POLITECNICO DI MILANO
Facoltà di Ingegneria
Dipartimento di Elettronica, Informazione e Bioingegneria

A NEURO-FUZZY CONTROLLER FOR SUPERCONDUCTING CABLE TESTING

RELATORE:
Prof. Ing. Cesare Svelto

CORRELATORE:
Prof. Ing. Pasquale Arpaia

TESI DI LAUREA DI:

Vincenzo DAPONTE
matr.n. 755767

Aprile 2013

Index

            1. Measurements plan
            2. Data analysis
            3. Numerical results
        2. System characterization
        3. Model definition
        4. Controller simulation
7. **Experimental validation**
    1. Primary tuning
        1. Measurements plan
    2. State of the art comparison

Appendix

Conclusions

Abstract

Superconductivity nowadays is finding more and more applications in addition to those strictly related to research and experimental physics. In particular, the critical current estimation plays a fundamental role in the characterization and usage of superconducting cables. At the European Organization for Nuclear Research (CERN) superconductivity is widely exploited in the development of particle accelerators such as the Large Hadron Collider (LHC). In this context the Facility for the Reception of Superconducting Cables (FReSCa) were created to perform test on superconducting cables and determine their properties in particular the critical current. There a measurement system based on a superconducting transformer and was introduced. The transformer different nonlinear behaviours showed led to a strategy for the transformer secondary current control resulted efficient under several constraints. Furthermore, the possibility to perform all tests only from the instrumentation rack computer does not allow the necessary view of all the relevant parameters to be monitored during a test session. The need for the measurement system to be integrated in the FReSCa control system together with the possibility to improve the current control performance resulted in the development of an integrated monitoring system. It provides the remote control of the measurement station and a control strategy that increases the performances with respect to the state of the art. Thus the proposed remote monitoring system for cable testing aims to allow the measurement system to be remotely driven through a CERN standardized remote device interface. Furthermore, it aims to overcome the limitations shown by the available measurement system related to the transformer secondary current control. The current control strategy was designed by following the Fuzzy Gain Scheduling logic which suggests changing in the controller parameters according to the working condition changes. The concrete implementation of the control logic was performed by means of a Neuro-fuzzy inference network.

Sommario

La Superconduttività al giorno d'oggi sta trovando sempre più applicazioni oltre a quelle strettamente connesse alla ricerca e alla fisica sperimentale. In particolare, la stima corrente critica gioca un ruolo fondamentale nella caratterizzazione e l'utilizzo di cavi superconduttori. Presso l'Organizzazione Europea per la Ricerca Nucleare (CERN) la superconduttività è ampiamente sfruttata per lo sviluppo di acceleratori di particelle come il Large Hadron Collider (LHC). In questo contesto, lo strumento per la Facility per i test su cavi superconduttori (FReSCa), è stata creati per eseguire i test su cavi superconduttori e determinare le loro proprietà, in particolare, la corrente critica. Un sistema di misura basato su un trasformatore superconduttore ed è stato introdotto recentemente. La dinamica non lineare mostrata dal trasformatore ha portato ad una strategia per il controllo della corrente sul secondario efficiente solo sotto diversi vincoli. Inoltre, la possibilità di eseguire tutti i test solo dal computer presente sul rack di strumentazione non consente la visualizzazione di tutti i parametri rilevanti durante una sessione di prova. La necessità che il sistema di misura debba essere integrato nel sistema di controllo a FReSCa insieme alla possibilità di migliorare le prestazioni del controllo di corrente hanno portato allo sviluppo di un sistema integrato di misura. Esso fornisce il controllo remoto della stazione di misura ed una strategia di controllo che aumenta le prestazioni della stazione rispetto allo stato dell'arte. Il sistema proposto vuole consentire al sistema di misura di essere guidato da remoto attraverso una interfaccia standard CERN per il controllo di dispositivi dispositivo remoti. Inoltre, si propone di superare i limiti indicati dal sistema di misura attuale relativi al controllo della corrente sul secondario del trasformatore. La strategia di controllo della corrente è stata progettata seguendo la logica Fuzzy Gain Scheduling che suggerisce di modulare i parametri del controllore in base ai cambiamenti delle condizioni di funzionamento. L'attuazione concreta della logica di controllo è stata effettuata per mezzo di una rete di inferenza Neuro-fuzzy.

Acknowledgements

A meno di un ora dalla consegna ufficiale di questo documento, faccio fatica a ricordare tutti coloro a cui debba e voglia dire almeno un “grazie”, non so se questo sia un bene perchè vuol dire che ho molti amici o un male perche vuol dire che ingnegneria fa davvero male come dicono. Sicuramente un grazie enorme per la disponibilitá e la pazienza lo porgo al prof. Svelto che mi ha aiutato in quest’ultima parte del mio cammino al Poli. Al prof. Arpaia non so davvero cosa dire, se prima ho detto grazie ora ci vorrá almeno un grazie infinite; ci siamo conosciuti in un modo molto tempo fa e ci siamo riconosciuti durante quest’esperienza (io non me lo ricordavo così). Un saluto ai miei amici di sempre Giovanni e Gino, presenze costanti e di grande supporto. Un grande in bocca a lupo voglio fare al Turco e signora con l’augurio di realizzare il loro sogno più grande. Un gigantesco grazie alla sorte per avermi fatto trovare due compagni di viaggio in una giungla, e uno a Francesco per la sua allegria e un altro enorme ad Alberto... lui sa per cosa. Un grande abbraccio voglio darlo a Chiara senza la quale sarei molto più scemo di quello che giá sono. Un pensiero affettuoso per il Direttore e signora che mi hanno accolto come fossi un rifugiato quando sbarcai a Milano e a Tonino per avermi fatto conoscere un’Italia che altrimeni non avrei mai visto. Un “cheers” ancora a tutti gli amici technical e all’Equipo C e agli amici italiani (G. e M. compresi) che hanno condiviso con me questo anno molto particolare. Un enorme grazie per tutto cio’ che mi hanno insegnato (non solo al lavoro) ai miei bodyguard della mente Lucio, Carletto, Giancarlo e il “Principale” più che un semplice capo. Un sincerro grazie a tre persone sincere come Robertone e Davidone e Andrea il presidente. Un fraterno grazie agli amici fraterni che ci sono sempre, anche quando non ci sono Francesco e Antonella. Oltre agli amici a Ginevra ho allargato anche la famiglia, ho acquisito due zie molto molto speciali, chi l’avrebbe mai detto! Mando un grande bacio ad entrambe.Il pensiero più profondo va alla famiglia, a quella lontana come i nonni, i miei angeli custodi, gli zii a Minori e a Roma in particolare va a Mamma e Papá perchè trovino la serenitá che meritano. A Sasha dico solo che gli voglio un bene dell’anima e gliene vorrò sempre, sperando che lui più di tutti riesca ad essere veramente felice.

Introduction

Superconductivity nowadays is finding more and more applications in addition to those strictly related to research and experimental physics. For each application, in addition to the usual cryogenics issues, the characterization of the superconducting cable properties is a key point in the development of this technology. In particular, the critical current estimation plays a fundamental role in the study and usage of superconducting cables. This current value identifies the point beyond which the cable loses its superconducting property assuming then a resistive behaviour. Particular attention has to be paid in this circumstance, in order to allow a proper dissipation of the residual current in the circuit under test. At the European Organization for Nuclear Research (CERN) superconductivity is studied and used for many years in the development of particle accelerators such as the Large Hadron Collider (LHC), and others are under construction. For this purpose, a facility able to host the test for the characterization of the superconducting magnets cables was built. In the Facility for the Reception of Superconducting Cables (FReSCa) [1] cables are tested to assess their properties, in particular the critical current. The current is fed in the sample to be tested through the power converter located outside of the cryostat. This configuration, although it allows high current values to be induced quite rapidly, results expensive both in terms of current consumed by the power converters and from the cryogenic point of view as it requires a large amount of helium to be provided at the cryostat. To overcome these issues a measurement system based on a superconducting transformer was introduced [2]. This system allows a relatively low current (tens of Ampere) to be fed in the transformer primary inducing a significantly higher current on the secondary where the sample cable is placed. The current is provided through a voltage controlled current source reducing drastically the consumption both of current and helium. The measurement system is locally controlled by the computer placed in the instrumentation rack. During the first working period the transformer showed different nonlinear behaviours which led to a strategy for the transformer secondary current control which

results efficient just under several constraints. In particular, the available measuring system does not allow strong accelerations and decelerations in the reference function especially if followed by very steep ramp rate. Furthermore, the possibility to perform all tests only from the instrumentations rack computer does not allow a complete view of all the relevant parameters to be monitored during a test session. In addition to the secondary current there are the cryogenic characteristic values of as well as other parameters typical of this type of measurement to be observed. These variables are generally monitored by the integrated control system of the FReSCa station, to which the above-mentioned measurement system is not connected. The necessity of the measurement system to be integrated together with the possibility to improve the current control strategy performance led to the development of an integrated monitoring system. Such a system is aimed to provide the remote control of the measurement station as well as a control strategy that increases the measurement performance with respect to the state of the art. The device remote control at CERN relies on the Front-End Software Architecture (FESA) [3] which provides a customizable interface of a remote device connected via a Front-End Computer (FEC) and communicating through the Technical Network (TN). This protocol provides a safe communication and control through a software abstraction of the physical device avoiding bottlenecks and latencies through the TN communication. Inside the integrated system for the station remote control, an adaptive strategy [4] for driving the current on the transformer secondary, thus on the cable, is introduced. The proposed strategy relies on an exhaustive description of the nonlinear dynamics of the transformer to design a control logic that adapts to the variation of the working conditions. The control strategy adopted follows the principle of the Gain Scheduling [5], by managing the transitions between the different system states through a fuzzyfication of the variables characterizing the states. This configuration is known as Fuzzy Gain Scheduling (FGS) and it was used with satisfactory results [6]. The concrete implementation of the control logic was performed by means of a Neuro-fuzzy inference network [7]. The network learns the input-output ideal behaviour of the FGS system, suitably generalized in order

to avoid abrupt transition between consecutive controller subdomains. To provide an exhaustive presentation of the proposed system this thesis is organized in four main parts, in the first a background is provided, in the second the prosed system is outlined, in the third part the numerical results are presented and the fourth reports the experimental results. The first part is composed by Chapter 1 providing a background on the superconducting cable test facility, and Chapter 2 where an overview of the Gain Scheduling strategy is given. In the second part the entire proposed system is presented in Chapter 3, while the current control strategy design is outlined in Chapter 4 and the development of the integrated remote monitoring system is presented in Chapter 5. The third part includes the Chapter 6 where the numerical results are exposed; finally the fourth part reports the experimental results discussion in Chapter 7.

References

[1] A.P. Verweij, J. Genest, A. Knezovic, D.F. Leroy, J.-P. Marzolf, L.R. Oberli "1.9 K Test Facility for the Reception of the Superconducting Cables for the LHC", *ASC'98 Superconductivity Conference, Palm Spring*, CA, USA, September 1998

[2] P. Arpaia, L. Bottura, G. Montenero, S. Le Naour, "Performance improvement of a measurement station for superconducting cable test", *Review of Scientific Instruments*, vol. 83, 095111, 2012.

[3] https://www-acc.gsi.de/wiki/pub/FESA/FESA29210/FesaEssentialsBundle.pdf.

[4] ROBUST ADAPTIVE CONTROL, 1996 Published By: PTR Prentice-Hall, Upper Saddle River NJ , Ioannou, P A Sun, Jing, 1996.

[5] Survey of Gain-Scheduling Analysis Design (1999) , We. Leithead ,International Journal of Control.

[6] Arnulfo Rodriguez-Martinez, Raul Garduno-Ramirez, and Luis Gerardo Vela-Valdes, "PI Fuzzy Gain-Scheduling Speed Control at Startup of a Gas-Turbine Power Plant", *IEEE Transactions on Energy Conversion*, vol. 26, no.1, pp. 310-317, March 2011.

[7] Jyh-Shing Roger Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference System, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. 23, NO. 3, MAYIJUNE 1993 b65.

Chapter 1

Monitoring system for cable testing

1.1 Test station for superconducting cables

In the years 1999-2004, about 6400 km of superconducting NbTi cable were manufactured in industry for the production of the coils for the main magnets of the Large Hadron Collider at CERN [1]. A crucial design parameter for any large-scale application of superconductivity is the current carrying capacity, also referred to as “critical current”. Its assessment needs for an accurate measurement of the voltage-current characteristic of the sample, in general, function of temperature, current, and magnetic field. To assess the critical current, the simplest solution is to test only few extracted strands forming the cable with a reasonable lower current. However, for a better understanding of the sample characteristic, the experiment should be carried out on the cable as a whole. This is a relatively delicate task for single wires. Nowadays, it is carried out through industrial standards. However, there are only few facilities of appropriate dimensions and functionality for large-size cables. Consequently, the difficulty and the cost of providing such a large and complex setup for assessing the device properties as a function of the abovementioned parameters become the main limitations. Cable critical current tests commonly involve current levels in the order of the tens of kA. To allow a gain in time and money, a test facility (called FReSCa - Facility for the Reception of Superconducting Cables) was built at CERN to fulfil the need for characterization of the electrical properties of the LHC superconducting magnet cables [2]. The main features of this test station are:

a) independently cooled background magnet;
b) DC sample current up to 32 kA maximum;
c) sample cooling either by superfluid helium at 1,8 K to 2,17 K either by liquid helium at about 4,3 K, both at atmospheric pressure;

d) measurement length of 56 cm;

e) magnetic field of 10 T homogeneous within 1 % along the sample length.

A manufactured LHC 1.3 m long dipole magnet (with an aperture of 56mm) is used to provide the background magnetic field. This is important to obtain a critical current of the cable that is a good average of all the strands in the cable. Thus, it is less sensitive to the non-uniform soldering resistances between the sample and the current leads. Due to the long cool-down time of the magnet (weight of about 2000 kg) a "double cryostat concept" is chosen (Fig. 1.1). The outer compartment contains the magnet, which is cooled down independently, and the inner compartment holds the sample. The magnet is connected through a pair of 18 kA current leads to a 16 kA external current supply. The lower bath of the cryostat, separated by means of a so called *lambda-plate* from the upper bath, can be cooled down to 1,9 K using a sub-cooled superfluid refrigeration system.

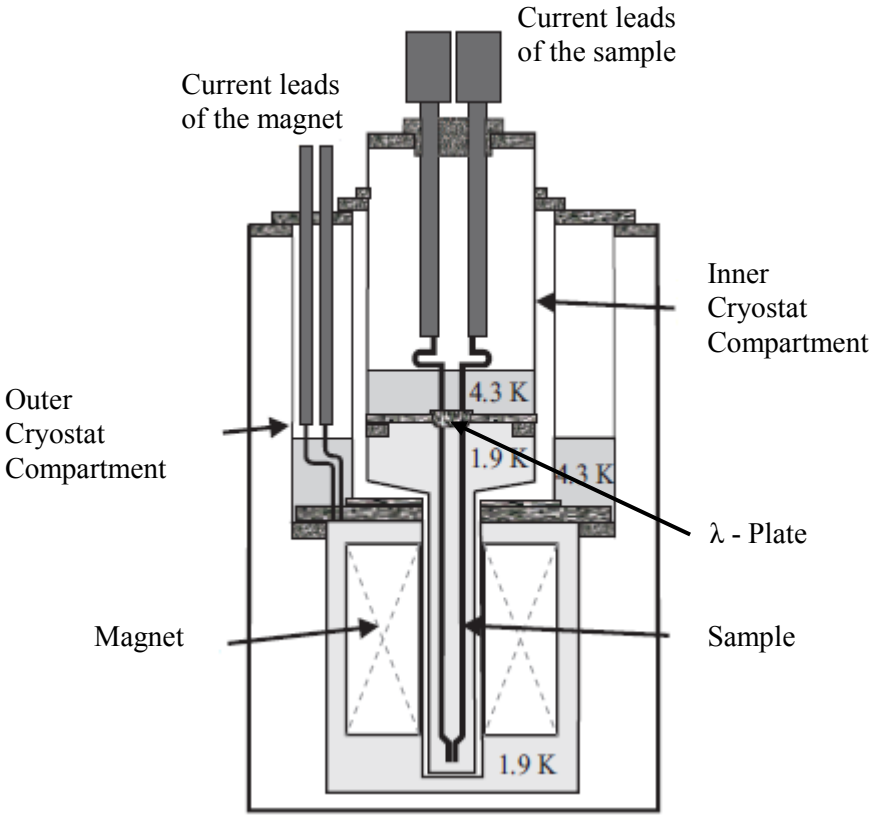

Figure 1.1 Overview of the FReSCa inner and outer cryostat compartments, containing the sample and the magnet, respectively.

This approach gives the possibility to change samples while keeping the background magnet cold, and thus, while decreasing the helium consumption and the cool-down time of the samples. The cable samples are connected through self-cooled leads to an external 32 kA power supply. The lower bath of the inner cryostat, containing the sample holder with two superconducting cable samples, can be cooled down to 1,9 K as well. The samples can be rotated while remaining at liquid helium temperature, making possible measurements with the background field perpendicular or parallel to the broad face of the cable. Several arrays of Hall probes are installed next to the samples in order to estimate possible current imbalances between the cables. The main advantage of this station resides in the possibility of externally producing, and thus measuring, the current. From another side, the excessive helium consumption, mainly due to the current leads cooling, is the principal drawback of the station. The use of a cryogenic superconducting transformer was introduced as a way to overcome room-temperature power converters limitations [3]. The need to measure currents at low temperature arises from exploiting transformers. DC superconducting transformers are composed by two air cored coils. The former is called *"primary"* winding, directly fed by the power supply and producing a variable magnetic field. The latter, called *"secondary"* winding, where the current is induced by the field variation. A low current is fed to the primary winding with a large number of turns, inductively coupled to the secondary one with a much smaller number of turns and directly connected to the sample under test (Fig. 1.2). The cold part of the transformer consists of a superconducting primary mutually coupled to a superconducting secondary. The transformer [4] were designed to reach in the secondary values of 1 μH for inductance and 10 nΩ for resistance, both assumed to be independent of the current. The large value for the resistance is due to the connections between the sample and the secondary: they are not soldered but clamped at high pressure in order to have a faster and easier mounting of the sample on the test stand. The primary winding of the transformer is wound from insulated NbTi wire with a diameter of 0.542 mm, a Cu/SC (Copper to Superconducting) ratio of 1.35, a RRR (Residual Resistivity Ratio) of 82, and a filament diameter of 45 μm.

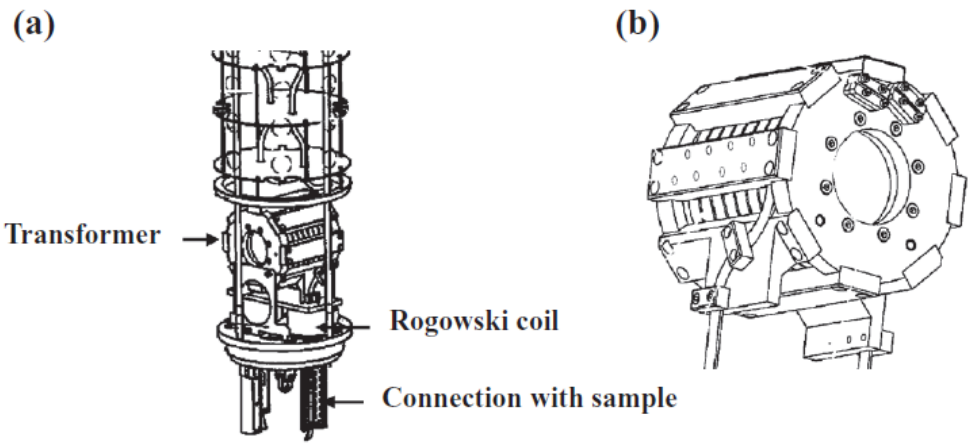

Figure 1.2 (a) The sample insert to introduce in the Inner compartment. (b) The superconducting transformer.

The primary has a solenoid shape with a height of 160 mm and inner and outer radii of 70 and 88 mm, respectively. The coil consists of 33 layers with a total number of turns of 10850. Its inductance value is 11.75 H. The secondary winding is wound directly over the primary, and consists of 7 turns of a NbTi *Rutherford* cable with a width of 15.1 mm. All along this cable, a copper strip of 1 mm thickness has been soldered for mechanical and electro-dynamical stability. The self-inductance of the secondary is 9 μH and the mutual inductance between primary and secondary is 8.77 mH. For protection purposes, voltage taps are soldered at 5 locations on the primary, at 5 locations on the secondary, and at three locations on the samples. For the application in the cable test facility, the secondary current should have a pre-defined ramp shape, and a regulation system is required in order to counter-balance the resistive losses in the secondary circuit. The modest primary current I_p is usually in the range of 100 A and is generated by standardized power supplies. The current feed through into the cryogenic environment is optimized to have lower cryogenic load. Such a device provides test capability at moderate operating cost.

1.2 Test station monitoring

In the superconducting transformer system architecture, the fundamental elements are the measurement system and the control strategy. Most of the standard methods for measuring current are not suitable for large currents in

cryogenic condition, and only less complex techniques can be applied directly. *Rogowski* coils represent a compromise between simplicity and accuracy of the measurement. However, in case of winding inaccuracy, a drift in long measurements can be expected and the sensibility to the external field of the transformer can be reduced. The measurement system (Fig. 1.3) is based on two coils connected in anti-series as sensing elements for the current, providing a good rejection to parasitic couplings. The transducer signal V_{RC} is integrated in time by using digital

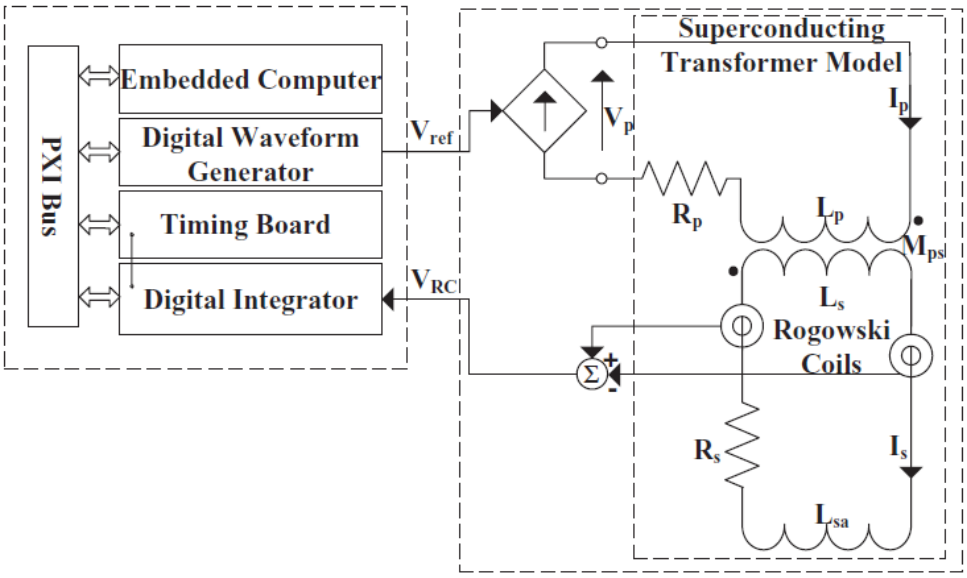

Figure 1.3 Architecture of the measurement system and the system under control. (Courtesy of Giuseppe Montenero).

integrators, the result is converted to current by means of a suitable calibration coefficient. The objective is to generate a test current I_s in the sample, i.e., in the transformer secondary, that will be proportional to the set point I_{ref}. With this aim, the main concern is to provide a suitable control of the current function, and an accurate measurement of the I_{sec}. Indeed, current measurement is the main factor for the control loop to be efficient. Resistive losses due to the interconnections between the secondary and the sample lead to an unavoidable decay of the current, unless the I_p is adjusted continuously to compensate and maintain the sample current at the desired set value. The control loop (Fig. 1.4) takes into account the physical characteristics of the coupled system that is formed by the

primary winding and its power supply, the secondary, and the current transducer. The controller, according to the feedback signal I_{mes} from the measurement system, acts to provide a reference voltage V_{ref} to the power supply, a voltage-controlled current source connected to the primary. The source drives the current I_p into the primary, inducing a secondary current I_s.

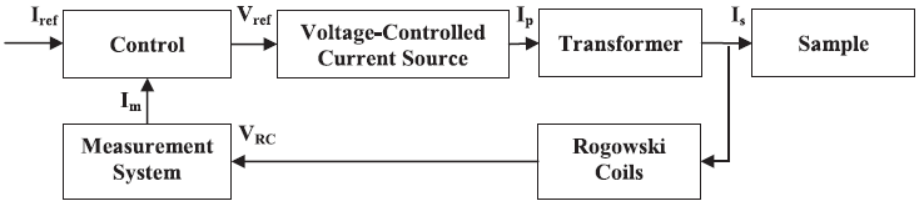

Figure 1.4 Architecture of the measurement system and the system under control. (Courtesy of Giuseppe Montenero).

The I_s is then sensed by the *Rogowski* coils, and a voltage signal V_{RC}, proportional to the secondary current variation dI_s/dt, is provided. The secondary current is consequently obtained by integrating the signal V_{RC}. Since the *Ragowski* coils allow only the variations of current to be measured, the initial value of the current has to be known when the integration is started for a suitable measure. At this aim, heaters are mounted on the secondary to warm up the cable above the critical temperature and to force a null current in the secondary. The measured current is finally compared to the reference I_{ref} in order to generate the feedback signal I_m compensating the resistive losses. A fully digital control algorithm (figure 1.2.4), taking into account only the plant characteristic without further analog signal handling, was developed. A discrete proportion-integral algorithm *PI(z)* was chosen for its simple implementation, tuning, and robustness.

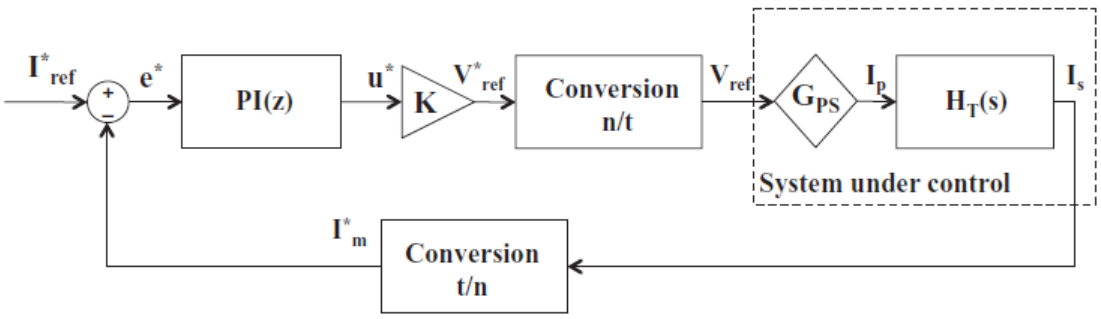

Figure 1.2.4 Diagram of the one-degree feedback controller. (Courtesy of Giuseppe Montenero).

