actractor is represented by black dashed line, LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green. 79
4.13 Actractor of the error between target and output of the networks in test phase on Lorenz system. LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green. 80
4.14 MSE per sample on the four state variables (x, y, z, ω) of the Lorenz system. On the top, the comparison between GD (blue) and GDL (orange), on the bottom the comparison between LM (blue) and LML (orange). 80
4.15 Evolution of the four components of the weights in GD method in the first 200 epochs in Hyperchaotic Lorenz system learning. 81

4.16 Evolution four components of the weights in LM method in the first 200 epochs in Hyperchaotic Lorenz system learning. 82
4.17 Comparison of the evolution by epoch of the weights components and of the error in GD and GDL for the case with number of neurons in the hidden layer $n_h = 2$. From the top, the weight profile of GD, the weight profile of GDL and the errors evolution (logarithmic scale) of both the method. 83
4.18 Comparison of the evolution by epoch of the weights components and of the error in GD and GDL for the case with number of neurons in the hidden layer $n_h = 10$. From the top, the weight profile of GD, the weight profile of GDL and the errors evolution(logarithmic scale) of both the method. 83
4.19 Comparison of the evolution by epoch of the weights components and of the error in LM and LML for the case with number of neurons in the hidden layer $n_h = 2$. From the top, the weight profile of LM, the weight profile of LML and the errors evolution (logarithmic scale) of both the method. 84
4.20 Comparison of the evolution by epoch of the weights components and of the error in LM and LML for the case with number of neurons in the hidden layer $n_h = 10$. From the top, the weight profile of LM, the weight profile of LML and the errors evolution (logarithmic scale) of both the method. 84
4.21 Signals of the $\mathbf{q}$ and ω vectors. The blue part is the one used for the learning phase, while the red part is the one used for test. 86
4.22 Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Spacecraft dynamics. N_{hn} is the number of adopted hidden neurons. GD is represented in blue, GDL is represented in orange. 87
4.23 Error bar for the MSE comparision (left) and for the MSE comparision (right) in test phase on Spacecraft dynamics. N_{hn} is the number of adopted hidden neurons. LM is represented in blue, LML is represented in orange. 87
4.24 Actractors of the vectorial part of the attitude quaternion (q_1, q_2, q_3) obtained by the four different methods from w-axis label. The original actractor is represented by black dashed line, LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green. 88
4.25 Actractor of the error between target and output of the networks in test phase on spacecraft system from w-axis label. LM is represented in blue, LML is represented in red, GD is represented in yellow, GDL is represented in green. 88
4.26 MSE per sample on the four part of the attitude quaternion (q_0, q_1, q_2, q_3) of the spacecraft system. On the top, the comparison between GD (blue) and GDL (orange), on the bottom the comparison between LM (blue) and LML (orange). 89

List of Tables

1.1 Main parameters of ITER. 7
1.2 Main parameters of JT-60SA. 8
1.3 Main parameters of DTT. 9

2.1 Average RTT for each device under test during the experiment. 20

3.1 Main electrical parameters of EF3 and EF4 transformers. . . . 26
3.2 Rectifier thyristors main parameters. 26
3.3 State transition criteria for the JT-60SA PF PS state machine. 27
3.4 Rectifier thyristors main parameters. 34
3.5 Variation of the order of weight gradient and the mean correction factor $C_{\%}$ on the feedforward phase dataset with respect to the number of training epochs. 56
3.6 Correction factors reached for different shots using the ANN approach. 59