The design of the Proportional Integral controller (PI) for the 36 kA superconducting transformer was based on the identification of the ideal transformer transfer function parameters. The scheme follows standard practice leading to a suitable choice of the controller parameters. The transfer function of the *PI(z)* was written for a backward digital integrator as

$$PI\,(z) = K_P + K_I \cdot z/(z+1) \quad (1.1)$$

where K_P and K_I are the gains of the proportional and integral actions, respectively. The gain K, in figure 1.2.4 (Fig. 3), is the transduction constant from the controller output u^* to V^*_{ref}, in turn related to the error signal e^*:

$$V^*_{ref}\,(z) = K \cdot PI(z) \cdot e^*. \quad (1.2)$$

The desired current in the transformer secondary I* ref is related to the measured current I* m, following

$$I^*_m = H_{CL}\,(z) \cdot I^*_{ref} \quad (1.3)$$

where H_{CL} is the closed-loop transfer function. The sample period depends on the required bandwidth. It has been customarily set to 5 to 20 times the inverse of the maximum allowed frequency. The resulting available 3 dB bandwidth is around 2 Hz (for a sampling frequency of 20 Hz) [3]. Moreover, based on experience, the proportional K_P and integral K_I gains have to be tuned in order to achieve the required performance for the noise affecting the voltage measurements of the sample under tests. The main limitation of this approach is the definition of the current ramps (set points) with a mandatory slow start and stop (narrow bandwidth). In other words, the beginning and the end of the linear ramp must be preceded and followed by acceleration and deceleration phases. The maximum available acceleration/deceleration is 800 As^{-2}. This value is limited by the

requirement of avoiding overshoots and undershoots of the current in the transformer secondary. The capability of the controller to follow such a fast change to reach the set ramp rate depends also on the K_P and K_I gains and on the value of the ramp rate as well. Therefore, during the transformer operation, the technician must be aware of all these details in order to properly set and carry out the test. Thus, the drawbacks of the available *PI* control strategy are:

a) Limited bandwidth;
b) Difficulties of the operator to set up a current cycle;
c) Tuning of the *PI* parameters depending on the working conditions.

References

[8] A.P. Verweij, J. Genest, A. Knezovic, D.F. Leroy, J.-P. Marzolf, L.R. Oberli "1.9 K Test Facility for the Reception of the Superconducting Cables for the LHC", *ASC'98 Superconductivity Conference, Palm Spring*, CA, USA, September 1998

[9] STABILITY OF SUPERCONDUCTING RUTHERFORD CABLES FOR ACCELERATOR MAGNETS - Geert Pieter Willering geboren op 25 oktober 1978 te Kampen

[10] P. Arpaia, L. Bottura, G. Montenero, S. Le Naour, "Performance improvement of a measurement station for superconducting cable test", *Review of Scientific Instruments*, vol. 83, 095111, 2012.

[11] A. P. Verweij, C-H. Denarie, S. Geminian, O. Vincent-Viry, "Superconducting Transformer and Regulation Circuit for the CERN cable test facility", *Journal of Physics: Conference Series 43* (2006) 833–836.

Chapter 2

The Gain Scheduling strategy

2.1 Adaptive control

Adaptive Control includes all the control strategies that are adapted to a controlled system with parameters that can vary or with initial uncertainty. For example, as an aircraft flies, its mass will slowly decrease as a result of the consumption of fuel; as a matter of fact, the control law must adapt itself to such changing conditions. Adaptive control is different from *robust control* since it does not need *a priori* information about the boundaries on these uncertainties or time-varying parameters; robust control guarantees that if the changes are within given boundaries, the control law does not need to be changed, while adaptive control is concerned with control laws changing themselves.

2.1.1 Main control approaches

There are two major divisions in control theory, namely, **classical** and **modern methods**, which have direct implications over the control engineering applications. The scope of *classical control theory* is limited to single-input and single-output (SISO) system design (Fig. 2.1), except when it is used to analyze disturbance rejection by using a second input.

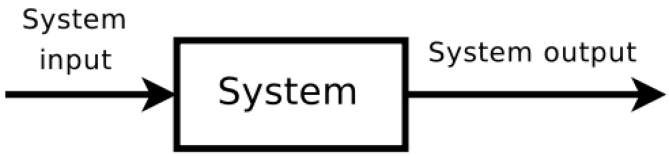

Figure 2.1 General scheme of a SISO system.

The system analysis is carried out in a defined time domain using differential equations, in a *complex-s* domain with *Laplace* transform or in frequency domain by transforming from the *complex-s* domain. Many systems may be assumed to have a second order and single variable system response in the time domain

ignoring multi-variable. A controller that is designed using classical theory often requires on-site tuning due to incorrect design approximations. Moreover, due to easier physical implementation compared to the systems designed using modern control theory, classical controllers are preferred in most industrial applications. The most common class of controllers designed using classical control theory is Proportional Integrative Derivative (PID) controllers. The ultimate goal is meeting a requirement set typically provided in the time-domain as the Step response, or at times in the frequency domain like the Open-Loop response. The Step response characteristics applied in a specification are typically: per-cent overshoot, settling time, etc. The Open-Loop response characteristics applied in a specification are typically Gain, Phase margin and bandwidth. These characteristics may be evaluated through simulation including a dynamic model of the system under control coupled with the compensation model. In contrast to the frequency domain analysis of the classical control theory, **modern control theory** uses the time-domain *state space* representation and can deal with multi-input and multi-output (MIMO) (Fig. 2. 2).

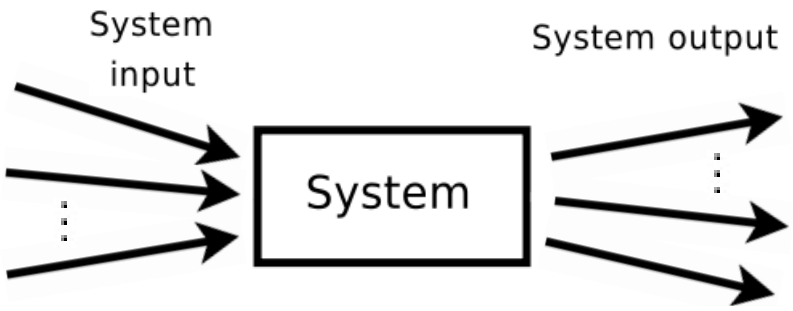

Figure 2.2 General scheme of a MIMO system.

The State space representation is a mathematical model of a physical system as a set of inputs, outputs and state variables related by differential equations of the first-order. To abstract from the number of inputs, outputs and states, the variables are expressed as vectors and the differential and algebraic equations are written in matrix form (the latter only being possible when the dynamical system is linear). The state space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyze the

MIMO systems. With several inputs and outputs, it is necessary to write down Laplace transforms to encode all the information about a system. Unlike the frequency domain approach, the use of the state space representation is not limited to systems with linear components and zero initial conditions. "*State space*" refers to the space where the axes are the state variables. The state of the system can be represented as a vector within that space. *Nonlinear*, *multivariable, adaptive* and *robust control* theories come under this division.

2.1.2 Adaptive control strategies

The key of adaptive control is a proper estimation of the parameters. Common methods of estimation include *recursive least squares* and *gradient descent*. Both of these methods provide update laws which are used to modify the estimatation made in real time (i.e., as the system operates). *Lyapunov stability* is used to derive these update laws and show convergence criterion (typically persistent excitation). Projection and normalization are commonly used to improve the robustness of estimation algorithms. In general the adaptive control techniques can be distinguished between:

a) Direct Adaptive Control;

b) Indirect Adaptive Control.

An adaptive controller is a combination of the estimation of different on-line parameter, which provides estimation of unknown plant parameters at each instant, with a controlled strategy that is related to the known parameter case. The way the parameter estimator, also referred to as adaptive law, is combined with the controlled strategies gives rise to two different approaches. In the first approach, referring to indirect adaptive control, the plant parameters are estimated on-line and are used to calculate the controller parameters. This approach is also known as explicit adaptive control, because the design is based on an explicit plant model. In the second approach, referring in this case to the direct adaptive control, a parameterized plant model is computed and the controller parameters are directly estimated without intermediate calculations

involving plant parameter estimates. This approach has also been set as implicit adaptive control since the design is based on the estimation of an implicit plant model.

2.2 Gain Scheduling

Gain-scheduling (GS) is one of the most popular approaches to nonlinear control design and has been widely and successfully applied in various fields such as aerospace or process control. It can be classified as an *open-loop adaptive control* system, because no feedback about the variations in the system performance is given to the decision block. Thus, the efficiency of the parameter adaptation cannot be checked. Although a wide variety of control methods are often described as "gain-scheduling" approaches, these are usually linked by a divide-and-conquer type of design procedure whereby the nonlinear control design task is decomposed into a number of linear sub-problems. This divide-and-conquer approach is mainly the reason of the popularity of gain scheduling methods since it enables the well-established linear design methods to be applied to nonlinear problems. The task decomposition is based on the relationship to be established between a nonlinear system and a family of linear systems. The main theoretical results which, for a broad class of nonlinear systems, relate the dynamic characteristics of a member of the class to those of an associated family of linear systems fall into two main sub-classes. Firstly, the relationship between the stability of a nonlinear system and the stability of an associated linear one constitutes the stability result. Secondly, the direct relationship between the solution of a nonlinear system and the associated linear systems one is established by doing an approximation on the results. It is important to distinguish these two classes of result. The formers are typically way more limited than the other one, since they are confined to specify conditions under which the solution boundedness to a particular linear system implies the nonlinear system solution to be bounded for an appropriate set of inputs and initial conditions. It is important to note that, under such conditions, even bounded solutions may be quite

dissimilar. Given a plant model *M* where, for each operating point *i* (where $i \in [1, N]$), the plant parameters are known, a feedback controller with constant gains k_i can be designed to meet the performance requirements for the corresponding linear model M_i. This leads to a controller *C(k)*, with a set of gains [k_1; k_2; …; k_i; …; k_N] covering N operating points (Fig. 2.3).

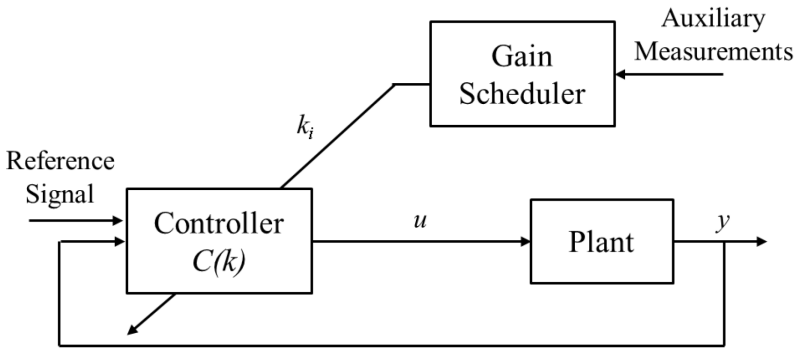

Figure 2.3 Gain Scheduling logic perspective.

Once an operating point is detected, the controller gains k_i can be changed considering the corresponding value from the pre-computed gain set. Transitions between different operating points leading to significant parameter changes may be handled either by interpolation or by increasing the number of operating points. Essential elements in the implementation of this approach are a look-up table, to store the k_i values, and the plant measurements correlating well the changes in the operating points. The key idea of a GS system consists on finding rules between the changes of the model and significant variables, hence the feedback is nonlinear and it may be implemented as a look-up table. Thus, a look-up table with an appropriate logic for detecting the operating point and choosing the corresponding controller values is needed. With this approach, the plant parameter variations can be compensated by switching the controller parameters according to the operating conditions (Fig. 2.4). The advantage of gain scheduling is that the controller parameters can be changed as quickly as the auxiliary measurements respond to parameter changes. Frequent and rapid changes of the controller gains, on another side, may lead to instability; therefore, a limit has to be placed on how often and how fast the controller gains can be changed. The pre-computed off-line adjustment mechanism of the controller

gains hides one of the potential GS disadvantages. An incorrect schedule caused by the lack of a compensating feedback together with the unpredictable changes in the plant dynamics may lead to deterioration of performance or even to complete failure.

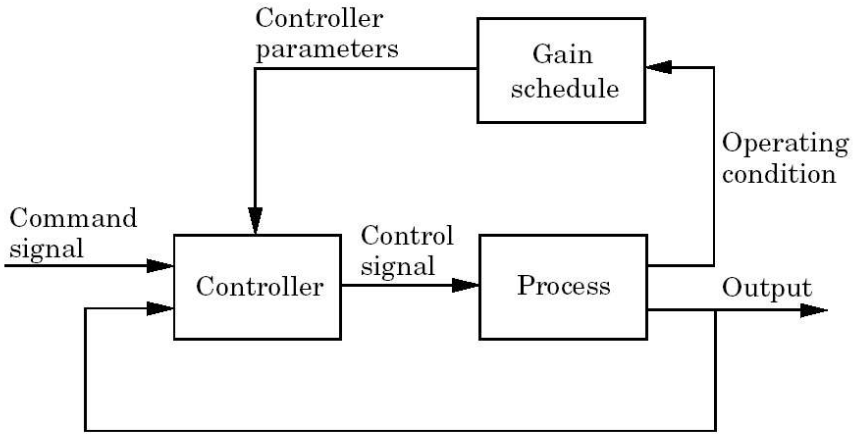

Figure 2.4 Gain Scheduling general scheme.

Another possible drawback of the GS method is the increase of the design and implementation costs according to the number of operating points. Despite its limitations, GS is a popular method for handling parameter variations in flight control and other systems. Another attractive approach for wide-range control of nonlinear processes is the Multimode control one. In this method, by switching among several controllers, each designed for a partition of the operating space is made in accordance to the operating conditions (Fig. 2.5).

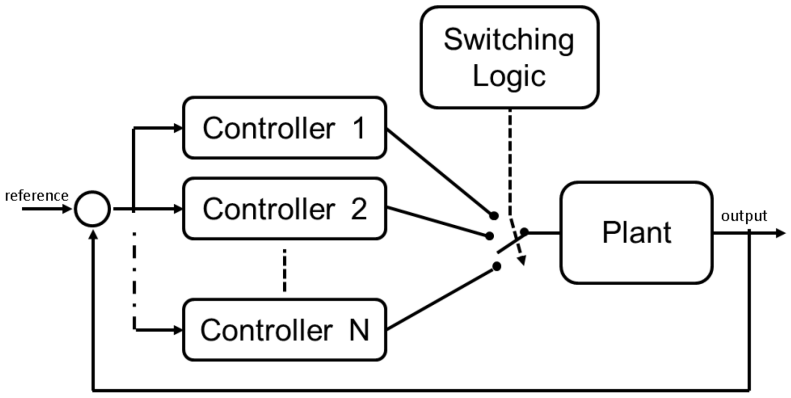

Figure 2.5: Multimode control scheme.

This approach is equivalent to the GS method provided the controllers to have the same structure, but in general, the controllers do not need to have the same structure or even to be of the same kind. This feature allows great flexibility in the type of control techniques that can be incorporated. The major concern is to achieve bumpless transfer between two different controllers and to avoid intermittent operation of two controllers in adjacent operating regions with sharp boundaries. In this method, as well as for the GS one, the implementation of the interpolation strategy and the switching logic is a key issue for the success of the control scheme to achieve a satisfactory behavior in a wide operating space. Even if there is not a generally accepted design procedure, the design of a GS-based controller typically proceeds in several steps. In a first time, a variable α, strongly correlated with the changes in the process dynamics, has to be chosen as the scheduling variable. This variable should be readily available and its time dependency be easily handled in a second time, a set of operating points covering the whole operating range of the process should be identified. This set defines a vector of values, A= $\{\alpha_1, ..., \alpha_n\}$ in the scheduling variable and a partitions of the operating space. In a third time, the linear controller at each operating point has to be designed, using the linear time-invariant models at those points if required, and store into the controller parameters. Finally, the gain scheduling interpolating scheme has to be delineated, to allow the controller parameters to be selected according to the corresponding operating points. Despite the many practical implementations of GS controllers, the lack of theoretical results about the global properties of the controlled system is one of the major concerns. Given excellent robust local stability and performance properties at the selected operating points, there is no guarantee that these properties hold at all points and between them. This eventuality is highlighted by the local nature of the control methods even if they are theoretically well-supported in a real application. Thus in general, and not exclusively for GS, the control system properties are demonstrated through extensive computer simulation experiments. Obviously, for wide-range operation, with many operating points of interest, the design could become a cumbersome labor.

2.3 Fuzzy Gain Scheduling

Fuzzy Gain Scheduling (PI-FGS) with PI controller is a particular implementation of the Gain scheduling. In this scheme, the system operating space is divided into a number of subspaces or partitions, assigning a generalized PI controller to each one of them. Each controller is tuned taking into account the plant dynamics in its partition. The main feature of PI-FGS resides in the use of Fuzzy logic as switching logic and the interpolation or GS function [4]. Fuzzy inference is used to implements the mechanism in order to detect the plant current operating conditions. Inference rules implement the generalized PI local controllers, one per rule (Fig. 2.6). The PI-FGS controller is based on a Takagi–Sugeno–Kan (TSK) fuzzy system [5] with two inputs and one output.

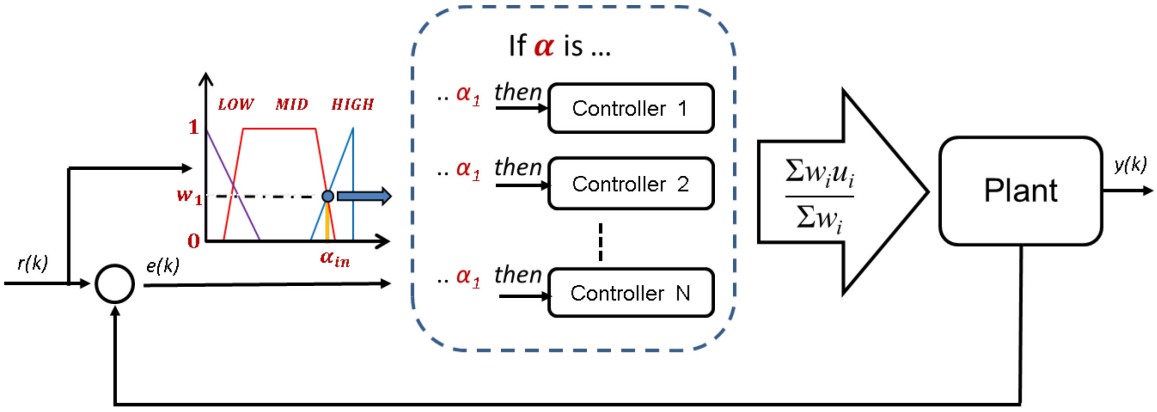

Figure 2.6 FGS system with one input divided into three subspaces and N controllers.

The first input enters the scheduling variable α, which, in the system under control, is identified as the current I_{ref} fed on the transformer secondary. The other input is the error signal $e(k)$, defined as the distance between the plant output $y(k\text{-}1)$ and the desired one $r(k)$, required by the generalized PI to calculate the control signal (figure 2.7). The output of the TSK fuzzy system is the control signal $u(k)$ that, in the superconducting transformers system is the power supply voltage input V_{ref}. The TSK fuzzy system has the following main characteristics. The scheduling variable membership functions are trapezoidal while a singleton fuzzyfication method is used to simplify calculations by the inference mechanism.

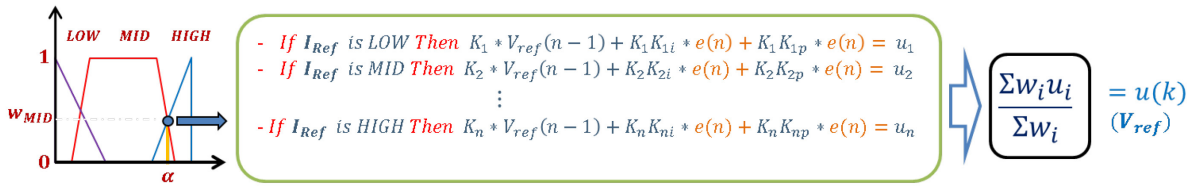

Figure 2.7 FGS with N – PI controllers implemented in the TSK rule base.

The inference system relies on a rule base with individual rules. The total output $u(k)$ is the weighted average combination of all rule outputs. Rules have the form

$$\text{IF } \alpha \text{ is } Ai \text{ THEN } u_i(k) = K_i.V_{ref}(k\text{-}1) - K_i{\cdot}K_{iI}.y(k\text{-}1) + K_i{\cdot}K_{iP}.e(k) \quad (2.1)$$

where $i \in [1, R]$ is the rule number, Ai is the fuzzy set defining the i^{th} partition of the operating space, K_{iP}, K_{iI}, and K_i are the generalized PI parameters or gains of the i^{th} rule or controller, and $u_i(k)$ is the control signal generated by the i^{th} rule or controller. The total control signal, generated by the TSK fuzzy system, is the weighted average of the control signals generated by each rule or controller

$$u(k) = \Sigma^R w_i\, u_i(k) \,/\, \Sigma^R w_i \quad (2.2)$$

where the weights w_i are the product of the membership values of the inputs being fuzzyfied. Since only the first input is being fuzzyfied

$$w_i = \mu\, Ai\, (\alpha). \quad (2.3)$$

From (2.1), (2.2), and (2.3), the control signal change $u(k)$ is

$$u(k) = \Sigma_i^R\, \mu Ai\, (\alpha){\cdot}\, (K_i.V_{ref}(\text{k-}1) - K_i{\cdot}K_{iI}.y(k\text{-}1) + K_i{\cdot}K_{iP}.e(k)) \,/\, \Sigma_i^R\, \mu Ai\, (\alpha). \quad (2.4)$$

This is the PI-FGS controller output for the plant, in the system in exam it would be the voltage value given to the power supply V_{ref}.

2.4 Neuro-Fuzzy inference

Both neural networks and fuzzy systems are non-linear modelling paradigms, both robust with respect to noise in the data; however, there are significant differences between them. The key idea of the Neuro-fuzzy inference is to combine the strengths of both techniques generating hybrid architecture combining the learning ability of neural networks with the capability to represent the problem in a simple and clear way, typical of the fuzzy systems. This combination allows the Neuro-fuzzy network to be trained with a suitable learning algorithm that, like in neural networks, iteratively adapts its parameters to optimize the degree of accuracy and to decrease the error. At the end of learning is possible to monitor the evolution of the membership function and the knowledge base, extracting from them the knowledge, which is not possible in artificial neural networks (ANN). The use of the Neuro-fuzzy inference implies the exploitation of the ANN to automatically extract, from the available data, the main parameters characterizing fuzzy logic-based system. The factors to be estimated are the membership functions parameters for the input variables and the fuzzy control rule coefficient implementing the local PI control. Therefore, a Neuro-fuzzy system is essentially a system able to learn knowledge from data by means of the typical ANN techniques and represent it explicitly in the form of fuzzy rules. Ultimately, by combining these two paradigms is possible to create networks that:

a) learn from experience (just like ANN);
b) are able to perform reasoning on inaccurate information (just like fuzzy systems);
c) Give a clear representation of the evolution of the model providing understandable explanations about how the model progression (such as fuzzy systems).

Such a network is characterized by a semantic, i.e. a precise meaning associated to each element of the network. If ANN can be made more complex by adding neuron layers, Neuro-fuzzy networks (NFN) can be enriched by introducing

additional rule layers to have a more specific model and describe particular aspects. The NFN, compared to the classical ANN, have the property of "driving" the parameters initialization, thanks to the well-defined semantics characterizing each of them (i.e., in a temperature domain mapping, the linguistic variable *HIGH* will be placed after the *LOW* one).

The output inference methods, given the input variable values, are mainly two, both adaptive. It means that the concepts expressed by the labels associated with the fuzzy-set are not precisely known, but they have to adapt to the particular application. These two methods essentially differ in the type of output:

a) Sugeno-Type Fuzzy Inference: the outputs are real values;
b) ANFIS (Adaptive Neuro-Fuzzy Inference System): the outputs are fuzzy set.

2.4.1 Sugeno inference

The Sugeno inference approach is widely used in control systems, in this model the rules are in the form:

$$\begin{aligned} &R_1: IF\,(x_1\ is\ A_{11}) \wedge (x_2\ is\ A_{12}) \wedge \ldots \wedge (x_I\ is\ A_{1I})\ THEN\ y = z_1 \\ &R_2: IF\,(x_1\ is\ A_{21}) \wedge (x_2\ is\ A_{22}) \wedge \ldots \wedge (x_I\ is\ A_{2I})\ THEN\ y = z_2 \\ &\ldots \\ &R_j: IF\,(x_1\ is\ A_{j1}) \wedge (x_2\ is\ A_{j2}) \wedge \ldots \wedge (x_I\ is\ A_{jI})\ THEN\ y = z_j. \end{aligned} \quad (2.5)$$

Where:

- $x_1, x_2, \ldots x_I$ are the system input variables;
- A_{ji} are the possible fuzzy set describing the input variables (each of them is associated to its own membership function);
- The antecedent of each rule is made up by a conjunction ("*AND"* operator) of propositional clauses ("*x is A"*);
- The y is the output variable and it can assume only crisp values z_j (i.e., numerical).

A system including such rules can be represented by means of a non-linear function of the input variables:

$$y = f(x_1, x_2, \ldots x_l). \quad (2.6)$$

The crisp value of the output variable is assessed by taking into account all the rules and their activation values, with respect to the z_j value assumed by the y variable in each rule:

$$y = \frac{z_1 \cdot VA(R_1) + z_2 \cdot VA(R_2) + \cdots + z_j \cdot VA(R_j)}{VA(R_1) + VA(R_2) + \cdots + VA(R_j)} \quad (2.7)$$

As an example, a two-rule model can be considered:

$$R_1\text{: IF } (x_1 \text{ is } A_{11}) \wedge (x_2 \text{ is } A_{12}) \text{ THEN } y = z_1$$

$$R_2\text{: IF } (x_1 \text{ is } A_{21}) \wedge (x_2 \text{ is } A_{22}) \text{ THEN } y = z_2 \quad (2.8)$$

The activation level a_j of each rule can be calculated as multiplication among the degrees of truth of the individual propositional clauses:

$$a_1 = A_{11}(x_1) \cdot A_{12}(x_2)$$
$$a_2 = A_{21}(x_1) \cdot A_{22}(x_2) \quad (2.9)$$

Then the y value, of the output variable, can be computed as:

$$y = \frac{a_1 \cdot z_1 + a_2 \cdot z_2}{a_1 + a_2} \quad (2.10)$$

Summarizing, the activation level of the j^{th} rule is expressed by the "*Larsen Product*" operator or by any other *T-norm* derivable for the "*AND*" operator:

$$a_j = \prod_{i=1}^{I} A_{ji}(x_i) \qquad (2.11)$$

Equation 3.13 shows how to calculate the activation level of the j^{th} rule with the *Larsen* product. The system output is then the centre of gravity of the local outputs:

$$y = \frac{\sum_j a_j z_j}{\sum_j a_j} \qquad (2.12)$$

2.4.2 Sugeno inference: network structure and learning

The network structure is derived from the rules of the fuzzy model by appropriately composing two types of neurons:

- Neurons implementing the inputs fuzzyfication. This implies the exact variables values (crisp) to be transformed into precise degrees of truth by the membership function related to the fuzzy-set specified in the antecedent clause;
- Special Neurons able to perform operations of sum, product and ratio.

In a Sugeno system the parameters to learn are:

- The fuzzy-set A_{ji} shape: for example, to define a sigmoidal shape is necessary to identify the position a_k and amplitude b_k;
- The rules output values z_j, i.e. the weights in input to the output layer.

Once the rules and fuzzy-set shape are defined it is possible to obtain the neural network that corresponds to the Sugeno system. At this point it is sufficient to define an appropriate figure of merit as

$$E = \frac{1}{2}\sum_{n=1}^{N}(t_n - y_n)^2 \qquad (2.13)$$

It can be then minimized by analytical techniques, such as the calculation of the partial derivatives of the error with respect to the outputs z_j and to the parameters

of the fuzzy-set a_k, b_k), or numeric (i.e. the gradient descent) to learn the weights. In order to provide a clear explanation, an example system can be considered. It can be characterized by two rules with two input variables, x_1 and x_2, and an output variable y, as described in (2.8). Let the fuzzy-set be A_1 (small) and A_2 (large) and let them to be characterized by a sigmoidal membership function defined by:

$$A_k(x) = \frac{1}{1 + e^{b_k(x-a_k)}} \qquad (2.14)$$

Three special neurons are then defined:

- P (.), it computes the product its inputs;
- S (.), it returns the summation of the inputs;
- R (.), it performs the ratio between the input terms.

The resulting neural network will have a customized structure (Fig. 2.8).

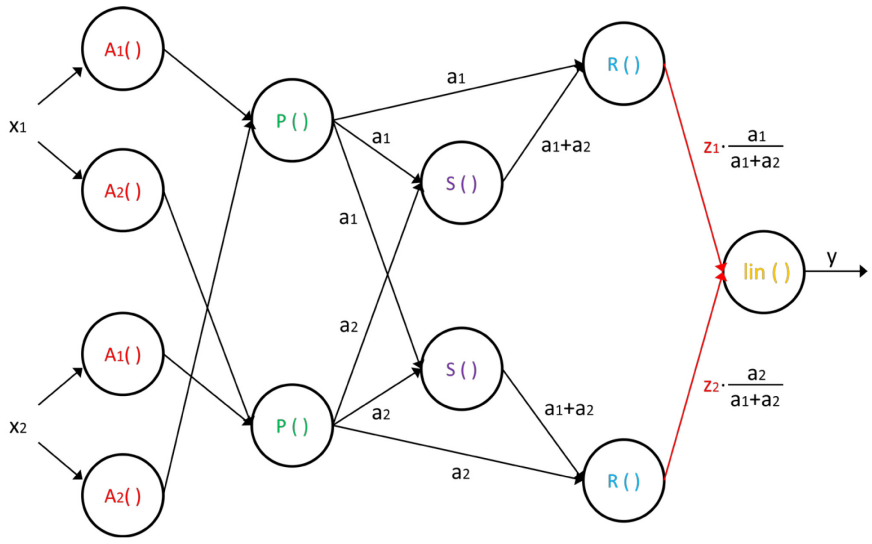

Figure 2.8 An example of Sugeno NFN with two input variables, two fuzzy sets per input, two fuzzy rules and one output.

Neurons $A_1(.)$ and $A_2(.)$ are appointed to the inputs x_1 and x_2 fuzzyfication. The neurons product $P(.)$, the sum $S(.)$ and the ratio $R(.)$, are placed in this order in to achieve the Sugeno inference by following the steps on the arches. The elements in red (the neurons $A_1(.)$ and $A_2(.)$ parameters, and the arc weights of the last layer

z_1 and z_2) are those to be trained with respect to the minimization of the figure of merit.

2.4.3 ANFIS inference

In the cases where the system output variables are fuzzy-set is necessary to use to the ANFIS inference [6]. With this kind of output variables, the fuzzy rules are of the type:

$$R_1\text{: IF } (x_1 \text{ is } A_{11}) \wedge (x_2 \text{ is } A_{12}) \wedge \ldots \wedge (x_I \text{ is } A_{1I}) \text{ THEN } y = O_1$$
$$R_2\text{: IF } (x_1 \text{ is } A_{21}) \wedge (x_2 \text{ is } A_{22}) \wedge \ldots \wedge (x_I \text{ is } A_{2I}) \text{ THEN } y = O_2$$
$$\ldots$$
$$R_j\text{: IF } (x_1 \text{ is } A_{j1}) \wedge (x_2 \text{ is } A_{j2}) \wedge \ldots \wedge (x_I \text{ is } A_{jI}) \text{ THEN } y = O_j. \quad (2.15)$$

Where:

- $x_1, x_2, \ldots x_I$ are the system input variables;
- A_{ji} are the possible fuzzy set describing the input variables (each of them is associated to its own membership function);
- The antecedent of each rule is made up by a conjunction ("*AND"* operator) of propositional clauses ("*x is A"*);
- The y is the output variable and is assigned to an appropriate output fuzzy set O_j.

As in the Sugeno inference, the activation level of the j^{th} rule is expressed by the Larsen Product operator (3.11) or by any other *T-norm* derivable for the "*AND"* operator. The j^{th} rule output is defuzzyfied starting from its activation value and from the fuzzy-set associated to the output variable of the same rule:

$$z_j = O_j^{-1}(a_j) \quad (2.16)$$

The output of the system is again the centre of gravity of the outputs as in (2.12).

2.4.2 ANFIS inference: network structure and learning

In the ANFIS inference, like in the Sugeno one, the network is structured in five logical layers, each of them with a precise logical task (Fig. 2.9):

I. Inputs fuzzyfication: the neurons of this layer calculate the degree of truth of one input with respect to a fuzzy set. In other terms, the degree of truth of each clause is computed;

II. AND operator (rule activation value computation): this operator is implemented by means of a differentiable *T-norm* as, for example, the product. In this layer there is one neuron for each rule. The j^{th} neuron calculates the activation level a_j of the j^{th} rule, i.e. the rule weight;

III. output defuzzyfication and normalized rule weight calculation: in this layer two operations are carried out:

   a. Output defuzzification z_j by using (2.16).

   b. Normalized rule weight calculation: the j^{th} rule weight is calculated by normalizing it with respect to the sum of all the rules weights

$$B_j = \frac{a_j}{\sum_j a_j} \quad (2.17)$$

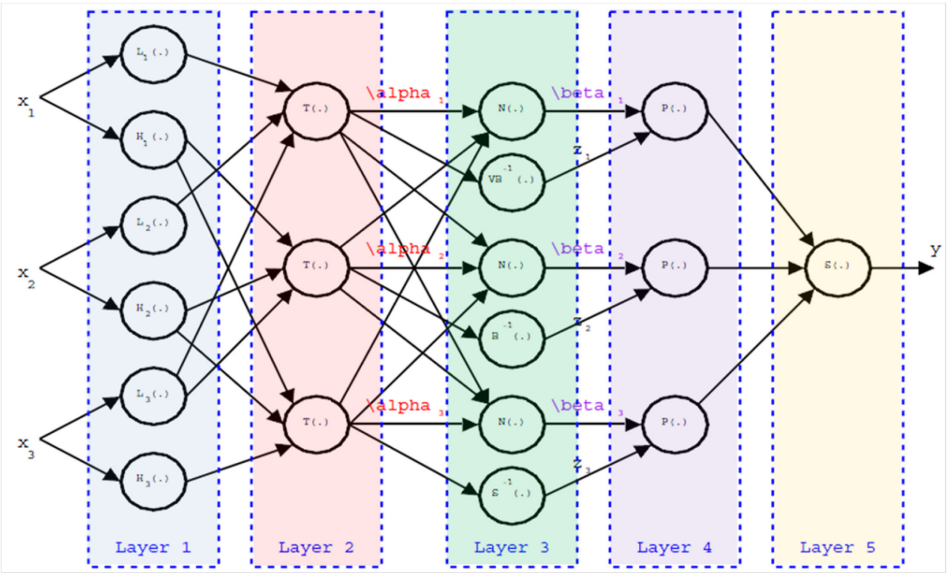

Figure 2.9 An example of ANFIS NFN with three input variables, and one output.

IV. Layer III outputs product: this layer performs the product of the two parameters calculated in the previous parameter B_j and z_j

$$y_{IVj} = B_j \cdot z_j \quad (2.18)$$

V. The network final output: the final output y is computed as the sum of the coefficient calculated in previous layer

$$y = \sum_j B_j z_j \quad (2.19)$$

Once the network is built following the described structure, a back-propagation algorithm has to be applied to learn the parameters of the rules from the data. In an ANFIS network no weight is learned because they are all set to one. On the other hand, only the activation function parameters of the layer I and III nodes (the layer where fuzzyfication and defuzzyfication are carried out) are learned.

It is worthy to note that any ANFIS system can be traced to a Sugeno inference system. Therefore, given a rule base with the consequent of any type, either Sugeno or ANFIS, it is possible to describe the output as a non-linear function *f(x)* of the inputs as in (2.6). If the function *f(x)* is differentiable (if and only if the T-norms, T-co-norms and inference operators defined for *f(x)* are differentiable), then also the figure of merit selected is differentiable, that is the error function E as defined in (2.13). It is then possible to use the gradient descent to minimize the error function E with respect to the parameters, as in neural networks

$$k^{t+1} = k^t - \gamma \left. \frac{\partial E(f, D)}{\partial k} \right|_{k^{n-1}} \quad (2.20)$$

Where:

- γ is known as *learning rate* and is a constant coefficient typically between 0 and 1;
- k^t is the solution chosen at the t^{th} step, starting from k^0 and randomly selected;

- $\frac{\partial E(f,D)}{\partial k}$ is the gradient of the error function E with respect to the network parameters.

References

[12] ROBUST ADAPTIVE CONTROL, 1996 Published By: PTR Prentice-Hall, Upper Saddle River NJ , Ioannou, P A Sun, Jing, 1996.

[13] Survey of Gain-Scheduling Analysis Design (1999) , We. Leithead ,International Journal of Control.

[14] Fuzzy Scheduling Control of a Power Plant, Raul Garduno-Ramirez, Student Member, IEEE and Kwang Y. Lee, Senior Member, IEEE. Department of Electrical Engineering, The Pennsylvania State University University Park, PA 16802 USA.

[15] Arnulfo Rodriguez-Martinez, Raul Garduno-Ramirez, and Luis Gerardo Vela-Valdes, "PI Fuzzy Gain-Scheduling Speed Control at Startup of a Gas-Turbine Power Plant", *IEEE Transactions on Energy Conversion*, vol. 26, no.1, pp. 310-317, March 2011.

[16] T. Takagi, M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, January/February 1985.

[17] Jyh-Shing Roger Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference System, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. 23, NO. 3, MAYIJUNE 1993 b65.

Chapter 3

Remote smart measurement system for superconducting cable test

3.1 The proposed system

In the current available configuration at FReSCa, the measurement station is placed near to the cryostat where the sample insert is located. The test session has to be run locally from the measurement rack computer, while all the cryostats are controlled from a central monitoring station. Typically, a measurement session includes first the achievement of the necessary cryogenic conditions. The next step involves the setup of the background magnetic field, then the sample powering and the critical current measurement. Simultaneously with the achievement of the critical current and the consequent quench, it is necessary to activate the system resistive configuration and dissipate the current through the heaters. All these operations have to be driven from a unique and integrated system allowing all the system variables to be supervised by the operator. Therefore, the need for the superconducting transformer measurement station to be integrated in a remote monitoring system arises. The measurement station is driven by a software application running on the rack computer, thus the remote control of this software system is a key point for the measurement system integration and utilization. At CERN, all the device remote control follows an interface standard ensuring a safe interaction with the controlled system provided by the use of the Technical Network as communication channel. The network offers connectivity for industrial systems and accelerator control devices and it is not directly connected to the Internet. Therefore, the proposed remote monitoring system for cable testing has the aim to fulfill the main twofold disadvantages of the available system. On one side, it has to allow the measurement system to be monitored remotely through a standardized remote device interface. On the other side, it has to give the possibility to overcome the drawbacks shown by the

available measurement system related to the transformer secondary current control, such as the limited bandwidth, the difficulties for the operator to set up a current cycle and the PI parameters tuning depending on the working conditions.

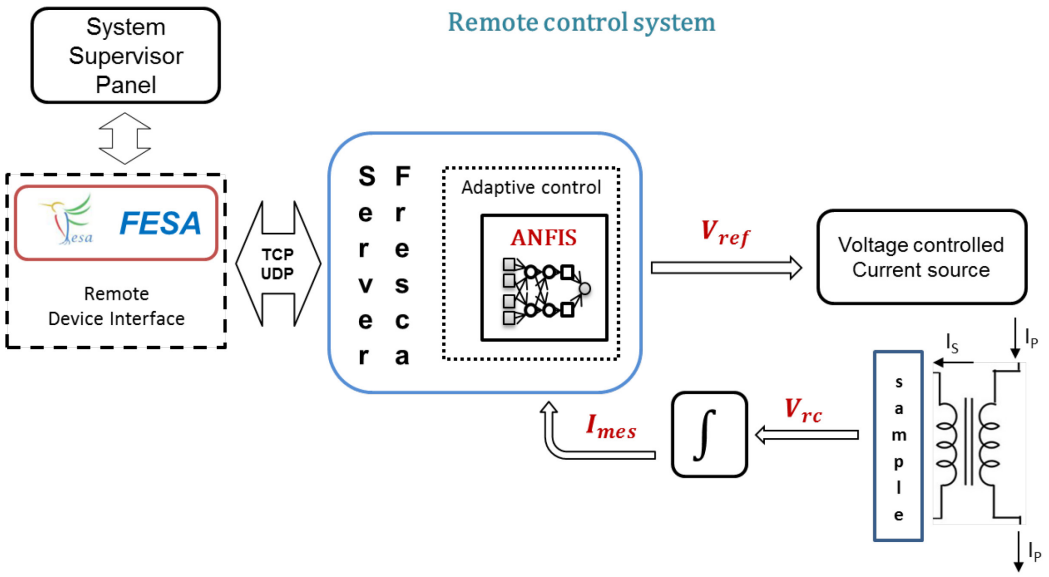

Figure 3.1 A logic view of the proposed remote monitoring system for superconducting cable testing.

Consequently, the proposed system was developed according to twofold main directions: the enhancement of the sample current controller by an adaptive control strategy and the remote control of the measurement station. In Fig. 3.1, a logic view of the proposed remote monitoring system for superconducting cable testing is reported. The system control panel, developed in LabView by an eternal operator, interacts with the remote device interfaces by proper library.

3.2 Adaptive current control

During the development of the available current control [1], the system under control showed several non-linear behaviours. In the development of its control strategy, a linearized model based on the ideal transformer physical equations was considered. As a result, a digital proportional integral controller operating within certain limitations was implemented. In order to obtain a significant performance improvement, a controller able to operate beyond the currently

imposed limits is needed. At this aim, in the design process, a more detailed model taking into account non-linear dynamics of the system under control is to be developed for its entire operating domain. An exhaustive characterization of the system input-output dynamics was obtained by carrying out an extensive measurement campaign in several different working conditions. The system to be controlled is composed by the cascade of the power supplier and the superconducting transformer (Fig. 3.2).

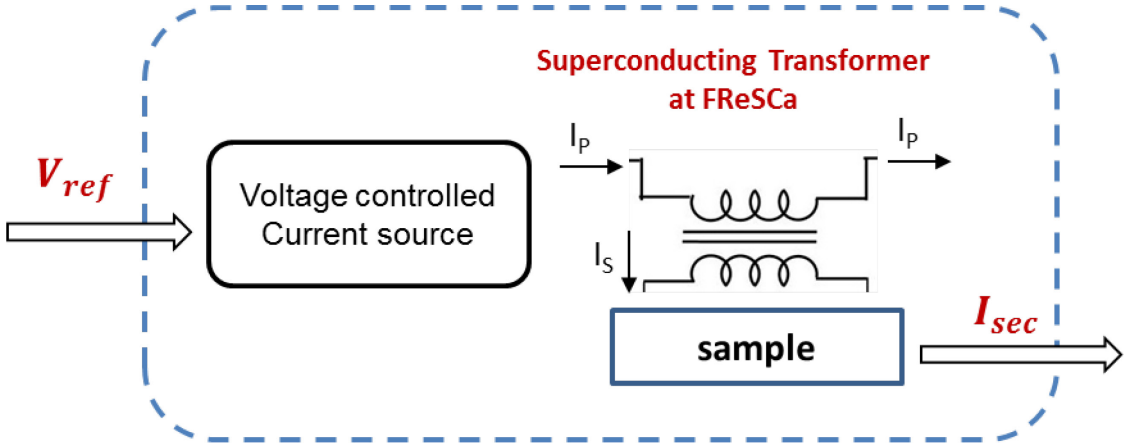

Figure 3.2 The system under control.

The system input is the reference voltage V_{ref} fed into the power supply while the output is the current I_{sec} measured on the transformer secondary. The field of the nonlinear systems identification is wide and the included techniques can be either mathematical or statistical inference-based [2]. The physical analysis of the magnetic couplings and the other physical phenomena present in the cryogenic part of the superconducting transformer is extremely complex. Therefore, it would be a highly difficult task to build a model based on a sufficiently detailed physical analysis; hence an inferential approach was adopted to define the model. Among the inferential identification methods, the autoregressive neural networks approach [3] was initially considered, but it did not provide sufficient generalization of the system dynamics for all operating conditions. More satisfactory results were provided by using a fuzzy identification [4]. From the resulting model, an ideal control strategy able to provide a control reference for each operating region was synthetized. Such a controller was designed according

to the PI-FGS strategy, where for each operating region identified an ideal PI controller is computed. This strategy is aimed at producing the reference for the ANFIS network training, carried out using the simulation results of the ideal controller. At the end of the training process, the *C++* ANFIS implementation was completed with the parameters computed in the simulation. The testing and tuning phases complete the development process of the integrated system.

3.3 Remote device interface

The design of the software for front end devices in the CERN control system architecture is a cumbersome task. Devices such as PLCs, VME/PXI modules, CCD cameras and so on have to be properly integrated. As the number of devices at CERN is unpredictably high, the control group decided to develop a framework where the user of a device is able to implement the required software according to an adequate standard. This framework is referred to as the Front End Software Architecture (FESA) [5]. On one or several user's front-end CPUs (FEC) or PCs, an executable program (device class) runs performing the required tasks for a specific hardware handling. FESA enables to implement the device class. The class is defined following the intermediate steps concerning the timing connection, deployment and instantiation. The main goal of this framework is to provide gets and sets functions that can be executed within an adequate time frame. The user can control his device (Fig. 3.3) trough requests addressed to the device model abstracted by the FESA (device) class. The class, invoking real-time handling services accesses the Hardware device providing the required service. The software model of an underlying hardware device is a data-holder that contains attributes which can be settings, acquisitions, or dynamic state-variables, and whose values at any given time provide an accurate snapshot of the underlying hardware device.

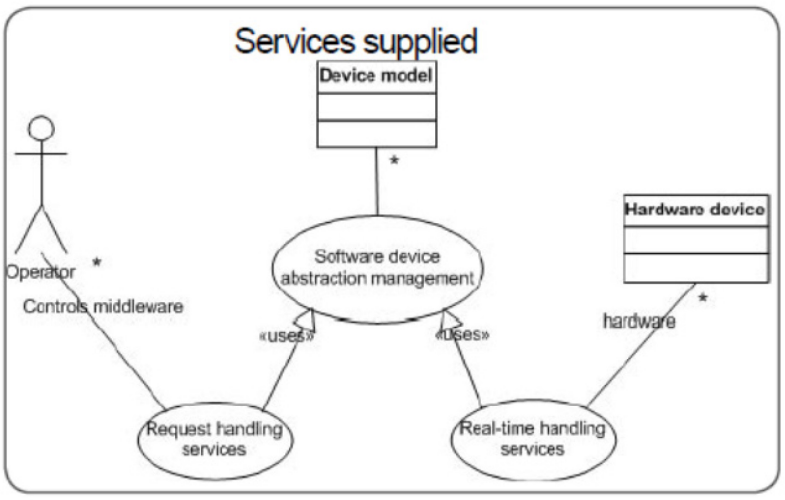

Figure 3.3 the custom FESA interaction diagram.

The device model core activity is to ensure that both the software abstraction to the user, and its underlying hardware device continuously reflect each other's state at runtime. Ensuring such a real-time correspondence involves information flowing in both directions:

- Controls flow from the device model and down to the hardware.
- Acquisitions flow from the hardware and up to the device model.

This approach defines a software package that provides a partial yet generic solution that can be tailored, i.e. customized, on a case-by-case basis in order to suit the specific needs of the equipment specialist.

References

[18] P. Arpaia, L. Bottura, G. Montenero, S. Le Naour, "Performance improvement of a measurement station for superconducting cable test", *Review of Scientific Instruments*, vol. 83, 095111, 2012.

[19] O. Nelles, "Nonlinear System Identification", Springer, 2001, XVII, 785 p.

[20] S. Chen, S. A. Billings, P. M. Grant, "Non-linear system identification using neural networks", *International Journal of Control*, 51:6, 1191 – 1214. January 1990.

[21] Xu, Chen-Wei Wei, "Fuzzy systems identification", Control Theory and Applications, IEEE Proceedings D, 136, 4, 146 – 150, Jul 1989.

[22] https://www-acc.gsi.de/wiki/pub/FESA/FESA29210/GettingstartedwithFESA.pdf

Chapter 4

Neuro-Fuzzy control system

4.1 Model Identification

The model identification of the system to be monitored is the first step in the control strategy development process. Having already underlined the presence of non-linear behaviours of the system, it became necessary to comprehensive measurement campaign covering all the working regions. Each run carried out to identify the system is based on the same current profile (Fig. 4.1).

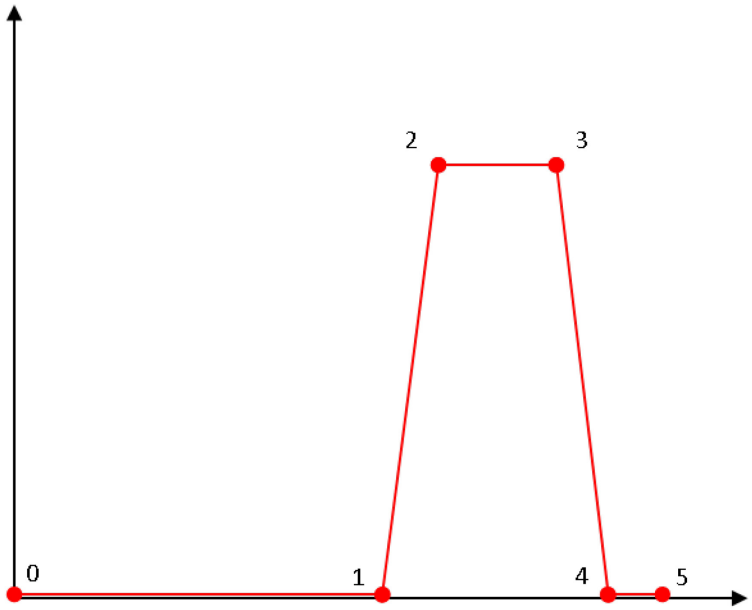

Figure 4.4 current profile for system identification measurements.

The first part of the profile (from point 0 to point 1) consists of a plateau at zero current that lasts for a minute. The corresponding measurement part is then used in the data analysis phase to estimate the measurement offset. From point 1 to point 2 the current ramp begins at a given ramp rate, here the profile continues with a ten seconds plateau (from point 2 to point 3) at the maximum current level. From step 3 to step 4 current drops back to zero through a symmetrical ramp. The profile is completed by a two seconds plateau at zero current. From the analysis

of the use of the system was possible outline a measurement plan taking into account the main current levels included in the domain of the transformer reached through five different current ramp rate (Table 4.1). The granularity with which the test runs were distributed was graded according to the different ramp rates: the current difference between consecutive measurements is short for lower ramp rate runs, while it is increased as the ramp rate increases.

Ramp Rate (A/s)	**50**	**100**	**300**	**500**	**800**
Current step (A)	500	500, 1000, 1500	500, 1000, 1500	1000, 1500	1500, 2000
Current Range (A)	From 500 to 20000	From 500 to 24500	From 500 to 24500	From 500 to 23000	From 500 to 30500

Table 4.5 Summary of the system identification measurements.

The considered current ramp rates start from a minimum of 50 A/s up to 800 A/s, the maxim current increasing steps are then selected from a minimum value of 500 A to a maximum of 2000 A in the last runs at 800 A/s. The current domain explored reaches a maximum value of 30500 A, achieved by an 800 A/s ramp rate. At this stage the collected data were subjected to a post-processing phase where the calculated offset was removed from the measures. The result was a set of one hundred twenty-five measurements run where for each test run were measured the input signal V_{Ref}, the current on the transformer primary I_P and on the secondary I_{Sec}. To derive a model from this data set several consecutive working regions were recognized. In each region is identified by considering the range where the system could be approximated by the same continuous time transfer function. When a transfer function is not able anymore to map satisfactorily the system dynamics, a new region has begun. Twelve regions were identified (Fig. 4.2) throughout the operating range (i.e. from 0 to 36 000 A).

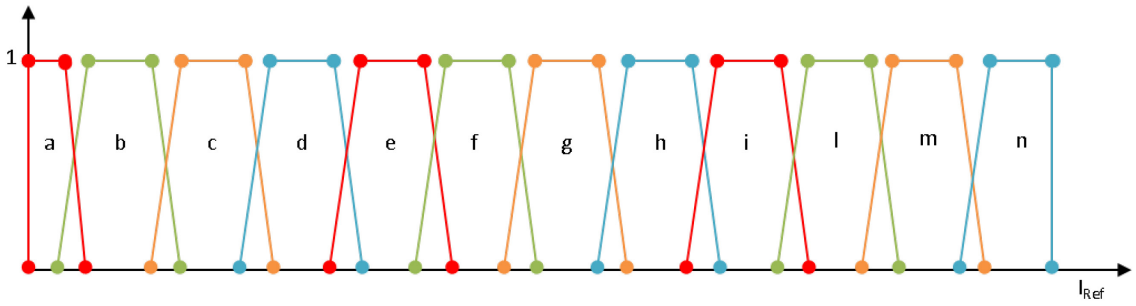

Figure 4.2 partitions of the operating domain.

It is worthy to note that the system dynamic, for each working condition was identified independently from the ramp rate; in particular the function defined for each region approximates the behaviour of the system for every considered ramp rate. At the end of this analysis to synthesize a single model of the entire system domain (Fig. 4.3), the various functions have been included into a Sugeno fuzzy system [1].

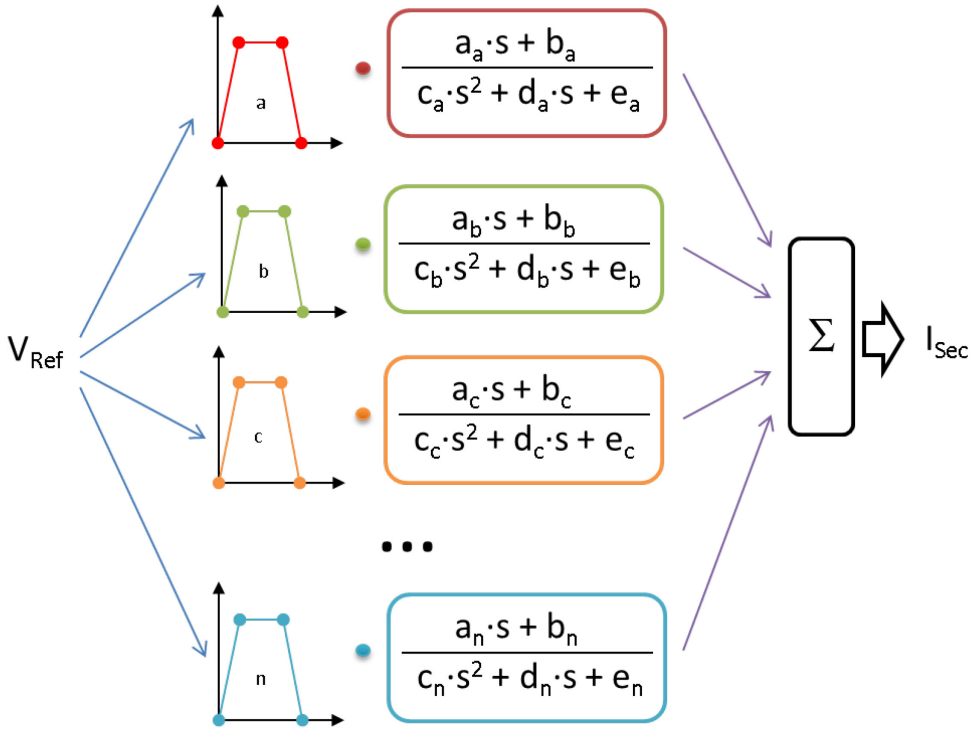

Figure 4.3 simulated system model.

The fuzzy system input variable is the power supplier input voltage V_{Ref}, the same of system to be controlled. The domain of this variable was fuzzyfied by taking into account the operating regions identified looking at the secondary current

measurements. The shape of the membership functions is trapezoidal in order to have a clear definition of the subdomain mapped. The consequent of the Sugeno system rules is represented by transfer functions computed. Each of them is characterized by a second order denominator and a first order numerator. Each rule, one for each membership function, has the same unitary weight. The output of the simulated system is given by the sum of the outputs of each consequent weighed with the truth value of the corresponding membership function.

4.2 Controller design

At this stage, from a model that satisfactorily replicates the system dynamics along the entire domain and for each ramp rate, a controller able to correct the non-linear behaviours can be designed. The reference strategy adopted is the gain scheduler one; in particular the multimode control logic [2] where all controllers are of type PI, thus the two strategies are equivalent. This development path has several advantages; first it produces a limited number of parameters to be identified, then a physical correlation between the parameters of the single PI and the domain region controlled, and finally reference values and performance are given by the available PI control [3]. The scheduling function, which changes the control according to the variations of the operating conditions, is implemented by a Sugeno fuzzy system [4]. The ideal controller (Fig. 4.4) is then based on the same structure of the system model. This fuzzy system expects two inputs, the reference current I_{Ref} and the error e and between the I_{Ref} and the measured current on the transformer secondary I_{mes}, and produces one output representing the voltage value V_{ref} to be fed into the power supplier. The reference current plays the role of scheduling variable and it is fuzzyfied following the same partition used for the system model. Even in this case trapezoidal membership functions were used; such functions have the unimodality property. Moreover, as for the model structure, for each value of the domain the sum of the values assumed by the membership functions of the different fuzzy sets is equal to one.

The second controller input, the error between I_{Ref} and I_{mes}, was not fuzzyfied because it is used only in the calculation of the control output value.

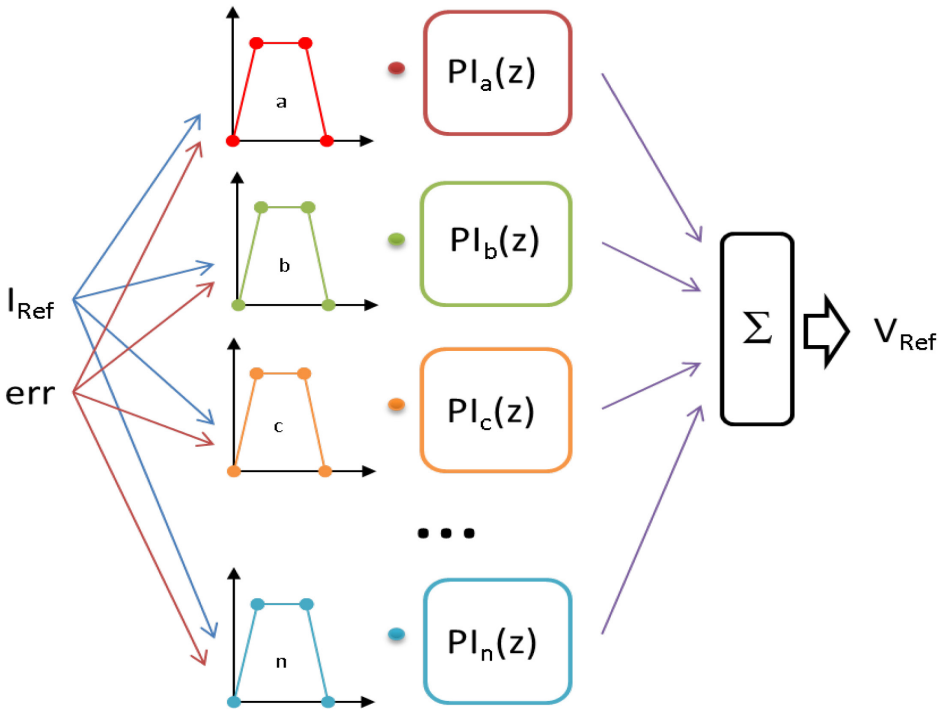

Figure 4.4 the structure of the computed PI-FGS controller.

Analogously with the system model the rule base consists of twelve rules, one for each membership function associated with the scheduling variable. A digital PI controller, a linear combination of the input variables, is implemented in the consequent of each rule. The single digital PI controller has the same structure of the one implemented in the available control system (Fig. 4.5).

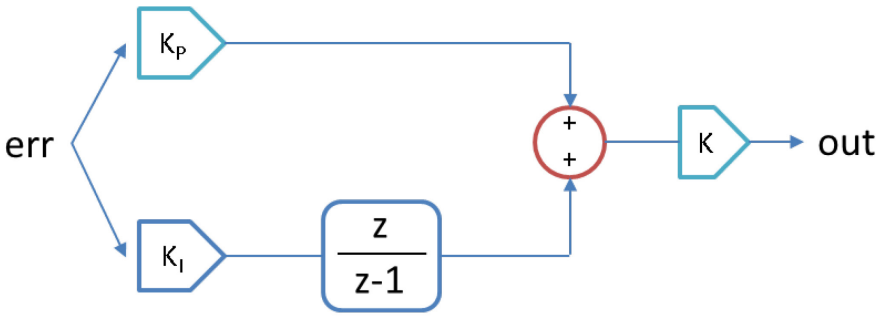

Figure 4.5 the structure of the digital PI controller with backward difference integration.

The backward differences algorithm was chosen as integration method. The controller output is given by the sum of the individual controllers outputs weighed with the truth value given by the corresponding membership function.

4.3 Neuro-Fuzzy strategy implementation

Once the validity of the controller was verified on the system model, the data to be used in the Neuro-fuzzy network training were produced. From the chain including the model and the ideal controller (Fig. 4.6) with feedback the output data of the controller and the associated error are produced.

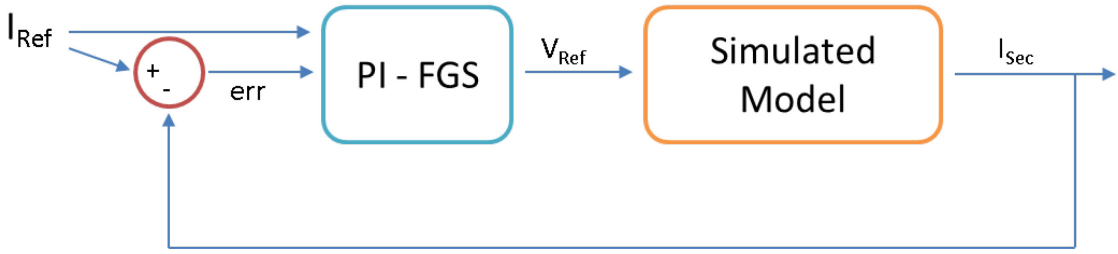

Figure 4.6 the simulated digital control chain.

Different reference current profiles were given as input to the system, obtaining many data series. Through the Matlab toolbox for the ANFIS systems development, a network which reflects the behaviour of the simulated controller was trained, given as inputs values the reference signal and error one. First, the simulated controller was redefined as a Sugeno fuzzy inference system (FIS), keeping its fuzzyfied inputs and outputs. Subsequently, the data collected from the simulations were used in the Neuro-fuzzy network training through the aforementioned toolbox using the back-propagation algorithm. The network based on the parameters derived from the learning process was then implemented in C++. The programming language choice was driven by the need for compatibility with the existing system where the new control logic is hosted. The development of the Neuro-fuzzy network was carried out by an *ad hoc* implementation; this choice is justified by the low complexity of the computed network. The inputs are easy to be fuzzyfied through functions mapping the

trapezoidal membership functions for the I_{Ref} input, while the error is forwarded without being fuzzyfied. Regarding the rules implementation, the complexity is limited by the presence of just one rule for each domain partition. Not being necessary to compute any *T-Norm* operator since there is just a single condition in the antecedent (i.e. the "IF" part) of each rule, the implementation is reduced to the calculation of the linear combination of the parameters learned by the ANFIS and the inputs. The network output is the sum of the output of each individual branch multiplied by the truth value obtained from the membership function computed at the first layer. The control logic that follows this implementation was placed inside the remote monitoring system of the measurement station.

References

[23] T. Takagi, M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, January/February 1985.

[24] Fuzzy Scheduling Control of a Power Plant, Raul Garduno-Ramirez, Student Member, IEEE and Kwang Y. Lee, Senior Member, IEEE. Department of Electrical Engineering, The Pennsylvania State University University Park, PA 16802 USA.

[25] P. Arpaia, L. Bottura, G. Montenero, S. Le Naour, "Performance improvement of a measurement station for superconducting cable test", *Review of Scientific Instruments*, vol. 83, 095111, 2012.

[26] Arnulfo Rodriguez-Martinez, Raul Garduno-Ramirez, and Luis Gerardo Vela-Valdes, "PI Fuzzy Gain-Scheduling Speed Control at Startup of a Gas-Turbine Power Plant", *IEEE Transactions on Energy Conversion*, vol. 26, no.1, pp. 310-317, March 2011.

Chapter 5

Remote control system

5.1 Requirements analysis

The remote control of the superconducting transformer at the FRESCA test station requires the implementation of the communication between the user interface (i.e. *LabView* client) and the remote control system. This communication has to flow through the FESA class. The communication is TCP-UDP based and the data to be handled have to take into account the special purpose of the system. The particular purpose of the system requires a careful requirements analysis both on the monitoring system side and on the FESA interface one.

5.1.1 Front-end software architecture

Due to the particular nature of FESA and its aims of robustness and safety, some steps and pre-requisites regarding permits, licenses and equipment has to be fulfilled in order to start the real design and coding process [2]. The first step to be followed is to get an *AFS* account. The next step involves the FESA support group. In particular, an *NFS* account, FESA development account. FESA developer has to send a request for a H*yper-V* virtual machine on the Technical Network (TN) through the following web site. To access the virtual machine a remote desktop client, like *NX client*, can be used. Once the virtual machine is available the NFS account can be enabled and the design in FESA can start. Using FESA on virtual machine only does not enable to deploy the designed application. In order to deploy the device class a FEC on the TN has to be requested.

5.1.2 Monitoring system

Before outlining the main requirements of the monitoring system interface, the assumptions made both on environment and on the System are presented. This step is essential in the requirements definition process; it is constitutes a starting point marking the beginning of the specifications analysis.

5.1.2.1 The Environment

All the assumptions made on the environment are reported. The environment includes all the equipment and the actors that have any sort of interaction with the System.

- The *Operator* (OP) is an expert in the FRESCA test station and has knowledge on how the System works.
- All the hardware devices involved are turned on and all the connection are plugged in and checked.
- The network connection between FRESCA test station and the System is established; in particular the FRESCA station is connected through the TN.

5.1.2.2 The System

The assumptions regarding the System are outlined.

- The System is connected to the TN.
- All the devices related to the systems are turned on and operative.
- The OP interface accesses the system through a FESA class.

5.1.2.3 Logic Model

A formal specification of the main contexts, in which the System will operate, is given. This specification is formalized using logical predicates for representing the main steps through which the System will handle the requests that will be submitted. It has to be pointed out that, from a strictly logical point of view, the description is incomplete, since it allows instances of the System which are not covered in the normal work context. However, it is necessary to remark that they are not structural defects, but logical closures omitted to avoid a

redundant notation. The scenarios described are the *Cycle definition*, *System on-line*, *Measurement and control*.

- **Cycle definition:** each cycle must be defined as a sequence of segments; the System does not allow empty cycles.

  c is a Cycle instance:

  $$\text{Cycle}(c);\ (5.1).$$

  Each Segment is associated to a Cycle:

  $$\text{Segment}(s, c) := s \text{ is a Segment of the Cycle } c;\ (5.2).$$

  Each Segment occupies a fixed position in the Cycle:

  $$\text{SegPlace}(s, c, n) := n^{th} \text{ Segment of the Cycle } c,\ n \in \mathbb{N};\ (5.3).$$

  Each segment has a current starting value which is mapped as a natural number:

  $$\text{StartVal}(s, v_s),\ v_s \in \mathbb{N};\ (5.4).$$

  Each segment has a current end value which is mapped as a Natural number:

  $$\text{EndVal}(x, v_e)\ v_e \in \mathbb{N};\ (5.5).$$

  The last Segment of a Cycle does not have any successor:

  $$\forall s,c\ \text{LastSeg}(s, c) \Leftrightarrow$$

$$\neg\exists s_2\ (\mathrm{SegPlace}(s, c, n_1) \wedge \mathrm{SegPlace}(s_2, c, n_2) \wedge (n_2 > n_1));\ (5.6).$$

Every Cycle has a starting segment, denoted by the place number one and, moreover it is unique:

$$\forall c\ \exists!\ s\ (\mathrm{SegPlace}(s, c, n) \wedge (n = 1));\ (5.7).$$

Every Cycle must begin from an initial value of zero:

$$\forall c\ \exists s\ (\mathrm{SegPlace}(s, c, n) \wedge (n = 1)) \Rightarrow \mathrm{StartVal}(s, 0);\ (5.8).$$

The last Segment for every Cycle must end with a final value of zero:

$$\forall s, c\ \mathrm{LastSeg}(s, c) \Rightarrow \mathrm{EndVal}(s, 0);\ (5.9).$$

Each Cycle is composed, at least, by one Segment:

$$\forall c\ \mathrm{Cycle}(c) \Rightarrow \exists s\ \mathrm{Segment}(s, c);\ (5.10).$$

Every Segment in the same Cycle is consecutive without overlapping another one in the same Cycle (The *order relation* is given by *N elements):*

$$\mathrm{Segment}(s_1, c_1) \wedge \mathrm{Segment}(s_2, c_1) \wedge$$
$$\mathrm{SegPlace}(s_1, c_1, n_1) \wedge \mathrm{SegPlace}(s_2, c_1, n_2) \Rightarrow (n_2 \neq n_1),$$
$$\forall\ n_1, n_2 \in \mathbb{N};\ (5.11).$$

- **System on-line:** all the conditions to be fulfilled for the System to be on line and ready to accept a request are illustrated:

x is a System instance:

$$\text{System}(x); \quad (5.12).$$

The main flow that leads the System to the on-line status passes through the Hardware and Software check.

$$\text{SystemOnLine}(x) \Rightarrow$$
$$\text{HardwareCheckOk}(x) \wedge \text{SoftwareCheckOk}(x); \quad (5.13).$$

In order the Hardware check to end successfully, the Devices check and the Logic control have to return no problem.

$$\text{HardwareCheckOk}(x) \Rightarrow \text{DevicesOK}(x) \wedge \text{LogicControlOK}(x); \quad (5.14).$$

The Logic Control has to ensure that no *Quench_E* detection, calibration to be done and all the Physical Signal to be checked:

$$\text{LogicControlOK}(x) \Rightarrow \text{NoQuenchEDetected}(x) \wedge \text{CalibrationDone}(x)$$
$$\wedge \text{PhSignalCheck}(x); \quad (5.15).$$

- **Measurement and Control:** in this scenario a running instance of the System and a well-defined Cycle are the preconditions, the System is approaching the process of a Cycle request,

$$\textit{System(x);}$$
$$\textit{System On Line(x);}$$
$$\textit{Cycle(c);} \quad (5.16).$$

In order the System, already in the on-line status, to start the Test Cycle, a light hardware control has to return no *Quench_E* detection signal:

$$\mathrm{StartTest}(x, c) \Rightarrow \mathrm{SystemOnLine}(x) \wedge \mathrm{Cycle}(c) \wedge \mathrm{NoQuenchEDetected}(x); \quad (5.17).$$

The normal test Cycle flow produce a control value according to the current read on the Secondary of the Transformer, after that a *Quench_1* control returns no quench signal:

$$\mathrm{TestCycle}(x,c) \Rightarrow \mathrm{StartTest}(x, c) \wedge \mathrm{ControlValue}(x, c) \wedge \mathrm{NoQuenchOneDetected}(x, c); \quad (5.18).$$

5.1.2.4 Functional requirements

All the requirements derived from the previously formalized specifications and assumptions are listed. First, those representing the functional requirements for all the System are presented, and then the hardware required for the on-field deployment is indicated.

- The main layer of the application has to run on the FRESCA measurement station [3] (i.e. the instrumentation rack), the client application runs on the OP terminal.
- In order to access to the System on the FRESCA station a FESA class is needed.
- Sampling frequency constraint: every measure and control loop iteration must end according to the sampling frequency of 20 Hz.
- The data to be send through the output stream as to be formatted according to a well-defined standard.

- All the signals reporting hardware events (i.e. Quench signal) have to be read after the generation and the forwarding of every sample and they cannot be triggered automatically from the boards.
- For all the check routine, at every level, if it does not end successfully after the third try, the System comes into a *Fatal Error* state and it is halted.
- Handling Hardware Errors: a distinction has to be made between *Quench_E,* to be handled as soft error, and Quench_1 which is a particular state of the System. The latter indeed, has to be faced by a special purpose discharge routine.
- The System can perform one, and only one, Cycle per time.
- All the Cycles must start from an I_{ref} value of zero and terminate again with a value of zero.

As described in chapter 3, for the development of a FESA interface several hardware requirements have to be fulfilled, in particular

- a *Hyper-V* virtual machine on the TN, to access and launch the Front-End, is needed;
- a FEC on the TN has to be obtained, in order to run the FESA class;
- accounts and the permission to develop on FESA framework have to be guaranteed.

5.1.2.5 Use Cases

The diagram of the use cases for System modelling is presented. A brief description of the cases is followed by the usual specification of the single cases, reporting for each of them the pre-conditions, the interaction flow and finally the post-conditions. The use cases diagram (Fig. 5.1) has the aim to show all the possible interactions among the use cases and the actor (i.e. the OP) that will lead the System in all its possible working state. The OP is in charge of the System start-up, as described in *SystemStartUp* use case; once the System is running, the OP can send a Cycle request as shown in the *TestCycleOk* use case, where an error free Cycle request processing is exposed. In the aforementioned use cases

some errors can occur, as a consequence the System will be driven to a different use case according to the type of error detected. In particular, the Quench_1 state handling is described in the *Quench_1* use case. There, like in all the other states, a light hardware check is performed and a Quench_E soft-error can be detected, this possibility is covered by the *Quench_E* use case. As mentioned in the functional requirements, for every kind of error if the recovery routine does not end successfully after the third attempt, a *Fatal error* occurs and the System is halted. This scenario is explained in the *Fatal Error* use case, which is marked as *final* since the System can halt at the end of this case.

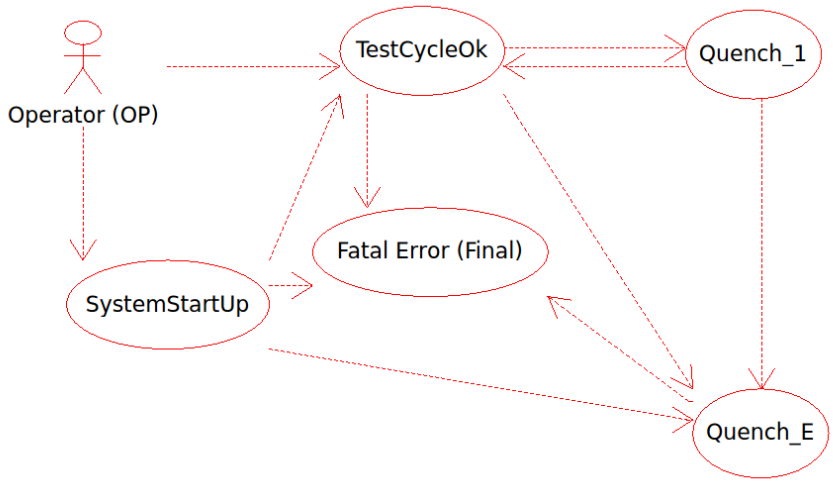

Figure 5.1: Use cases diagram for the monitoring system.

The use cases are then listed, for each of them the precondition, the operations and the post conditions are reported.

System start-up

Notes

1. Only one OP at time can be logged-in the System.
2. Even if the System can be accessed only from trusted machines, authentication at the beginning of every work session is required.

Actors

1. The Operator (OP)

Pre-conditions

1. All the hardware devices (FEC, FRESCA measurement station and OP computer) are turned on and operative.
2. All the devices are connected to the TN.
3. The System is installed correctly both on server layer and on the OP remote computer.

Interaction flow

1. The OP starts the System both on FRESCA station-layer and on OP remote computer.
2. The System shows the authentication form to the OP.
3. The OP is requested to submit is credentials.
4. One among the following possibilities can occur:
   - The credentials submitted are valid:
     (a) The System goes on through the work session.
   - The Credentials submitted are not valid:
     (a) The System sends the OP an error message.
     (b) The System shows again the authentication form.
     (c) The authentication is repeated until valid credentials are not submitted.
5. The System performs the first hardware check, that includes:
   - Devices control;
   - One among the following possibilities can occur:
     • The device control reports no error.
     • The device control reports one or more error:
       (a) The System performs again the Device control;
       (b) The steps from *Device Control* will be repeated for three times, after that a Fatal Error will be triggered to the OP *[Fatal Error detected]*.
   - Logic control of the station:
     • Quench_E detection:
     • One among the following possibilities can occur:

1. No Quench_E is detected.
2. A Quench_E signal is read:
    (a) The System performs the actions for the Quench_E *[Quench_E detected]* recovery.

• Manual calibration of the *FDI* gain.

• Physical signals check.

• One among the following possibilities can occur:

• The Physical signals check reports no error.

• The Physical signals check reports one or more error:

(a) The System performs again the Physical signals check;

(b) The steps from *Physical signals check* will be repeated for three times, after that a Fatal Error will be triggered to the OP *[Fatal Error detected]*.

6. The System performs the first Software check, that includes:
    - Run the Server module.
    - The Server module is on-line.
7. The System is ready to receive Cycle request.

Post-conditions

One (and only one) among the following cases will occur:

- Success:
    1. The System is correctly started up.
    2. The Op correctly accessed the front end interface.
- Error:
    1. The System sends the OP a message reporting the error.

Test Cycle: NO Error

Notes

1. The OP keeps track of the Cycles sequence.

Actors

2. The Operator (OP)

Pre-conditions

1. The System is correctly running both on FRESCA station and on OP computer.
2. The System is on line and ready to receive Cycle requests.

Interaction flow

1. The OP defines a Cycle.
2. The OP sends the Cycle request to the System.
3. The System receives the request.
4. One among the following possibilities can occur:
    - The Cycle was successfully designed and the System sends an ACK to the OP.
    - The Cycle was not well defined:

    (a) The System sends a message to the OP, reporting the error in the Cycle definition.

    (b) The OP has to submit a new Cycle request.

    (c) The steps from *The OP defines a Cycle* will be repeated for three times, after that a Fatal Error will be triggered to the OP *[Fatal Error detected]*.
5. The System performs a light hardware control.
6. One among the following possibilities can occur:
    - No Quench_E is detected.
    - A Quench_E signal is read:

    (a) The System performs the actions for the Quench_E *[Quench_E detected]* recovery.
7. The System estimates the offset.
8. The test Cycle begin:

    (a) The System acquires the I_{mes}.

    (b) The System compares the I_{mes} value with I_{ref} value.

    (c) The control logic computes the V_{ref} value.

    (d) The sample is sent to the OP terminal.

    (e) The Quench detector is read by the System.

(f) One among the following possibilities can occur:

2. No Quench_1 is detected.
    i. The System continues running the current Cycle.
3. A Quench_1 is detected:
    i. The System sends a message to the OP, reporting the Quench_1 state.
    ii. The System runs the routine for Quench_1 handling *[Quench_1 detected]*.
    iii. End of test Cycle.

(g) The System runs a light hardware control.

(h) One among the following possibilities can occur:

4. No Quench_E is detected.
5. A Quench_E signal is read:
    i. The System performs the actions for the Quench_E *[Quench_E detected]* recovery.

(i) The steps from the point (a) are repeated until the end of the Cycle.

9. The Measurement and Control phase is over.
10. The System is ready to receive a Cycle request.

Post-conditions

One (and only one) among the following cases will occur:

- Success:
    1. The test Cycle is correctly performed.
- Quench_1:
    1. The System shows the OP a message reporting the state.
- Quench_E:
    1. The System shows the OP a message reporting the error.

Quench_1 detected

Notes

1. The Quench_1 detection is performed at any Measurement and Control iteration by means of reading the Quench_1 detection state variable.

Actors

1. The Operator (OP).

Pre-conditions

1. The System began a test Cycle.
2. The output sample is being sent to the OP terminal.

Interaction flow

1. The Quench detector variable is read by the System.
2. One among the following possibilities can occur:
   - No Quench_1 is detected.
     (a) The System continues running the current Cycle *[TestCycleOk]*.
   - A Quench_1 is detected:
     (a) The System sends a message to the OP, reporting the Quench_1 state.
     (b) The System runs the routine for Quench_1 handling:
         i. The System shut down immediately all the hardware devices.
         ii. The System sets Vref value to zero.
         iii. The System runs the voltage discharge routine.
         iv. The System resets all the Control output.
         v. The System runs a light hardware control.
         vi. One among the following possibilities can occur:
             - No Quench_E is detected.
             - End of test Cycle.
             - A Quench_E signal is read:
               (a) The System performs the actions for the Quench_E *[Quench_E detected]* recovery.

Post-conditions

One (and only one) among the following cases will occur:

- End of test Cycle:
    1. The System ends the current test session and can receive new Cycle request.
- Quench_E:
    1. The System shows the OP a message reporting the error.

Quench_E detected

Notes

- Quench_E error can be detected every time is performed a control on the board in and out of a test Cycle.

Actors

1. The Operator (OP)

Pre-conditions

1. A light hardware control routine is launched by the System.
2. A Quench_E error is detected.

Interaction flow

1. The System resets the Hardware boards.
3. The System waits for 5 seconds.
4. The System reads the Quench_E detection variable.
5. One among the following possibilities can occur:
    - No Quench_E is detected.
        (a) The System is ready to receive Cycle request *[TestCycleOk]*.
    - A Quench_E signal is read:
        (a) The steps from point (1) will be repeated for three times, after that a Fatal Error will be triggered to the OP *[Fatal Error detected]*.

Post-conditions

One (and only one) among the following cases will occur:

- The Error is fixed.

1. The System ends the current test session and can receive new Cycle request.

- The Error cannot be fixed.
  1. The System shows the OP a message reporting the error.

Fatal Error detected

Notes

- The OP has no control when a Fatal Error occurs.

Actors

1. The Operator (OP)

Pre-conditions

1. The Error state that caused the Fatal Error has being tried to repair at least three times.

Interaction flow

1. The System shouts down all the devices.
2. The System sends to the OP a report message.
3. The System halts.

Post-conditions

- The System is halted.

5.2 System design

5.2.1 Monitoring system design and standards

A customized format of data is required for the cycle definition phase and for the communication with the remote control. Other kinds of data to be shared are the system status (updated by the controller) and the samples of the measured current delivered to the user interface layer, as well as the error codes to be forwarded to the client interface.

5.2.1.1 Cycle definition

The cycle is composed of several segments. Each of them represents a specific current profile. The string of characters codifying a segment is made up by a head followed by several parameters. The head contains a unique identifier of the segment. The number of the parameters is related to the segment type, and it is *a priori* fixed. According to [4], four different current profiles can be described by four kinds of segment. A segment can be defined using up to five parameters (Table 5.1). Each segment's head must have the proper identifier, such as CP, LR, PP or PLP, in order to represent uniquely the function.

	Segment parameters					
Segment Type	***I start* (A)**	***I stop* (A)**	***Time Plateau* (s)**	***Acceleration* (A/s^2)**	***Deceleration* (A/s^2)**	***Ramp Rate* (A/s)**
CP	✓		✓			
LR	✓	✓				✓
PP	✓	✓		✓	✓	
PLP	✓	✓		✓	✓	✓

Table 5.1: Segments and parameters

A segment head is followed by the exact number of required parameters. A segment has to be codified according to the following expression:

$$SegID \cdot (\text{'/'}\ PARAM)^{+} \quad (5.19)$$

Where $SegID \in \{\text{"CP", "LR", "PP", "PLP"}\}$, and *PARAM* is a string that maps either a real or an integer number according to the range of values of each parameter. As real number, the *PARAM* string can have the "." *(dot)* character to separate the integer part from the decimal one. The total number of digit that can be represented has not to exceed the ***double***-type limits. The *PARAM* range depends on the specific parameter to be transmitted. The ranges and number formatting required to specify properly a segment are then presented (table 5.2).

Parameter	**Range**	**Formatting**	**Example**
Current		Maximum 2 digits	"23 546. 78" ☑

(*Istart, Istop*)	[0, 34] KA	after the dot "."	"34 897. 214" ☒ "12 400. 235" ☒
Time plateau	[0.05, 1800] s	Maximum 2 digits after the dot ".", the decimal numbers **must** be multiple of "0.05"	"0. 25" ☑ "10. 87" ☒ "647. 45" ☑
Acceleration and Deceleration	(0, 800] A/s^2	Maximum 2 digits after the dot "."	"67. 59" ☑ "800. 87" ☒ "700. 235" ☒
Ramp Rate	(0, 800] A/s	Maximum 2 digits after the dot "."	"67. 59" ☑ "800. 87" ☒ "700. 235" ☒

Table 5.2: Ranges and number formatting for all the parameters

In the segment definition, the head and the parameters are separated by the character '/'. An example of the syntax follows:

"PLP/0.0/34000.0/20.5/15.75/34.3**"**

Different segments can be concatenated according to the order of execution to define a cycle. To face this requirement a cycle can be built following the syntax

$$SegID \cdot (\text{'/'}\ PARAM)^{+} \cdot (\text{'\#'}\ SegID \cdot (\text{'/'}\ PARAM)^{+})^{*} \quad (5.20)$$

Where the symbols *SegID* and *PARAM* assume the same meaning as in 5.19. It is worth to note that the segments, listed by the final execution order, are separated by the special character '#'.An example is reported

"CP/0/5 # **LR**/0/1000.0/20.0 #**PLP**/0.0/30000.0/20.5/15.75/34.3 # **LR**/23000.0/0.0/20.0**".**

A blank character can occur between different words, even if it is not recommended. A blank cannot be placed inside a single word or number.

5.2.1.2 Status and errors definition

In this subsection the codes used by the system are reported. In particular, the status configurations and the errors are identified and their codes are detailed. First the status definition is presented; Table 5.3 lists the System status codes. Each status is identified by a numeric code and a label defined in the System configurations. The table is completed with a brief description of every status.

Code	Macro Name	Description
100	**CHECK_OK**	All the preliminary check completed with no error.
200	**SYS_ON_LINE**	The System is on line (Even with a hardware check error).
300	**APT_RQT**	The system is ready to accept a “Status” request.
350	**STAT_UPD**	The Status is updated to the client, cycle request can be received.
400	**RQT_RCVD**	The cycle request has been received by the System.
500	**CYC_VLD**	The cycle submitted is valid.
550	**CYC_BLT**	The cycle samples have been built.
560	**SYS_CAL**	The system is performing the calibration.
570	**CYC_RUN**	The cycle is running.
580	**QUENCH**	A Quench occurred during the cycle execution.
590	**LIM_OVTKN**	The voltage limits have been overtaken.
900	**FRESCA_ERROR**	An error occurred during the test session.
999	**FATAL_E**	A failure as occurred. The System must be halted.

Table 5.3 all the possible status of the System.

These states are related to each other according to the following Finite State Automata (Fig. 5.2). From the initial state other three states can be reached, according to how the device check routine ends. When all the devices are successfully checked, the state reached is **CHECK_OK**; when an error occurs there are two possibilities depending on the nature of the error: if it is a hardware error (failure), the system goes into **FATAL_E** state; otherwise it is driven to the

FRESCA_ERROR state. From the last two error states only the *FINAL* state can be reached but, the System is still able to handle request by reporting the current state and the code of the specific error occurred, before being halted. The **CHECK_OK** state is the first along the path to follow in order to run a test cycle. It is followed by the **SYS_ON_LINE** state and soon after by the **APT_RQT**. Since it may come a "close" request too, from the **APT_RQT** state also the *FINAL* state can be reached. Before submitting any Cycle string, it is necessary to check the status of the system, once this operation has been completed; the System reaches the **STAT_UPD** state. At this stage a Cycle request should be submitted, this action leads to the **RQT_RCVD** state, from which the cycle execution process begins. When the next step, the Cycle validation, ends successfully, the **CYC_VLD** state is gained. The Cycle building is the next operation performed, when it ends without error, **CYC_BLT** state is on. The Cycle running is approaching but, before going ahead, it is necessary to perform the System calibration. The Cycle can be run and before this operation begins the System moves to **CYC_RUN** state. During this part of the working session, a situation related to the voltage limits can occur; every time this problem happens the **LIM_OVTKN** state is reached; the System goes back to **CYC_RUN** state at the following measurement iteration. A test Cycle can be interrupted by a cable Quench phenomena, to handle this situation the System will move to the **QUENCH** state. Afterwards, if the Quench handling routine ends successfully, the System goes back to **APT_RQT** state.

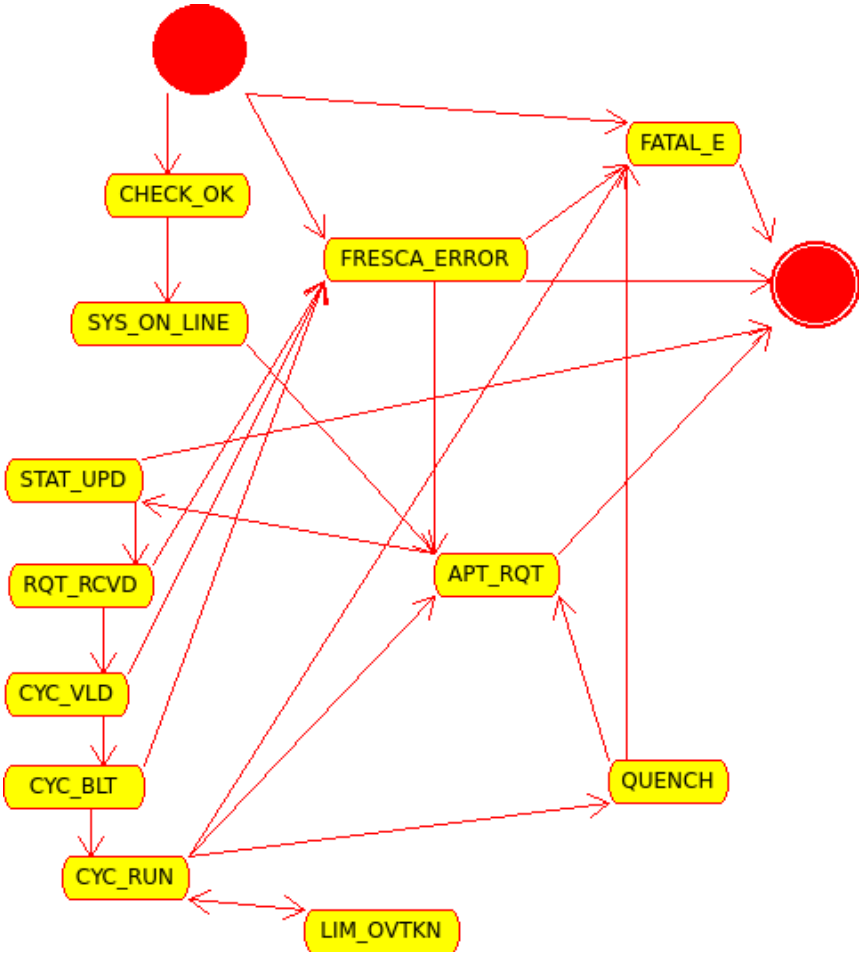

Figure 5.2: finite state automata of the System states.

From the Cycle submission to the Cycle execution, several errors can occur. When the error detected is a simple error the System moves to **FRESCA_ERROR** state. It is worth to note that from this state is still possible to continue the work session by checking twice the System state, as shown in the next section. When a hardware failure is triggered the System is driven to the **FATAL_ERROR** state. After the status definition, the list of errors that can occur during a test session is given. For each error the numeric code with the corresponding label and a brief explanation is reported. In order to have a clear picture of the entire situation, the codes are dived into four categories according to the nature of the exception that reports the error. The *Hardware Check* error group contains the codes concerning the errors related to the preliminary checks.

The former table presents the possible errors occurring during the initial device check.

Code	Error Name	Description
600	**LOG_NOPEND**	Error occurred during the log file creation.
601	**DAQ_INIT_FAIL**	NI PXI-6281 Hardware problem, initialization failed.
602	**DAQ_CNT_RESET**	NI PXI-6281 cannot even be reset, hw or sw problem.
603	**DAQ_CNT_ENBLD**	NI PXI-6281 cannot be enabled.
604	**DAQ_PRT_QNCH**	NI PXI-6281 Ch P0.0 T3: read anomalous signal (quench detected).
605	**DAQ_ERREAD_0**	NI PXI-6281 Ch P0.0 T3: error read.
606	**DAQ_ANOM_STAT**	NI PXI-6281 Ch P0.1 T2: read anomalous signal (acquisition system ready).
607	**DAQ_ERREAD_1**	NI PXI-6281 Ch P0.1 T2: error read.
608	**DAQ_ERR_OUT**	NI PXI-6281 Ch P1.0 T1: output error.
610	**TTI_CNT_RESET**	TTi PL330 cannot be reset.
611	**TTI_BAD_STAT**	TTi PL330 hardware problem.
612	**TTI_COM_PROB**	TTi PL330 GPIB communication problem.
620	**LSPS_CON_GPIB**	Lake Shore 622 GPIB communication problem.
621	**LSPS_BAD_STAT**	Lake Shore 622 hardware problem: unable to recover bad state.
622	**LSPS_SET_VLIM**	Lake Shore 622 software problem: unable to set 5 V voltage limit.
630	**TBD_COM_HWD**	NI PXI 6682 either PXI communication or hardware problems.
631	**TBD_COMM**	NI PXI 6682 PXI communication problem: communication failed.
640	**FDI_ CNT_RESET**	FDI v3 PXI communication problem: enable to reset FDI.
641	**FDI_DSP_FAT**	FDI v3 DSP error: FATAL error, DSP failure boot.
650	**ACQ_ZERO**	NI PXI 6682 zeroing of the analog outputs error.
651	**ACQ_TTI**	TTi PL330 heaters not connected.
652	**ACQ_TIME**	FDI v3 test acquisition failed (Timeout).

Table 5.4: Hardware Check errors: device checks possible errors.

The latter lists the error that can be detected during the run of a test Cycle.

Code	Error Name	Description
661	RUN_FDI_TO	FDI timeout reading: run the checking routine.
662	RUN_TTI_HF	Heaters fault (TTi PS): run the checking routine.
663	RUN_DAQ	NI PXI 6281 fault: run the checking routine.
664	RUN_DEV_SET	One of the Device cannot be set properly: run the check routine.
665	RUN_DEV_CNF	One of the Device cannot be configured properly: run the checking routine.
666	RUN_UNKN	Unknown Error: run the checking routine.

Table 5.5: Hardware Check errors: test cycle possible errors.

The *Connection* group collects the errors related to the communication between the System and the FESA class layer.

Code	Error Name	Description
701	RECEIVE MSG ERROR	Due to an error in the string receiving function.
702	INVALID REQUEST HEADER	The header of the received request is not recognized, hence is not valid.
703	SYSTEM STATUS ERROR	This error occurs when an inconsistency is encountered in the system status update.
704	SEND MSG ERROR	Due to an error in the string sending function.

Table 5.6: Connection errors.

In the *Cycle Definition* group lists the errors concerning the cycle request in the Table 5.7. This group describes the errors from the dispatch phase to the cycle validation one.

Code	Error Name	Description

801	**INVALID REQUEST SIZE**	Occurs when an inconsistency in the cycle request is found. Can be reported when the request dispatched is empty.
802	**NOT ALL SEGMENT DEFINED**	During a cycle definition, the number of the recognised segments doesn't match the segments in the request string.
803	**SEGMENT DEFINITION WRONG**	An invalid segment is found while performing the validity check on the cycle. Usually is referred to the first or the last.
804	**INVALID SEGMENT INDENTIFIER**	During the cycle request handling, an invalid segment identifier is found in the request string.
805	**CYCLE NOT VALID**	The cycle sent does not match the validity condition [2].
806	**CHECK THE STATUS FIRST**	Reported when a cycle request is submitted without a previous status update.

Table 5.7: Cycle Definition errors.

5.2.1.3 Classes design

The analysis of the system requirements and the data definition lead to the design of the classes implementing the system logic. A conceptual distinction has been operated in design the system modules: The data abstraction is mapped by the *Entity* classes, the logic implementation is stored in the *Manager* modules and the error signals are wrapped and handled through the *Exception* classes. A class diagram for each of these categories details the operation structure and the type of handled data (Fig. 5.3, 5.4, 5.5). It is worthy to note that for each class are outlined the instance variables with their relative type and scope (i.e. '-' for private, '+' for public and '#' for protected) and the default initialization value (e.g. *iStart*: double = 0). The methods are also presented with their input and output parameters and the relative scope using the same symbolism of the instance variables.

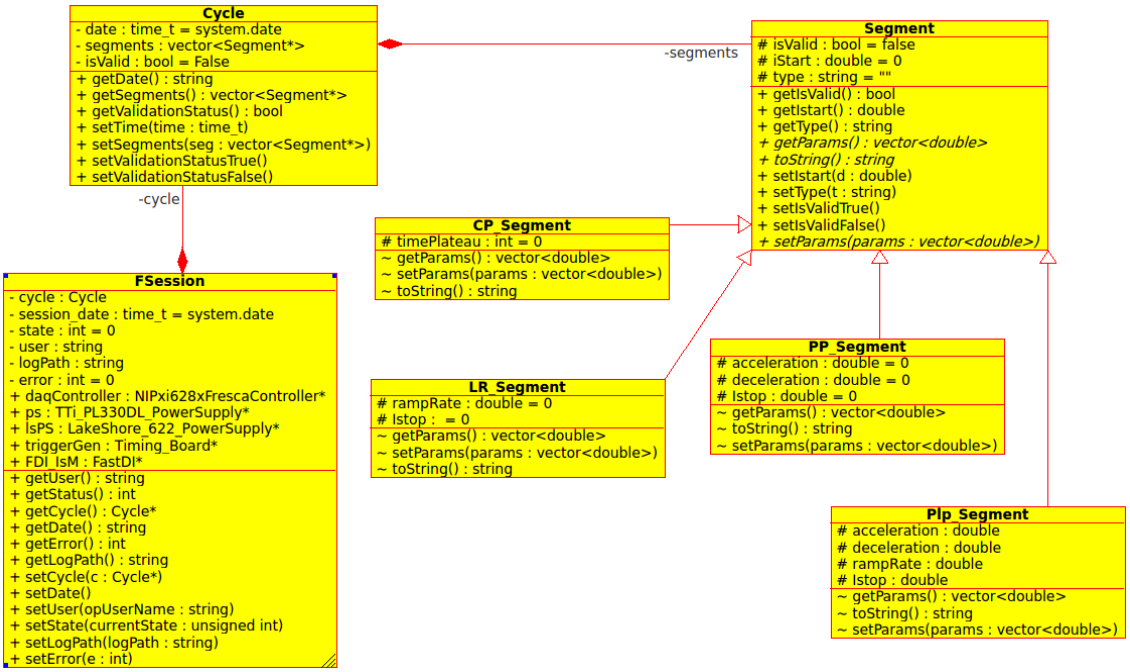

Figure 5.3: Entity Class diagram.

As the Entity class diagram shows, three main entities can be distinguished, the *Segment*, the *Cycle* and the *FSession*. The lowest abstraction level data is represented by the *Segment* class. This unit stores the information of the atomic element of a test cycle: the segment. *Segment* generalizes the common characteristics of each current profile such as a validity field (*isValid*: boolean) used in the Cycle validation, the electric current beginning value (*iStart*: value) and the type (*type*: string). The common implemented methods are the "getters" and "setters" for the abovementioned variables. Other three customizable methods are available in *Segment*, namely the segment parameters getter and setter and the classic *toString* operation. Since there are four different current profiles, four different *Segment* inherited implementations have been created: *CP_Segment*, *LR_Segment*, *PP_Segment* and *PLP_Segment*. According to the description provided in the Cycle definition subsection, every profile has been enriched with the proper parameter among the stop current value (*iStop*: double), current ramp rate (*rampRate*: double), parabolic acceleration (*acceleration*: double) and deceleration (*deceleration*: double). These parameters are set and read respectively by *setParams* and *getParams* methods. They handle a *double*-

type vector to take and return the segment parameters. The entity representing the test run is the Cycle class. The current function is stored in a vector of Segment (*segments*: vector), additional information are the current function validity value (*isValid*: boolean) and the cycle creation date (*date*: time_t). These three variables can be accessed and modified through custom getter and setter methods. The FSession is the highest abstraction level data since it enclose all the information of the test session. It contains the operator identifier (*user*: string), the creation date of the session (*date*: time_t) and the current state of the system (*state*: int). A reference to an instance of Cycle (*cycle*: Cycle*) is also stored as the path of the log file (*logPath*: string), finally the error code eventually returned from a system interaction is kept into the *error* variable (*error*: int). As for all the other entities, the instance variables can be read and set by the usual getter and setter operations. Other special purpose instance variables are stored in the session; they are the singleton class instances abstracting the physical devices of the measurement system. In particular, the data acquisition board controller (*daqController*: NIPxi628xFrescaController*), the *TTi* power supply (*ps*: TTi_PL330DL_PowerSupply*), the *Lake Shore* power supply controlling the current on the transformer primary (*lsPS*: LakeShore_622_PowerSupply*), the timing board (*triggerGen*: Timing_Board*) and the digital integrator for the transformer secondary measured current (*FDI_IsM*: FastDI*). The data of a test session are handled through the *Entity* classes by the *Manger* classes (figure 5.4). The modules where the logic of the application is implemented are the *SessionManager*, *CycleManager*, *MeasureAndControlManger* and the *ErrorManger* for the error treatment. The other Manager classes handle all the triggered exception by forwarding them to the singleton instance of Error Manager (*errManager*: ErrorManager*) referred among their instance variables through the *reportError* method. The logging system is also common to all the managers; the system log is tracked by the *SimpleFileChannel* instance reference from POCO *C++* library [5]. The last mutual aspect of the manager classes is the communication handling through an instance of the *ConnectionHandler* class.

This Class provides all the services and the error treatment support needed for establish a safe connection, both TPC and UDP, with the *FESA* interface.

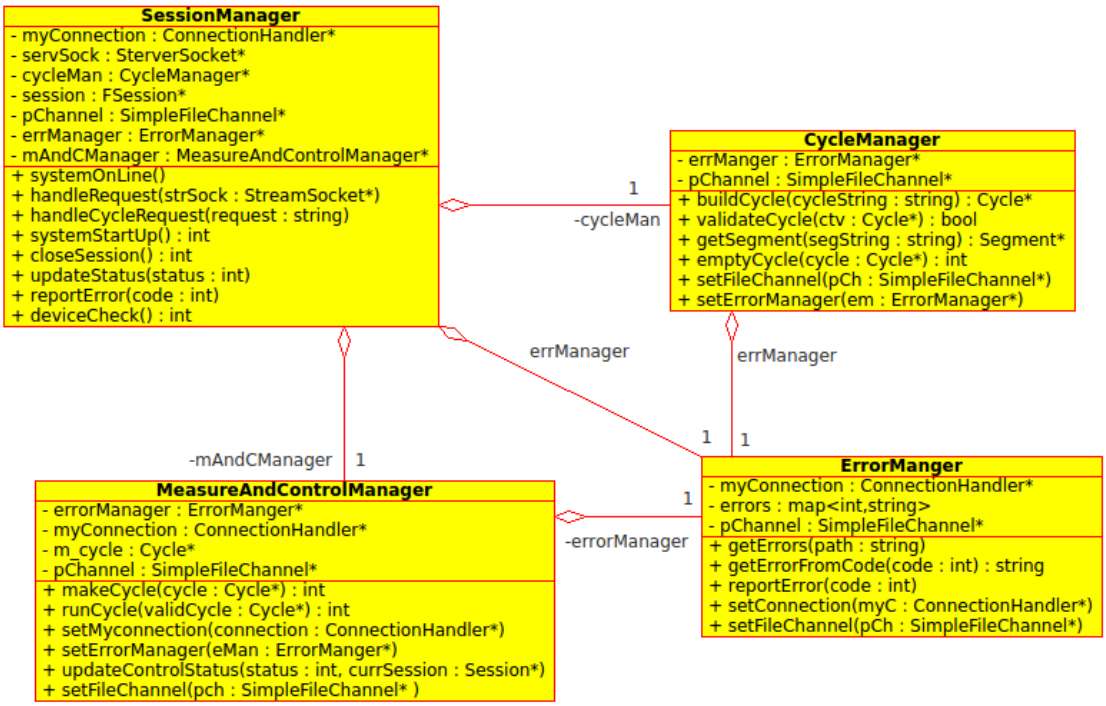

Figure 5.4: Manager Class diagram.

The *CycleManger* class accomplishes the task of current profile building and validation. It takes the string coding the test cycle in the *buildCycle* method and segment by segment, through the *getSegment* operation, it builds a Cycle instance. The resulting cycle is then validated through by *validateCycle* and it can be empty by *emptyCycle* method. The error treatment and forwarding is a task assigned to an *ErrorManager* instance. All the error codes and labels are updated from a configuration file and stored in a map (*errors*: map<int, string>), where the key is the error code (int) and the value is represented by the denomination of the error (string). To retrieve the error from the map the *getErrorFromCode* is used, and to upload the error information from the configuration file, the file path is provided to the *getErrors* operation. Forwarding the error to the remote FESA class is a task accomplished by the *reportError* method. All the operation regarding the effective cycle run and the measurement process are driven by the *MeasureAndControlManager*. The built current cycle, if successfully validated, is

turned into an appropriate sample sequence by the *makeCycle* method. The cycle can be then run through the *runCycle* action, where all the logical operation of measurement and control are implemented. The evolution of the system along all the test process is tracked by the *updateControlStatus* method. The general control of the test station is performed by a *SessionManager* instance. The system initialization is executed first by the *systemStartUp* method, and then from the *deviceCkech* action that completes the physical devices check. The remote connection is then established by the *systemOnLine* method, once the station is able to accept request, the incoming messages are processed by the *handleRequest* operation. Among the received ones, the cycle requests are treated by *handleCycleRequest* action. The work session can be concluded by invoking the *closeSession* action and the status evolution is updated locally and to the remote interface by the *updateStatus* method. The system exceptions, occurring during the normal working flow, are structured according to their nature (figure 5.5).

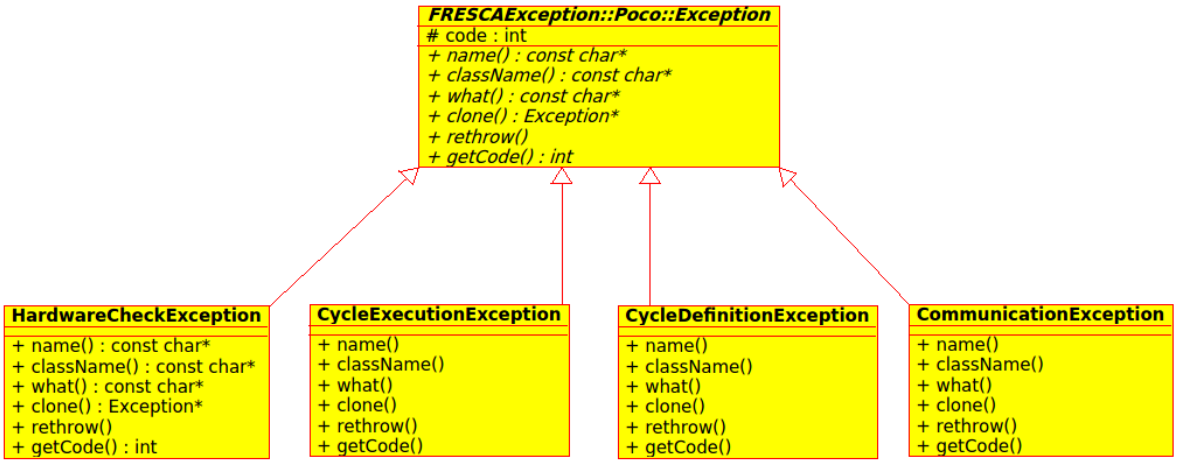

Figure 5.5: Manager Class diagram.

The main base exception class is *Exception* one form POCO library, the *FRESCAException* is a special purpose base class inheriting the operations and the main fields. The faults regarding the physical devices are mapped by the *HardwareCheckException* class, while the errors occurring during the test cycle definition are carried by a *CycleDefinitionException* instance. All the error in the

communication between the test station layer and the remote *FESA* interface are triggered by the *CommunicationException* and the possible faults occurring during the test cycle execution are handled by firing a *CycleExecutionException*. Each class stores the error code (*code*: int), that can be read by invoking the *getCode* method. The *name* action returns a brief description of the exception, while the *className* gives the name of the exception class. The *what* method gives a description of the error triggering the exception, like the *name* one, but it keeps the compatibility with the *std::exception* class. The *clone* action creates an exact copy of the exception; the copy can later be thrown again by invoking *rethrow* method on it.

5.2.1.4 Interaction flow

A view of the general interaction flow between the Lab View client, driven by the operator, and the FESA class is given. The error-free interaction flow sequence diagram is then reported (Fig. 5.6). The preconditions shown in the diagram assume the LabView client and the FESA class to be running, and after the preliminary checks have ended with no error, the System to be in a consistent status. Before submitting a cycle (by setting the *cycleString* property), is mandatory for the Operator to check the status of the System. This action can be performed by setting the *acquireStatus* property to ***true***, through the *SetAcquireStatus* action. Once the status is updated and no has error occurred, the System should be in **STAT_UPD**. This status gives to Operator the possibility to submit a new current cycle by using the *SetCycleString* action.

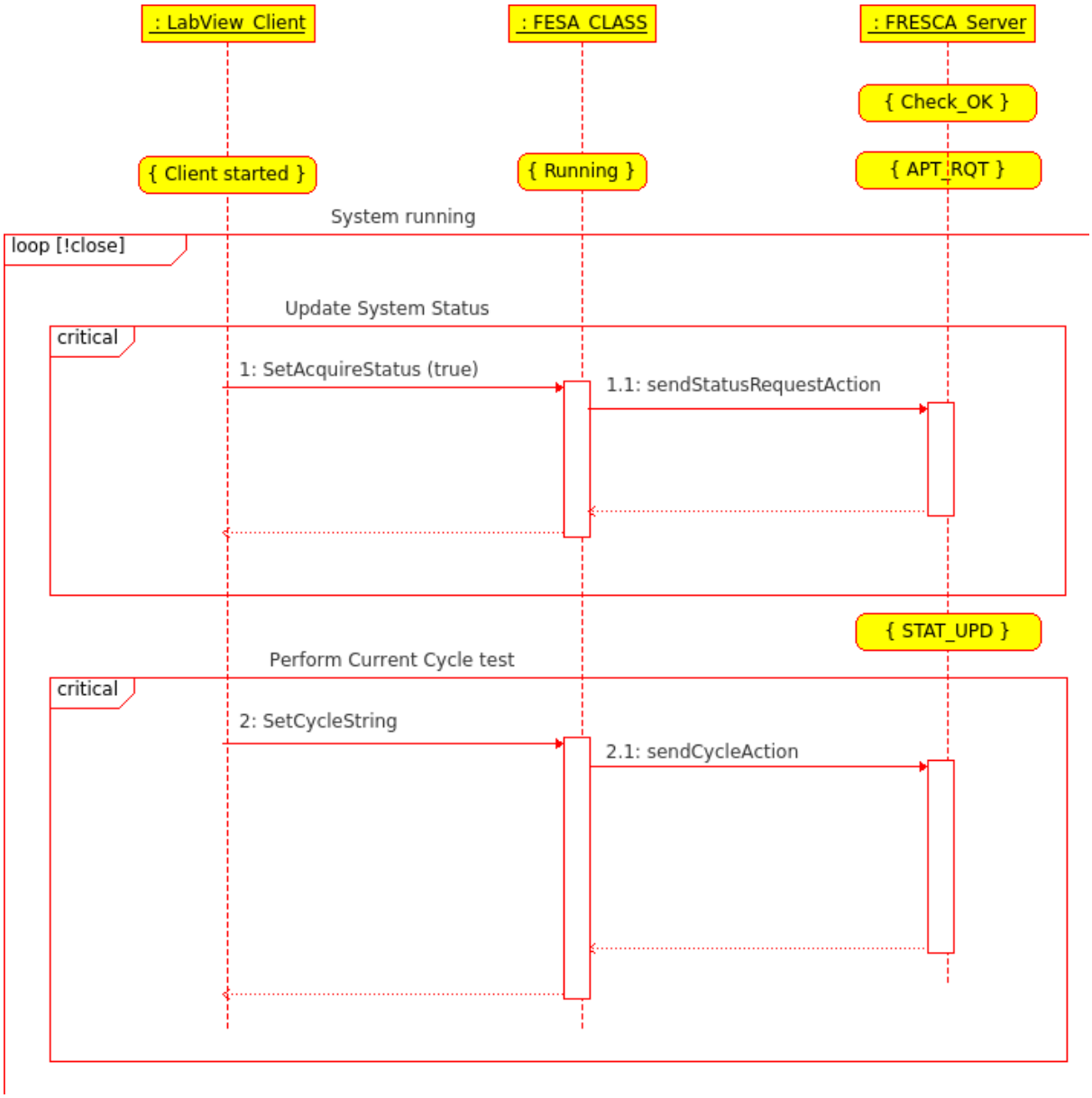

Figure 5.6: Sequence diagram of the error-free state.

When the initial device checks end reporting an error, the System moves to **FATAL_ERROR** state (Fig. 5.7). In this situation the System cannot perform any test Cycle, but the Operator can still ask for the current status.

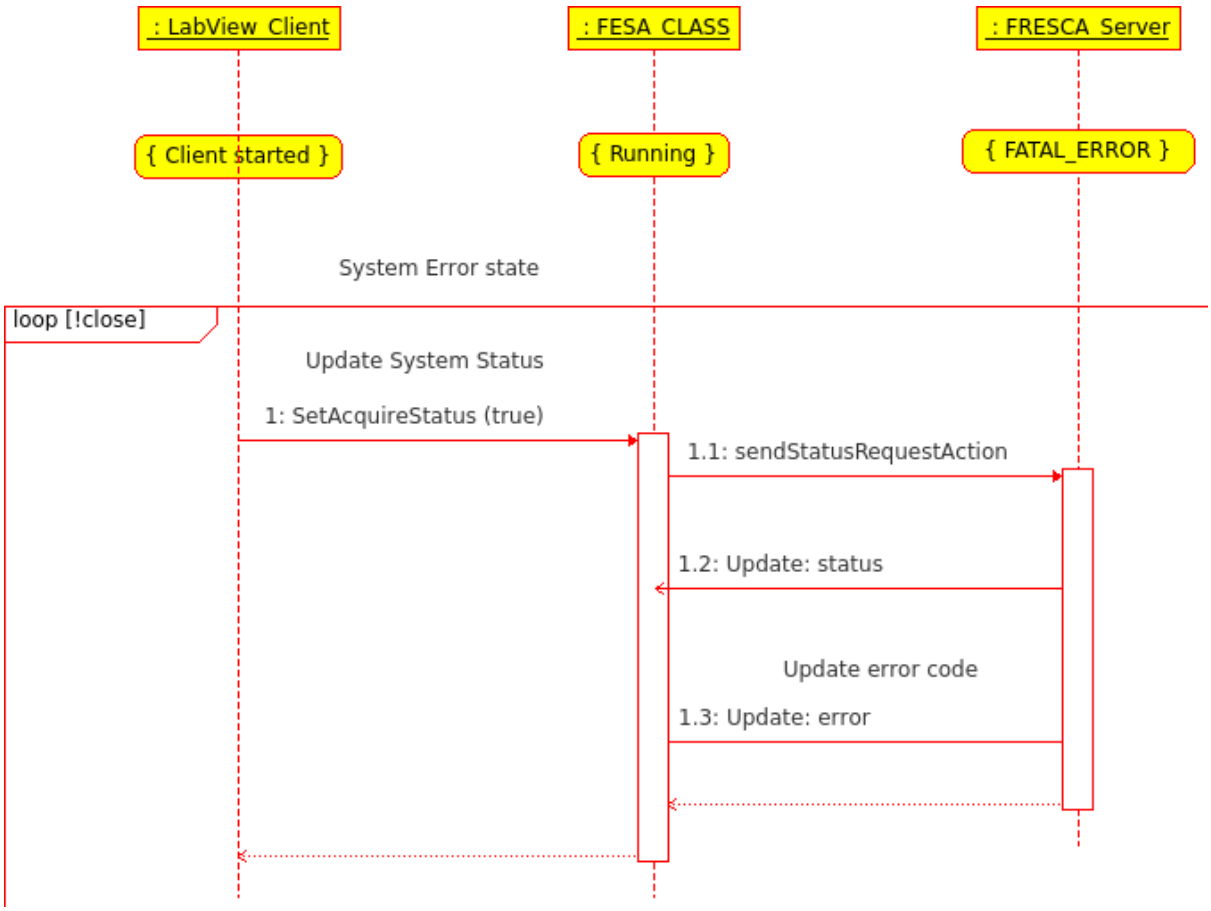

Figure 5.7: Sequence diagram in the FATAL_Error state.

This goal is achieved in the same way as for the error free situation. In this case the System will reply updating the *status* field first and the *error* field with the error code soon after. The System comes into a **FRESCA_ERROR** state after a non-critical error in the communication or in the Cycle building such as a wrong message formatting or an invalid Cycle. Those kinds of error do not require the System to be halted (Fig. 5.8).

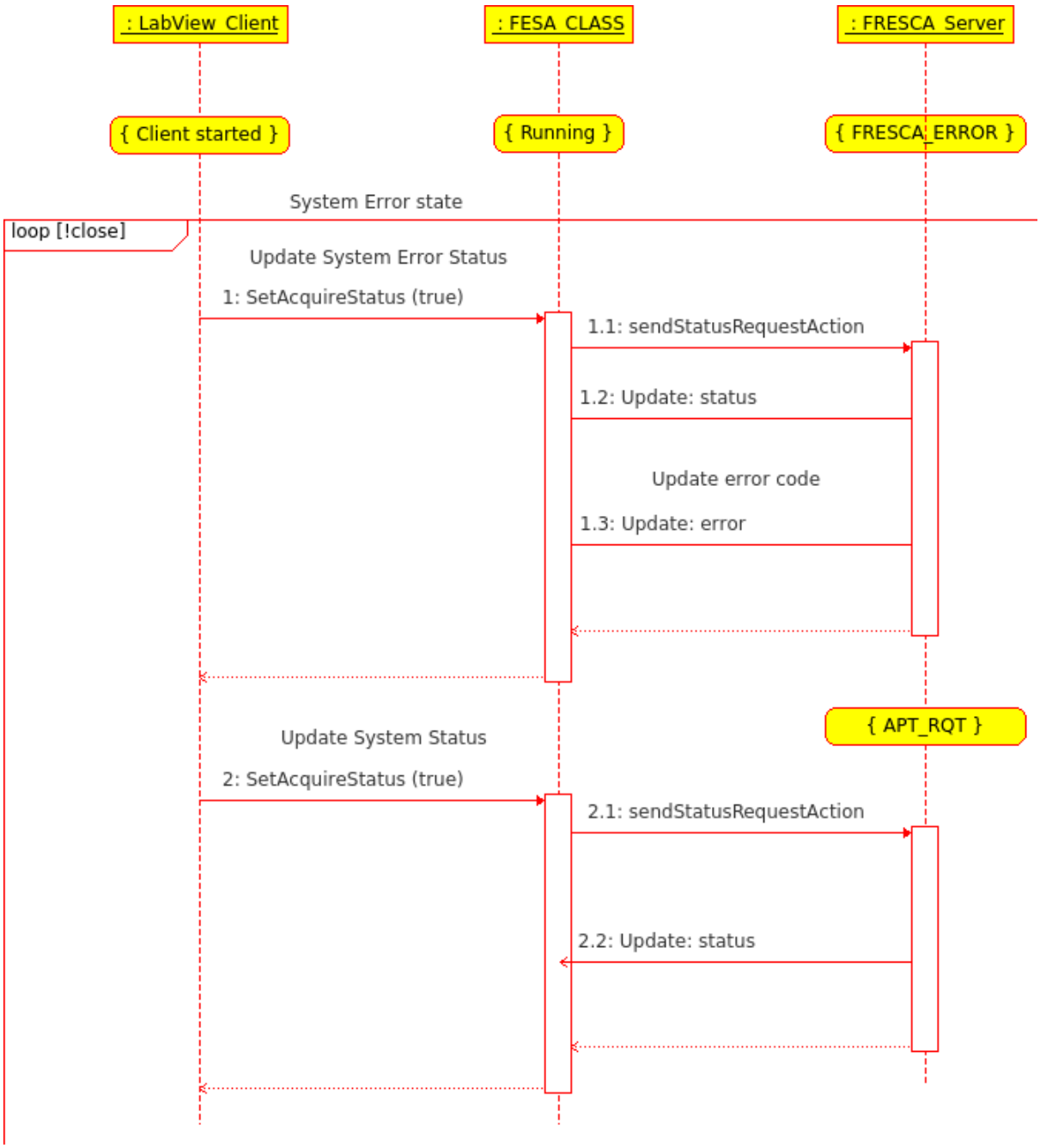

Figure 5.8: Sequence diagram in the FRESCA_Error state

The work session can continue after the following operations. Soon after the error has been triggered the System is in the **FRESCA_ERROR** status, the Operator should check the status and acquire (again) the error code. At this stage the System has changed status to **APT_RQT** and after another status check, another Cycle can be submitted. The last scenario that will be presented is related to the System shutdown (Fig. 5.9).

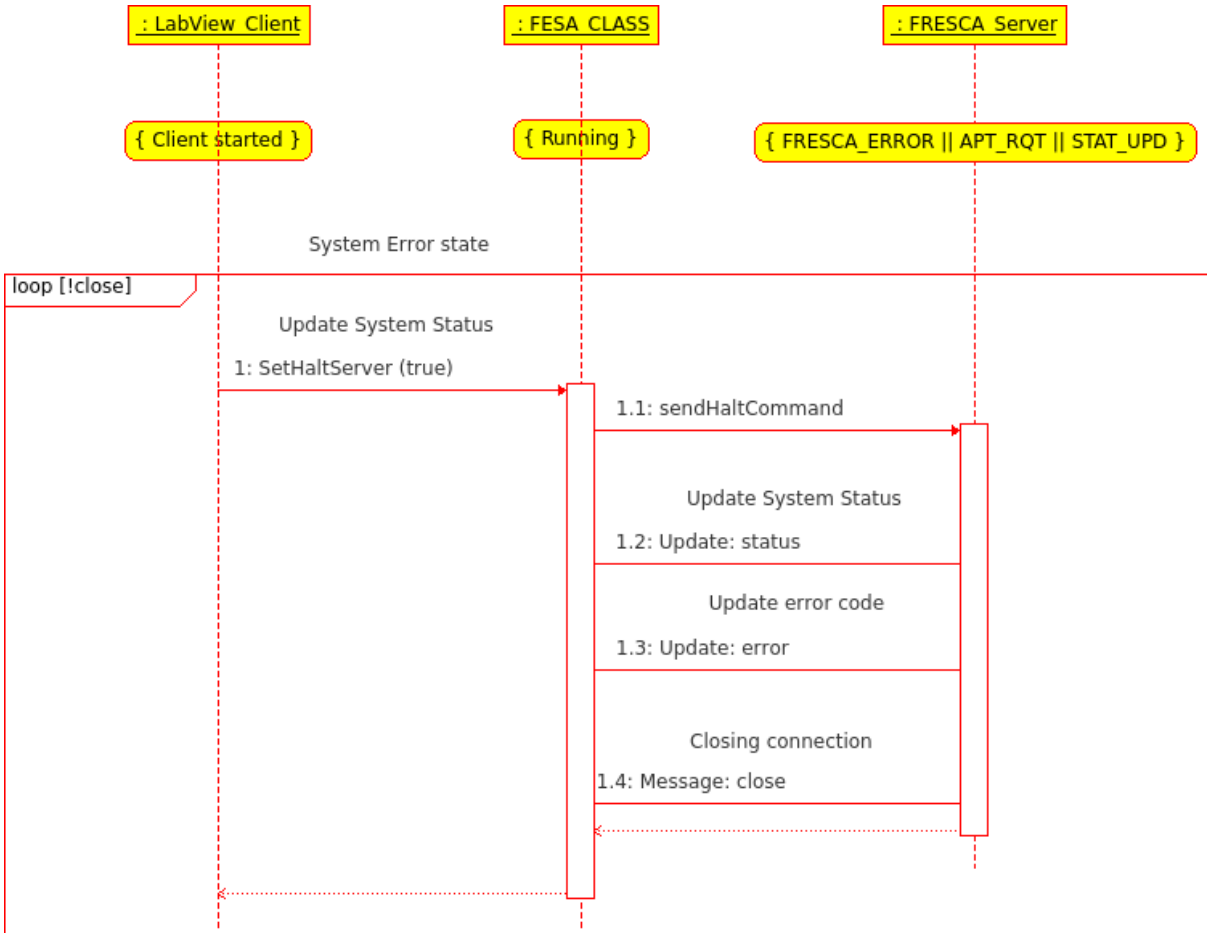

Figure 5.9: Sequence diagram of the System shutdown.

The System can be halted just if it is in an error state (**FRESCA_ERROR** or **FATAL_E**) or both in the **APT_RQT** and **STAT_UPD** state, as shown in the system finite state automata (Fig. 3.20) they are the only states from where it is possible to reach the *FINAL* state. In order to send this command, the Operator has to set the *HaltServer* property to ***true***, through the *SetHaltServer* action. The System will reply by updating the status first and then, if the status is an error one, by updating the *error* property before closing the connection and shutdown.

5.2.2 *FESA* Class design

The FESA class represents the System abstraction, as a device, to the *LabView* client. Through the properties and the Server action is possible to run a test session; the interface of the class is then presented (figure 5.10). The FTS01 class includes both properties and methods (actions).

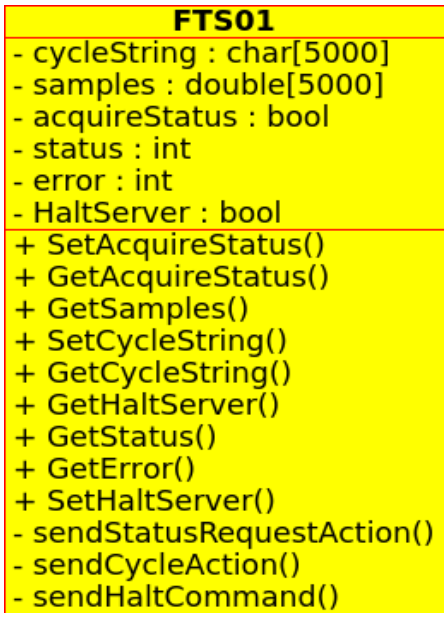

Figure 5.10 FESA class interface

The properties of the class are:

- "**cycleString**". A 5000 *char* type array that is set with the final string that codifies the Cycle. A Cycle string cannot contain more than 5000 characters.
- "**samples**". A 5000 *double* type array that will be filled, according to a *circular buffer* strategy, with the samples measured during the test session.
- "**acquireStatus**". A *boolean* type field to be set to *true* in order to update the state of the System.
- "**status**". An *int* type field updated with the current state of the System.
- "**error**". An *int* type field updated with the error code returned in case of Error occurrence.
- "**HaltServer**". A *boolean* type field to be set to *true* in order to halt the System remotely.

The properties can be set and read when new data are available via the following *Server Actions*:

- *SetAcquireStatus*. This S*et Server Action* sets the *acquireStatus* property to *true/false*. In particular, the client must set this property always *true*. The changing of this value will trigger the *Real Time Action*

sendStatusRequestAction that will update the *status* property with the new value read from the System. The corresponding *Get Server action* has no particular utility but a syntactical consistency function.

- *SetCycleString*. This *Set Server Action* sets the *cycleString* property. The update of this property will trigger the *Real Time Action sendCycleAction,* this action will send the cycle and begin the pre-running operations. The corresponding *Get Server action* has no particular utility but a syntactical consistency function.
- *GetSamples*. This *Get Server Action* returns the updated *samples* property. This property is updated automatically during the cycle running process. No *Set Server Action* is associated, since the property notified cannot be updated by the Operator.
- *GetStatus*. This *Get Server Action* returns the *status* property. This property is updated automatically if a cycle running session is on; or can be updated on demand by setting to *true* the *acquireStatus* property. No *Set Server Action* is associated, since the property notified cannot be updated by the Operator.
- *GetError*. This *Get Server Action* returns the *error* property. This property is updated automatically when an Error occurs during the session. No *Set Server Action* is associated, since the property notified cannot be updated by the Operator.
- *SetHaltServer*. This S*et Server Action* sets the *HaltServer* property to *true/false*. In particular, the client must set this property always *true*. The changing of this value will trigger the *Real Time Action sendHaltCommand* that will send a "*close*" message to the System. The corresponding *Get Server action* has no particular utility but a syntactical consistency function.

References

[27] https://www-acc.gsi.de/wiki/pub/FESA/FESA29210/GettingstartedwithFESA.pdf.

[28] https://www-acc.gsi.de/wiki/pub/FESA/FESA29210/FesaEssentialsBundle.pdf.

[29] P. Arpaia, L. Bottura, G. Montenero, S. Le Naour, "Performance improvement of a measurement station for superconducting cable test", *Review of Scientific Instruments*, vol. 83, 095111, 2012.

[30] Pasquale Arpaia, Giuseppe Montenero – "*Software Requirements: Control System for Superconducting Transformer*". CERN – TE/MSC. 2011.17.08.

[31] http://pocoproject.org/.

Chapter 6

Numerical results

6.1 Preliminary study

To underline the non-linearity behaviors showed by the system in the first working period, a study on the system dynamic under different working conditions was carried out. In order to spot possible dependencies, a set of measurement previously taken was analyzed and the main parameters were highlighted.

6.1.1 Measurements plan

The set of measurements considered, taken with the last transformer settings on December 2010, cover different ramp rates and current on the transformer primary. The measurement plan is shown in Table 6.1 and Table 6.2.

Ramp Rate (V/s)	V plateau (V)	t plateau (s)	V stop (V)	Ramp Rate (-V/s)	I Max (A)
0.002	0.05	10	0	0.002	5
0.002	0.05	50	0	0.002	5
0.002	0.05	100	0	0.002	5
0.002	0.1	10	0	0.002	10
0.002	0.1	50	0	0.002	10
0.002	0.1	100	0	0.002	10
0.002	0.2	10	0	0.002	20
0.002	0.2	50	0	0.002	20
0.002	0.2	100	0	0.002	20
0.002	0.25	10	0	0.002	25
0.002	0.25	50	0	0.002	25
0.002	0.25	100	0	0.002	25

Table 6.1: First set of measurement with variable time plateau and maximum current.

This first set of measurements refers to different maximum primary current levels; namely 5, 10, 20 and 25 Ampere. Each current level was kept for three different time plateau of 10, 50 and 100 seconds. In this set of measurements, the ramp rate has been kept constant to 0.002 V/s.

Ramp Rate (V/s)	V plateau (V)	t plateau (s)	V stop (V)	Ramp Rate (-V/s)	I Max (A)
0.002	0.2	100	0	0.002	20
0.004	0.2	100	0	0.004	20
0.006	0.2	100	0	0.006	20
0.008	0.2	100	0	0.008	20
0.002	0.25	100	0	0.002	25
0.004	0.25	100	0	0.004	25
0.006	0.25	100	0	0.006	25
0.008	0.25	100	0	0.008	25
0.002	0.3	100	0	0.002	30
0.004	0.3	100	0	0.004	30
0.006	0.3	100	0	0.006	30
0.008	0.3	100	0	0.008	30

Table 6.2: First set of measurement with variable ramp rate and maximum current.

The second set of measurements sees a variation of the ramp rate for every maximum primary current level (I Max), which are 20, 25 and 30 Ampere.

6.1.2 Data analysis

The analysis, carried out using Matlab, follows the idea of comparing the measured current on the secondary with the output of an ideal model of the system, given the same real input current of the primary. The Simulink model (figure 6.1) compares the output (IsID) coming from the transformer ideal model (M1), with the measured current on the secondary (Is), kept the same input current on the transformer primary (Ip).

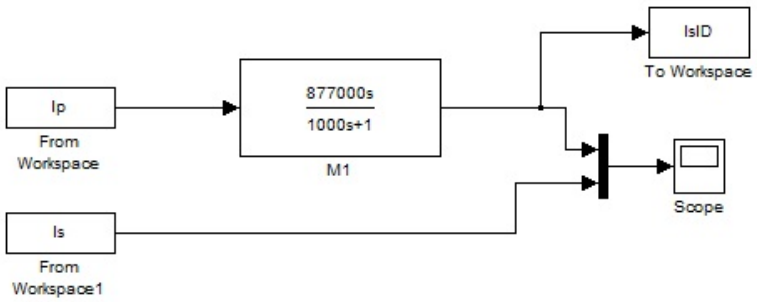

Figure 6.1: Simulink model for the secondary output currents comparison.

For every analyzed data set, several parameters have been computed: Means Squared Error (MSE), Root Mean Squared Error (RMSE), standard deviation and

the one-complement of the Goodness of fit between the two curves. The RMSE has been chosen as parameter of interest to evaluate the non-linearity of the system. The equation used for computing the RMSE is shown in (6.1),

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{k=0}^{n}(IsID(k) - Is(k))^2} \tag{6.1}$$

where n is the number of samples.

As function of the ramp rate and the I Max, RMSE clearly shows the difference in the real system behavior from the ideal one. Hence, the data collected are aggregated showing the variation of the RMSE as function of the ramp rate and I Max.

6.1.3 Numerical results

In order to analyze the data from the two measurements keeping a homogeneous set of samples, all the data from the second set together with some data from the first set was taken. In particular, since the second set of measurement was performed by keeping the time plateau at 100 seconds, from the first set only the samples with the same time plateau were taken. The following Table 6.3 and Table 6.4 show the numerical results taken from the two sets.

Ramp Rate (V/s)	t plateau (s)	Max I (A)	RMSE
0.002	100	5	207.5155
0.002	100	10	205.4294

Table 6.3: Data taken from the first set of measurements.

Ramp Rate (V/s)	t plateau (s)	Max I (A)	RMSE
0.002	100	20	140.2902
0.004	100	20	156.8188
0.006	100	20	168.9354
0.008	100	20	174.1776
0.002	100	25	183.0137
0.004	100	25	170.312
0.006	100	25	175.1858
0.008	100	25	175.8518
0.002	100	30	186.4659
0.004	100	30	211.1258
0.006	100	30	204.5264
0.008	100	30	215.0495

Table 6.4: Data taken from the second set of measurements.

The analyzed data are then presented as a surface graph (Fig. 6.2).

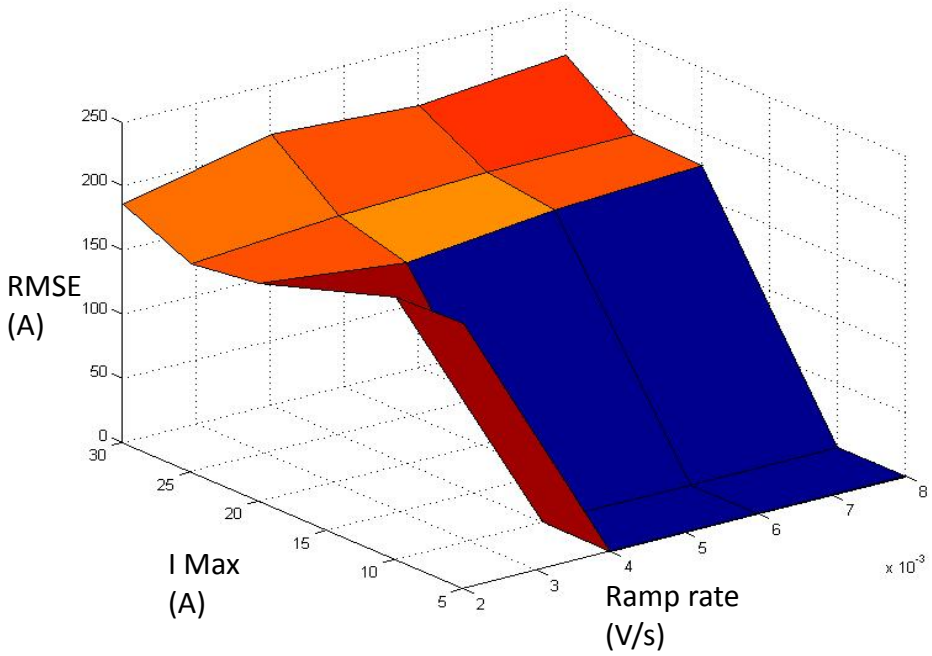

Figure 6.2: Surface of the RMSE as function of the I Max and Ramp rate.

The surface shows the trend of the RMSE as function of the different levels of Ramp rate and I Max on the primary. The zero point, the blue part in the surface, corresponds to missing values, indeed for that couple of I Max and Ramp rate no measure were taken. Even with this lack of information, it is possible to observe a

non-negligible variation of the RMSE. From another perspective the dependency both from the I Max and the Ramp rate it is more appreciable (Fig. 6.3).

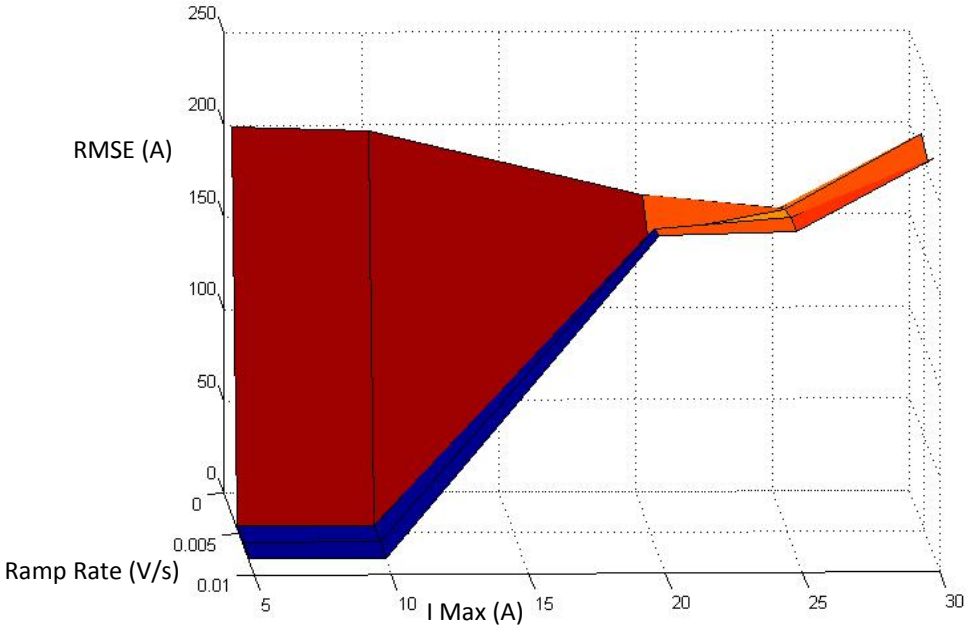

Figure 6.3: Surface of the RMSE on the I Max side

The above perspective shows the RMSE changing according to the different values of I Max. This observation gives the possibility to divide the working space of the System in operating points with similar RMSE. In the design of the Fuzzy Gain scheduling control strategy, those points can identify the fuzzy sets for the scheduling variable, I Max.

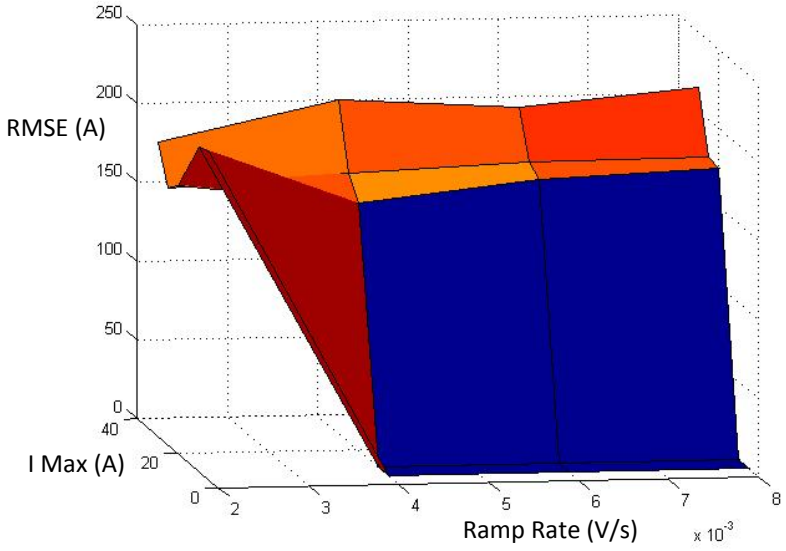

Figure 6.4: Surface of the RMSE, Ramp rate perspective.

Even with the lack of data for some Ramp rates it is possible to evaluate the RMSE variations as function of the Ramp rate also (Fig. 6.4). From the analysis of the presented data it is possible to assert that for evaluating the performance of the proposed system the RMSE is a good indicator.

6.2 System identification

The task of detailing the input-output system dynamic under all possible operation conditions lead to an exhaustive measurement campaign. The exploration of the system behaviour touched most of the domain I_{sec} values through five different ramp rates, namely 50, 100, 300, 500, 800 A/s. Are then detailed all the measurement sessions made for each ramp rate, starting from measurements made with 50 A/s ramps (Table 6.5). For each measurement session are provided a measure identifier (*ID*), the ramp rate of the voltage signal input to the power supplier (*Ramp Rate* (V/s)), the maximum plateau tension fed (V_{Ref} *plat* (V)), the time duration of the linear plateau (*t plat* (s)), the corresponding current signal ramp rate obtained on the transformer secondary (RR_{sec} (A/s)) and the maximum secondary current reached (I_{sec} (A)).

ID	Ramp Rate (V/s)	V_{Ref} plateau (V)	t plat (s)	RR_{sec} (A/s)	I_{sec} (A)
1	0.000570125	0.005701254	10	50	500
2	0.000570125	0.011402509	10	50	1000
3	0.000570125	0.017103763	10	50	1500
4	0.000570125	0.022805017	10	50	2000
5	0.000570125	0.028506271	10	50	2500
6	0.000570125	0.034207526	10	50	3000
7	0.000570125	0.03990878	10	50	3500
8	0.000570125	0.045610034	10	50	4000
9	0.000570125	0.051311288	10	50	4500
10	0.000570125	0.057012543	10	50	5000
11	0.000570125	0.062713797	10	50	5500
12	0.000570125	0.068415051	10	50	6000
13	0.000570125	0.074116306	10	50	6500
14	0.000570125	0.07981756	10	50	7000
15	0.000570125	0.085518814	10	50	7500

16	0.000570125	0.091220068	10	50	8000
17	0.000570125	0.096921323	10	50	8500
18	0.000570125	0.102622577	10	50	9000
19	0.000570125	0.108323831	10	50	9500
20	0.000570125	0.114025086	10	50	10000
21	0.000570125	0.11972634	10	50	10500
22	0.000570125	0.125427594	10	50	11000
23	0.000570125	0.131128848	10	50	11500
24	0.000570125	0.136830103	10	50	12000
25	0.000570125	0.142531357	10	50	12500
26	0.000570125	0.148232611	10	50	13000
27	0.000570125	0.153933865	10	50	13500
28	0.000570125	0.15963512	10	50	14000
29	0.000570125	0.165336374	10	50	14500
30	0.000570125	0.171037628	10	50	15000
31	0.000570125	0.176738883	10	50	15500
32	0.000570125	0.182440137	10	50	16000
33	0.000570125	0.188141391	10	50	16500
34	0.000570125	0.193842645	10	50	17000
35	0.000570125	0.1995439	10	50	17500
36	0.000570125	0.205245154	10	50	18000
37	0.000570125	0.210946408	10	50	18500
38	0.000570125	0.216647662	10	50	19000
39	0.000570125	0.222348917	10	50	19500
40	0.000570125	0.228050171	10	50	20000

Table 6.5: system measurements taken with 50 A/s reference signal ramp rate.

Higher current levels were reached through the test run with 100 and 300 A/s ramp rate (Table 6.6, 6.7), where an I_{sec} of 24 kA is achieved. It has to be pointed out that the ramp rate of the voltage input signal fed into the power supplier is calculated in order to allow the current ramp rate on the transformer secondary to be the desired one.

ID	Ramp Rate (V/s)	V_{Ref} plateau (V)	t plat (s)	RR_{sec} (A/s)	I_{sec} (A)
1	0.001140251	0.005701254	10	100	500
2	0.001140251	0.017103763	10	100	1500
3	0.001140251	0.028506271	10	100	2500
4	0.001140251	0.03990878	10	100	3500

5	0.001140251	0.051311288	10	100	4500
6	0.001140251	0.062713797	10	100	5500
7	0.001140251	0.074116306	10	100	6500
8	0.001140251	0.085518814	10	100	7500
9	0.001140251	0.096921323	10	100	8500
10	0.001140251	0.108323831	10	100	9500
11	0.001140251	0.11972634	10	100	10500
12	0.001140251	0.131128848	10	100	11500
13	0.001140251	0.142531357	10	100	12500
14	0.001140251	0.153933865	10	100	13500
15	0.001140251	0.165336374	10	100	14500
16	0.001140251	0.176738883	10	100	15500
17	0.001140251	0.188141391	10	100	16500
18	0.001140251	0.1995439	10	100	17500
19	0.001140251	0.210946408	10	100	18500
20	0.001140251	0.222348917	10	100	19500
21	0.001140251	0.233751425	10	100	20500
22	0.001140251	0.245153934	10	100	21500
23	0.001140251	0.256556442	10	100	22500
24	0.001140251	0.267958951	10	100	23500
25	0.001140251	0.27936146	10	100	24500

Table 6.6: system measurements taken with 100 A/s reference signal ramp rate.

The current difference between each measure run was increased after the first run, and the step width is enlarged at each ramp rate measure session.

ID	Ramp Rate (V/s)	V_{Ref} plateau (V)	t plat (s)	RR_{sec} (A/s)	I_{sec} (A)
1	0.003420753	0.005701254	10	300	500
2	0.003420753	0.017103763	10	300	1500
3	0.003420753	0.028506271	10	300	2500
4	0.003420753	0.03990878	10	300	3500
5	0.003420753	0.051311288	10	300	4500
6	0.003420753	0.062713797	10	300	5500
7	0.003420753	0.074116306	10	300	6500
8	0.003420753	0.085518814	10	300	7500
9	0.003420753	0.096921323	10	300	8500
10	0.003420753	0.108323831	10	300	9500
11	0.003420753	0.11972634	10	300	10500

12	0.003420753	0.131128848	10	300	11500
13	0.003420753	0.142531357	10	300	12500
14	0.003420753	0.153933865	10	300	13500
15	0.003420753	0.165336374	10	300	14500
16	0.003420753	0.176738883	10	300	15500
17	0.003420753	0.188141391	10	300	16500
18	0.003420753	0.1995439	10	300	17500
19	0.003420753	0.210946408	10	300	18500
20	0.003420753	0.222348917	10	300	19500
21	0.003420753	0.233751425	10	300	20500
22	0.003420753	0.245153934	10	300	21500
23	0.003420753	0.256556442	10	300	22500
24	0.003420753	0.267958951	10	300	23500
25	0.003420753	0.27936146	10	300	24500

Table 6.7: system measurements taken with 300 A/s reference signal ramp rate.

The system characterization is completed with the highest ramp rate sessions at 500 and 800 A/s (Table 6.8, 6.9), during these sessions the maximum I_{Sec} current is driven, namely 29 kA at 500 A/s and 30.5 KA at 800 A/s.

ID	Ramp Rate (V/s)	V_{Ref} plateau (V)	t plat (s)	RR_{sec} (A/s)	I_{sec} (A)
1	0.005701254	0.005701254	10	500	500
2	0.005701254	0.022805017	10	500	2000
3	0.005701254	0.03990878	10	500	3500
4	0.005701254	0.057012543	10	500	5000
5	0.005701254	0.074116306	10	500	6500
6	0.005701254	0.091220068	10	500	8000
7	0.005701254	0.108323831	10	500	9500
8	0.005701254	0.125427594	10	500	11000
9	0.005701254	0.142531357	10	500	12500
10	0.005701254	0.15963512	10	500	14000
11	0.005701254	0.176738883	10	500	15500
12	0.005701254	0.193842645	10	500	17000
13	0.005701254	0.210946408	10	500	18500
14	0.005701254	0.228050171	10	500	20000
15	0.005701254	0.245153934	10	500	21500
16	0.005701254	0.262257697	10	500	23000
17	0.005701254	0.27936146	10	500	24500

18	0.005701254	0.296465222	10	500	26000
19	0.005701254	0.313568985	10	500	27500
20	0.005701254	0.330672748	10	500	29000

Table 6.8: system measurements taken with 500 A/s reference signal ramp rate.

In the 800 A/s ramp rate measurement session the step between two consecutive measures reached the 2000 A.

ID	Ramp Rate (V/s)	V_{Ref} plateau (V)	t plat (s)	RR_{sec} (A/s)	I_{sec} (A)
1	0.009122007	0.005701254	10	800	500
2	0.009122007	0.028506271	10	800	2500
3	0.009122007	0.051311288	10	800	4500
4	0.009122007	0.074116306	10	800	6500
5	0.009122007	0.096921323	10	800	8500
6	0.009122007	0.11972634	10	800	10500
7	0.009122007	0.142531357	10	800	12500
8	0.009122007	0.165336374	10	800	14500
9	0.009122007	0.188141391	10	800	16500
10	0.009122007	0.210946408	10	800	18500
11	0.009122007	0.233751425	10	800	20500
12	0.009122007	0.256556442	10	800	22500
13	0.009122007	0.27936146	10	800	24500
14	0.009122007	0.302166477	10	800	26500
15	0.009122007	0.324971494	10	800	28500
16	0.009122007	0.347776511	10	800	30500

Table 6.9: system measurements taken with 800 A/s reference signal ramp rate.

In order to have clean data available to compute the system a model, the first sixty seconds of zero current for each measure were used to calculate the measurement offset and correct the data accordingly.

6.3 Model definition

For all the twelve system working regions identified the transfer function that better approximates the system dynamics in that interval was computed with the available data. The resulting system (Fig. 6.5) is a composition of the subdomains identifications.

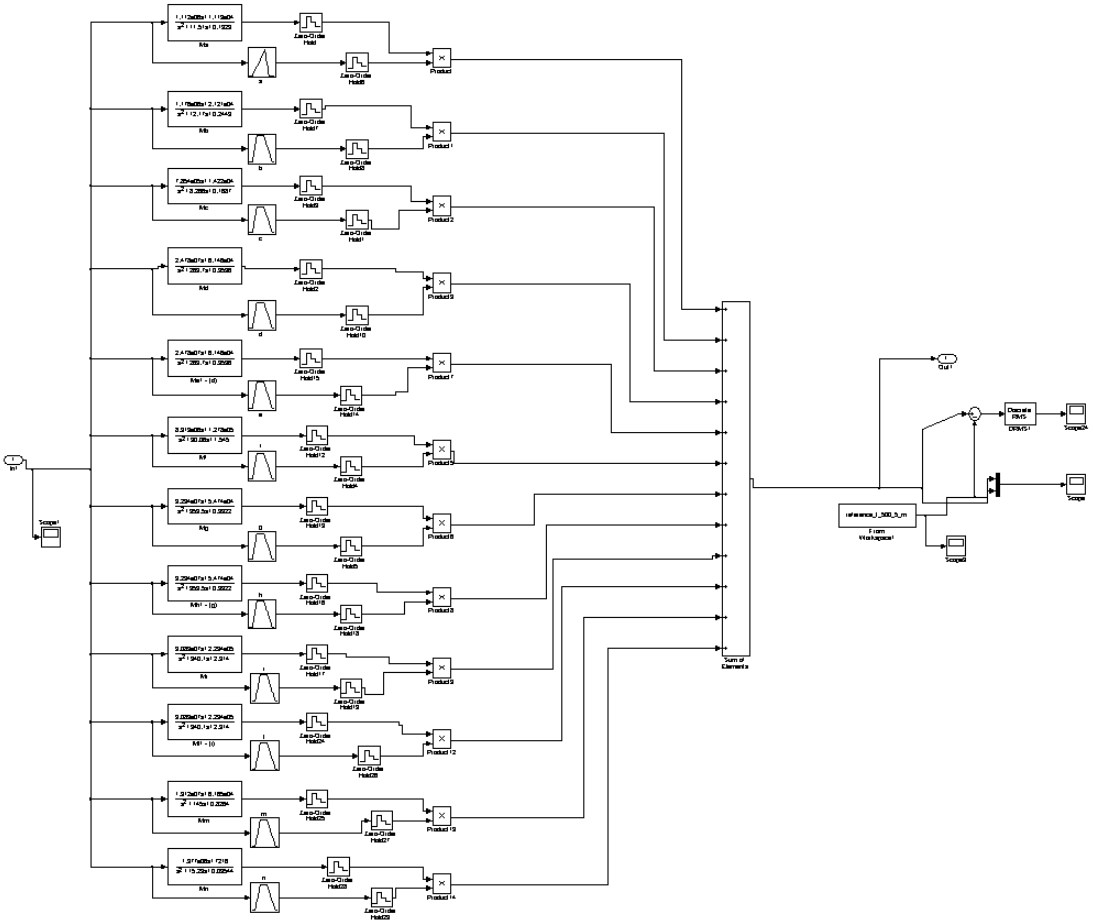

Figure 6.5: The Simulink block scheme of the inferred model.

The transfer functions were calculated using the Matlab toolbox for system identification providing as input the measures data related to the working region to identify. The model validity was assessed through the fitting percentage (6.2) computation with respect to the data collected in the measures campaign.

$$Fit\ \% = \left(1 - \frac{\sum_{i=1}^{n}\left(I_{meas,i} - I_{mod,i}\right)^2}{(n-1)\cdot s^2}\right)\cdot 100 \qquad (6.2)$$

Where the I_{meas} is the measured secondary current, I_{mod} is the output predicted by the system model given the same input, n is the number of the measurement sample and s is the standard deviation of the measured data. The domain fuzzyfication, through trapezoidal membership function, can be described by each trapezoidal function four parameters: A, B, C and D (Fig. 6.6).

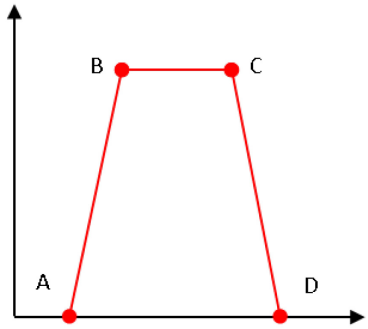

Figure 6.6: The trapezoidal membership function and its parameters.

The parameters of the identified I_{sec} subdomains membership function are then listed (Table 6.10a, 6.10b).

MF / params	*a*	*b*	*c*	*d*	*e*	*f*
A	0	500	2500	5500	8000	11500
B	0	1500	3500	6500	9000	12500
C	500	2500	5500	8000	11500	14500
D	1500	3500	6500	9000	12500	15500

Table 6.10a: parameters of the membership function of the subdomain from *a* to *f*.

MF / params	*g*	*h*	*i*	*l*	*m*	*n*
A	14500	17500	20500	23000	25000	27000
B	15500	18500	21500	24000	26000	28000
C	17500	20500	23000	25000	27000	36000
D	18500	21500	24000	26000	28000	36000

Table 6.10b: parameters of the membership function of the subdomain from *g* to *n*.

The model performance are then summarized (Table 6.11a, 6.11b),

Region	*a*	*b*	*c*	*d*	*e*	*f*
Fit %	98.48	97.37	98.36	97.93	98.12	98.84

Table 6.11a: fitness of the system model in subdomains *a* to *f*.

Region	*g*	*h*	*i*	*l*	*m*	*n*
Fit %	98.27	98.69	99.41	99.44	99.66	99.67

Table 6.11b: fitness of the system model in subdomains *g* to *n*.

for each operating region the average fitting value among all measurement data, for all the ramp rates, afferent to the same region is indicated.

6.4 Controller simulation

From the region-wise modelled system, for each subdomain a digital PI controller was tuned following the same structure of the one already used for the transformer control (Fig. 6.7). The conversion from continuous to discrete domain was carried out through the zero-order-hold (ZOH) model with a sampling time of 0.05 s. After the Simulink model transfer functions a ZOH block was placed in order to better approximate the behaviour of the real system.

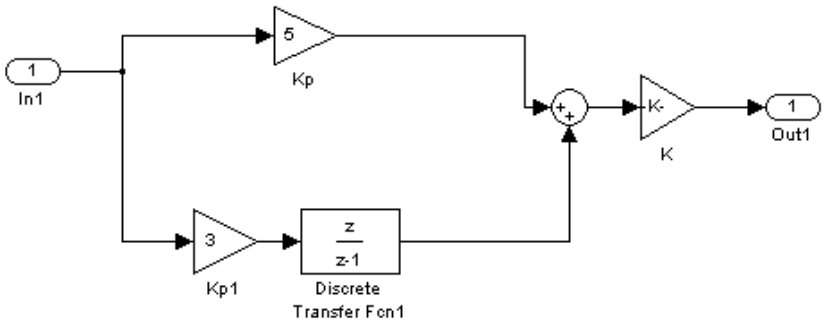

Figure 6.7: The Simulink block scheme of the available digital PI controller.

The structure of this PI controller provides in addition to the two usual parameters for this type of controllers a gain that multiplies the sum of the proportional and integral contributions. This gain which has a value of 1.36e-6 and was kept in the new configuration in order to allow a comparison, especially

in terms of magnitude order, with the currently used parameters. The parameters calculated for each of the operating regions identified are summarized (Table 6.12a, 6.12b)

MF / PI params	*a*	*b*	*c*	*d*	*e*	*f*
K_P	9.056	9.8172	13.9459	14.0587	14.3259	14.4852
K_I	7.0368	7.2635	7.6542	7.7484	7.8911	7.8968

Table 6.12a: parameters of the identified digital PI controllers.

MF / PI params	*g*	*h*	*i*	*l*	*m*	*n*
K_P	14.3489	14.3482	14.2679	13.9783	13.8768	13.9856
K_I	7.7223	7.7056	7.6248	7.3411	7.1540	7.2587

Table 6.12b: parameters of the identified digital PI controllers.

and included in the PI-FGS control system (Fig. 6.8).

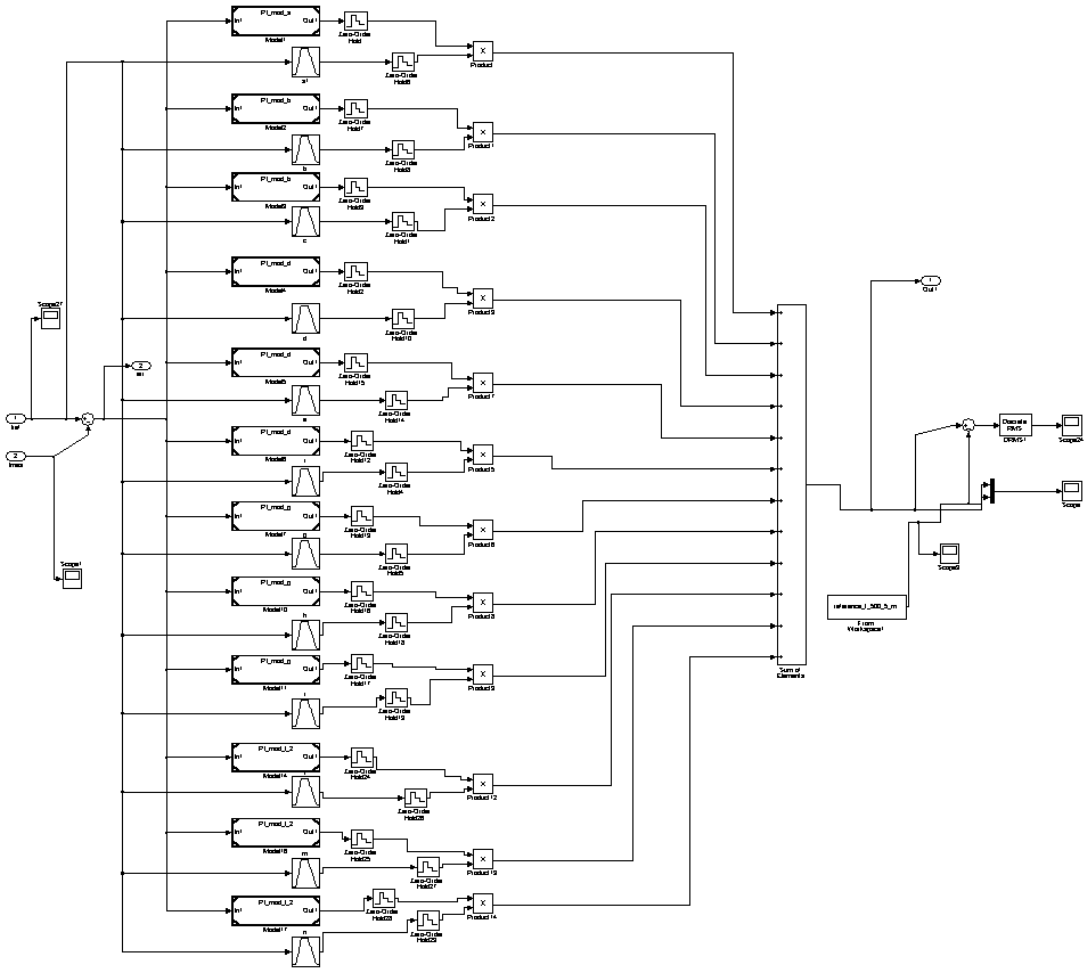

Figure 6.8: The Simulink block scheme of the computed PI-FGS controller.

The controller simulation performance are then presented and compared with those given by the currently available controller, for each current reference chosen the high maximum current plateau response, the absolute error and the RMSE are reported. The profiles simulated are then listed (Table 6.13).

Ramp Rate	*800 A/s*	*500 A/s*	*300 A/s*	*100 A/s*
$I_{sec,max}$	30 kA	25 kA	20 kA	10 kA
$I_{sec,max}$	25 kA	20 kA	15 kA	5 kA

Table 6.13: Simulation plan.

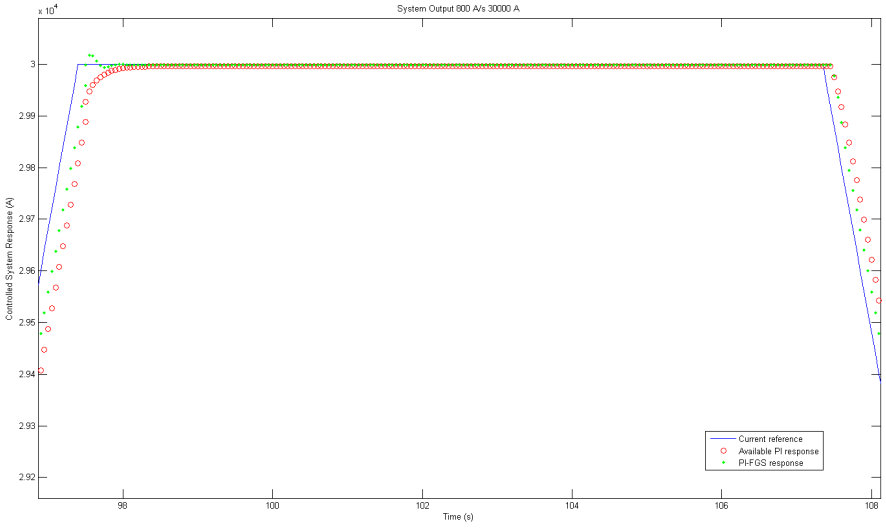

Figure 6.9: Controlled system response (30 kA, 800 A/s): Reference (blue), PI (o), PI-FGS (·).

The most critical conditions are simulated (Fig. 6.9) with a maximum ramp rate reference up to 30 kA. The system responses are quite different even thou they show almost no error on the high plateau. The ramp-plateau transitions are different; the PI control slowly follows the reference changes while the proposed control presents a slight overshoot together with a slightly higher capability to follow the reference ramp. Further information can be gathered from the absolute error plot (Fig 6.10) showing a significant difference between the two errors. The PI-FGS error indeed is almost half on average of the PI one. It is worthy to note

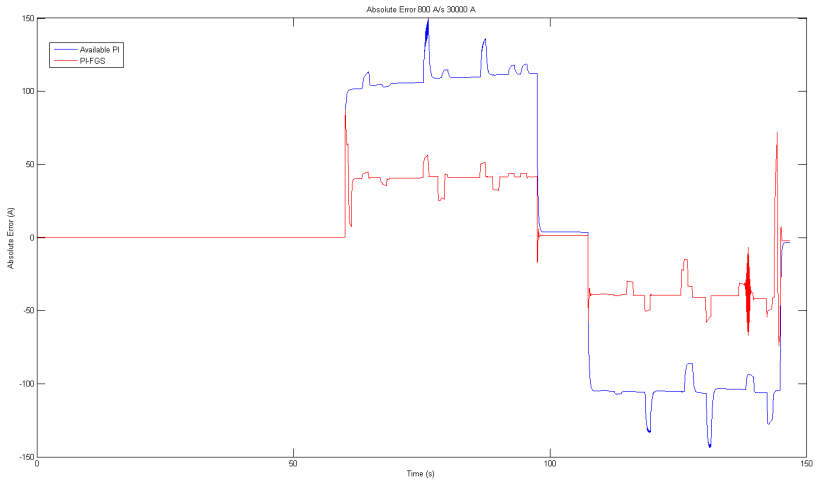

Figure 6.10: Controllers error (30 kA, 800 A/s): PI (blue), PI-FGS (red).

the several undershoot and overshoot most likely deriving from a model subdomain unsmoothed transition. The absolute error trend is then confirmed by the RMSE simulation plot (Fig 6.11) where the difference between the magnitude values of the two errors is more evident.

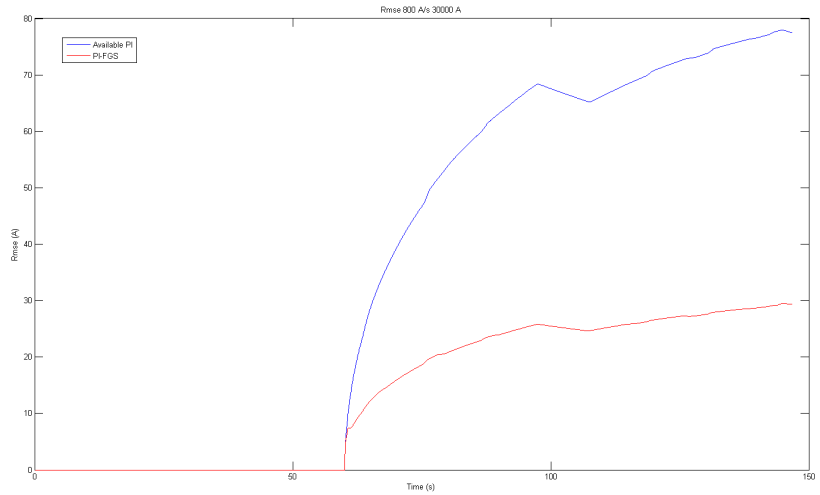

Figure 6.11: Controllers RMSE (30 kA, 800 A/s): PI (blue), PI-FGS (red).

The numerical analysis continues for the 800 A/s simulations by considering the 25 kA maximum current responses (Fig. 6.12). Apart from a more nervous behaviour on the high plateau the analysis given for the 30 kA can be replicated in this case even with a slightly less marked overshoot.

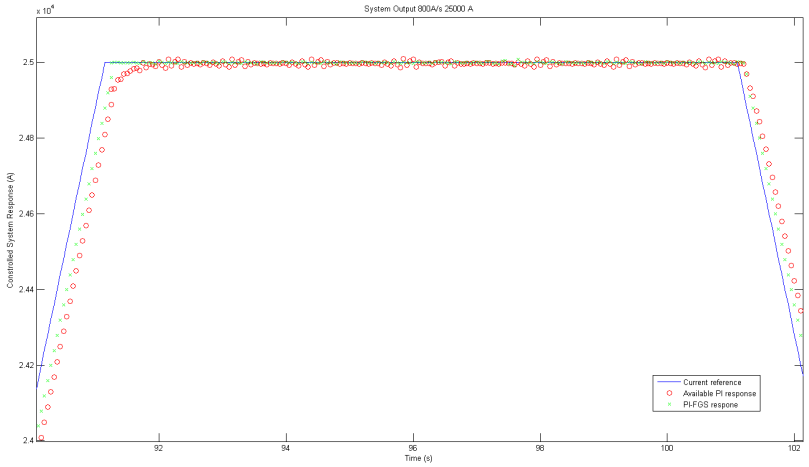

Figure 6.12: Controlled system response (25 kA, 800 A/s): Reference (blue), PI (o), PI-FGS (·).

The absolute error comparison (Fig. 6.13) underlines again the PI-FGS higher capability to follow the reference ramp.

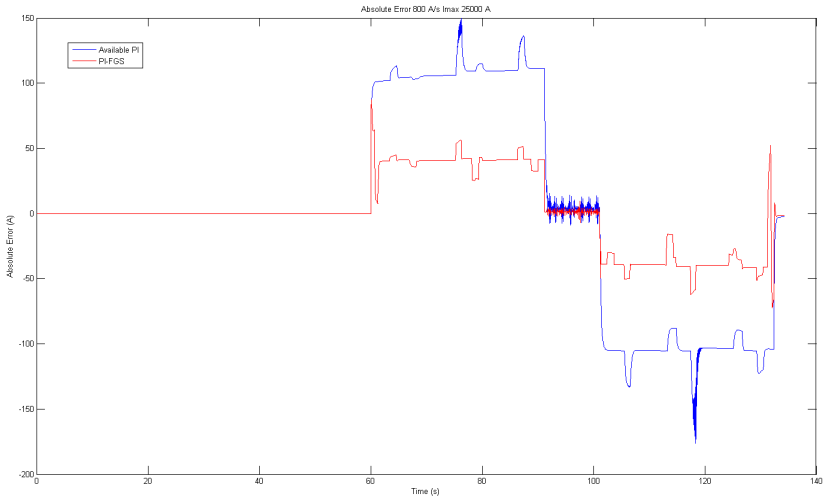

Figure 6.13: Controllers error (25 kA, 800 A/s): PI (blue), PI-FGS (red).

The RMSE plot (Fig. 6.14) confirms the reactivity of the PI-FGS with respect to the PI. The simulation is then focused on the 500 A/s ramp rate references, the first one analysed is the one reaching the 25 kA of maximum current. The response plot (Fig. 6.15) can be assimilated to the analogous measurement at 800 A/s with the same behaviour in the ramp-plateau transitions and the same ripples on the plateau. The effectiveness in following the ramp reference is again observable from the absolute error plot (Fig. 6.16) and the RMSE one (Fig 6.17).

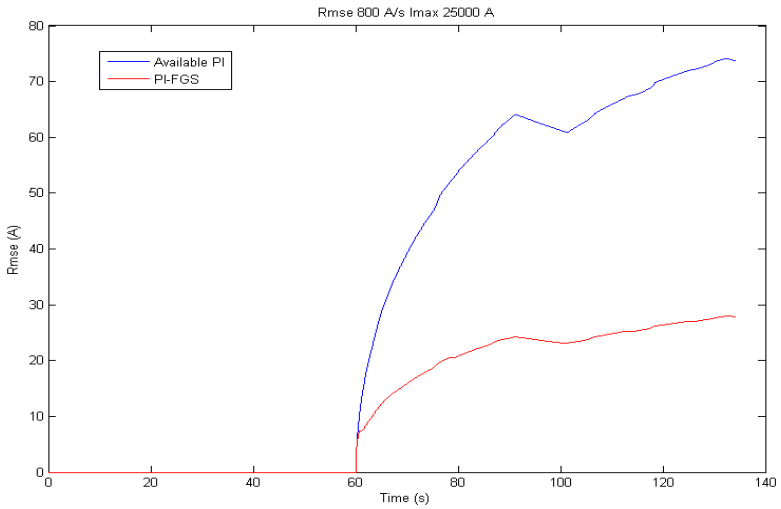

Figure 6.14: Controllers RMSE (25 kA, 800 A/s): PI (blue), PI-FGS (red).

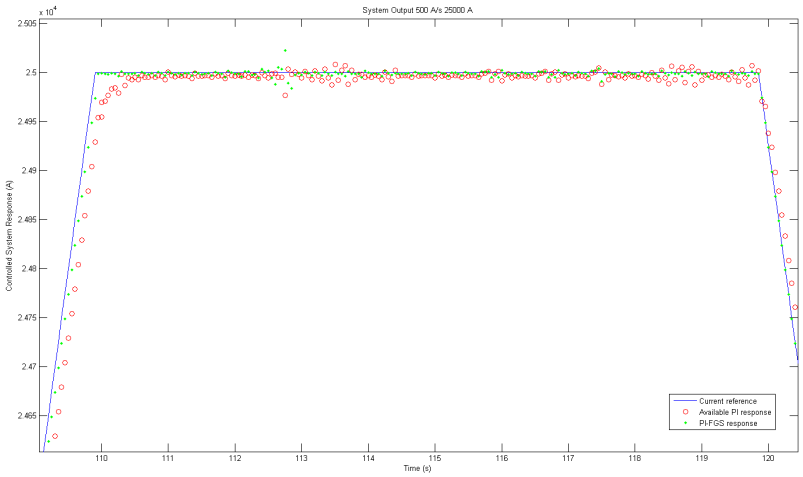

Figure 6.15: Controlled system response (25 kA, 500 A/s): Reference (blue), PI (o), PI-FGS (·).

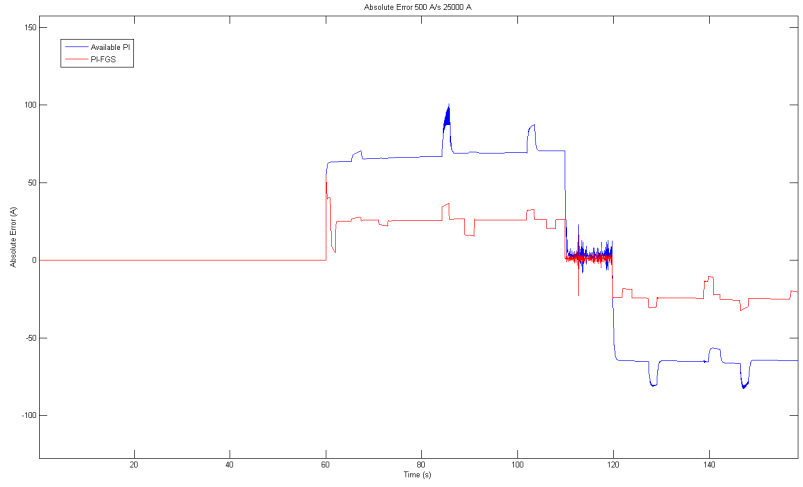

Figure 6.16: Controllers error (25 kA, 500 A/s): PI (blue), PI-FGS (red).

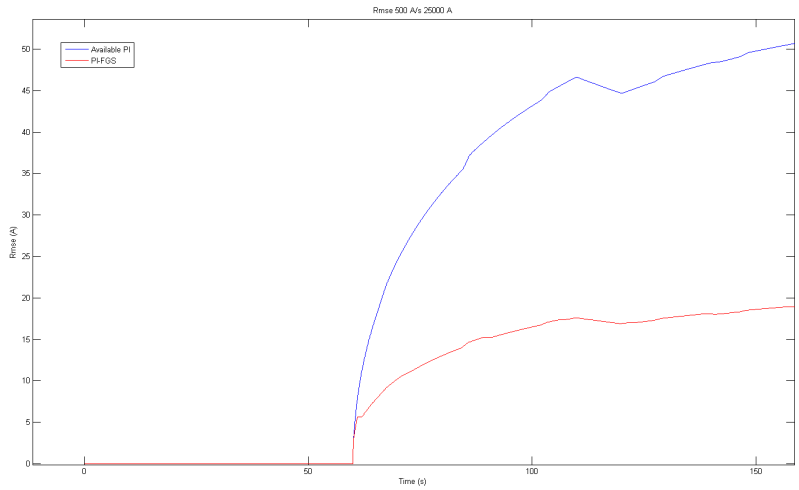

Figure 6.17: Controllers RMSE (25 kA, 500 A/s): PI (blue), PI-FGS (red).

Observing the same behaviour again in 20 kA responses plot (Fig 6.18) lead to the hypothesis that there is and effective capability of the PI-FGS to follow better the reference ramp as well as there is an extreme sensibility to the variation in the system dynamics.

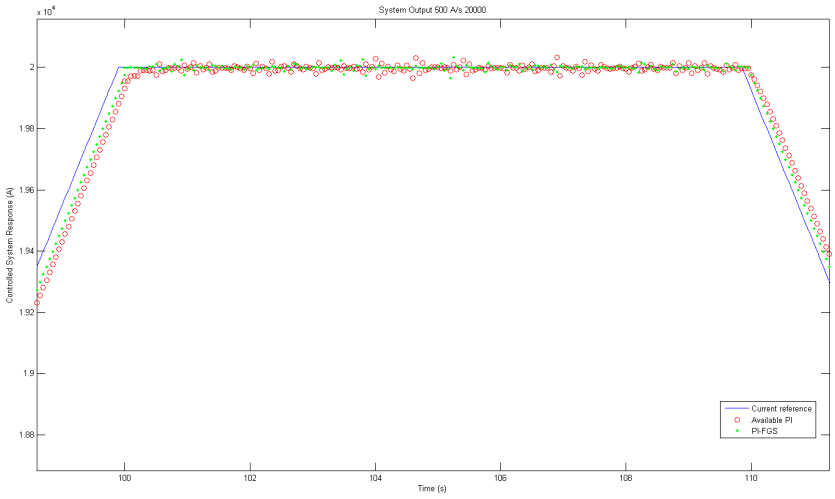

Figure 6.18: Controlled system response (20 kA, 500 A/s): Reference (blue), PI (o), PI-FGS (·).

Another possibility can be represented by the system model dynamics not sufficiently well defined. This possibility can be better evaluated observing the absolute error oscillating trend (Fig. 6.19), while the RMSE plot (6.20) reports the already seen difference between the two responses. The simulation study continues switching to the 300 A/s ramp rate references.

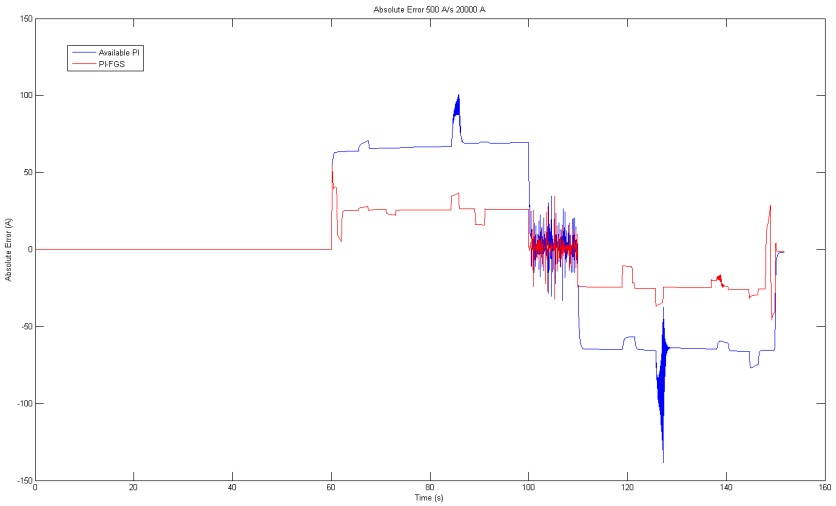

Figure 6.19: Controllers error (20 kA, 500 A/s): PI (blue), PI-FGS (red).

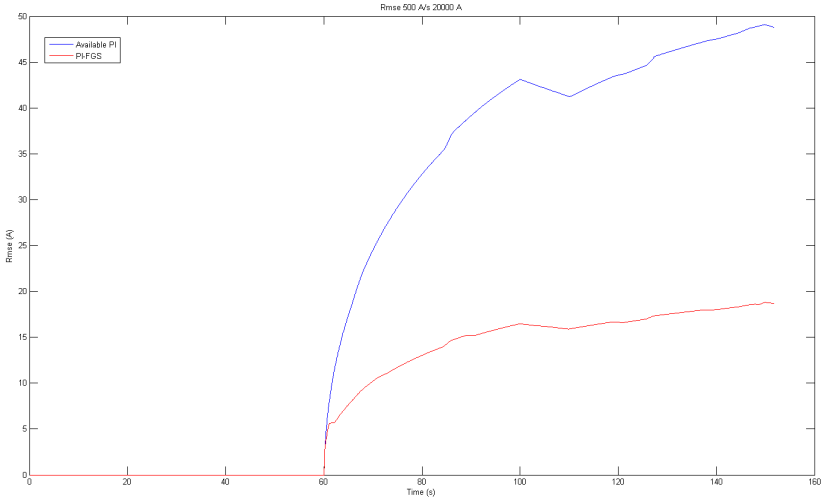

Figure 6.20: Controllers RMSE (20 kA, 500 A/s): PI (blue), PI-FGS (red).

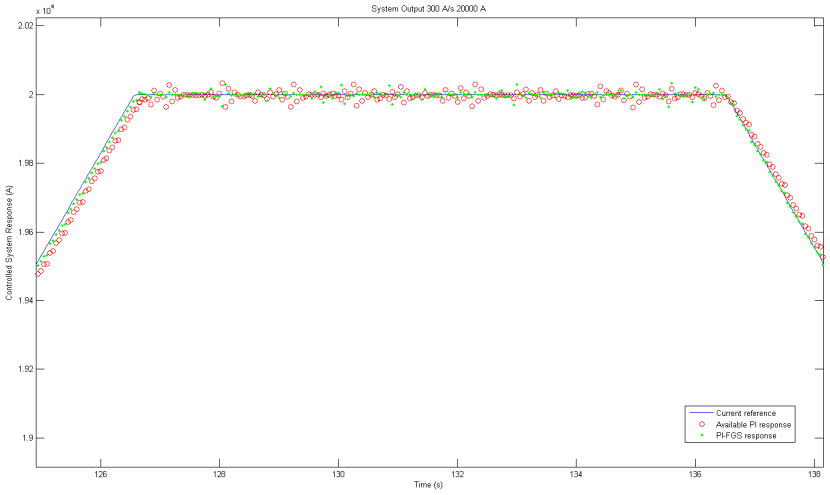

Figure 6.21: Controlled system response (20 kA, 300 A/s): Reference (blue), PI (o), PI-FGS (·).

The higher current reference simulated is at 20 kA, the resulting responses (Fig. 6.21) besides endorsing the assumptions made above shows the PI decreasing difficulty in following less steep ramp.

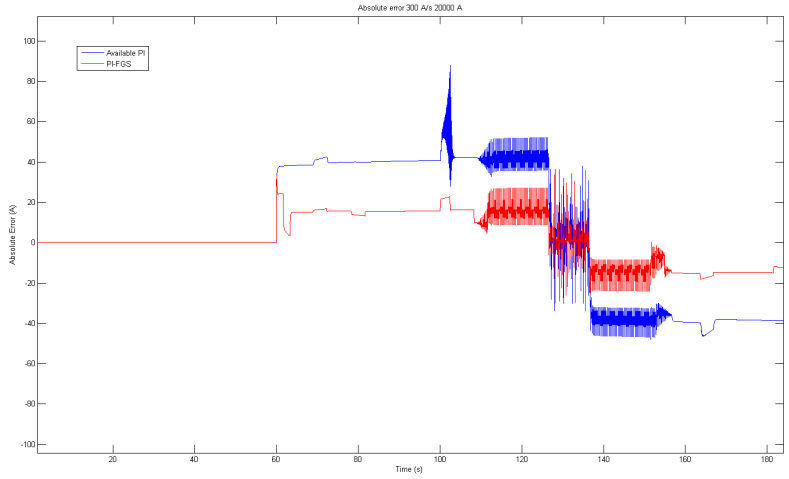

Figure 6.22: Controllers error (20 kA, 300 A/s): PI (blue), PI-FGS (red).

This trend is also confirmed by the absolute error plot (Fig. 6.22) and from the RMSE one (Fig. 6.23), even slightly.

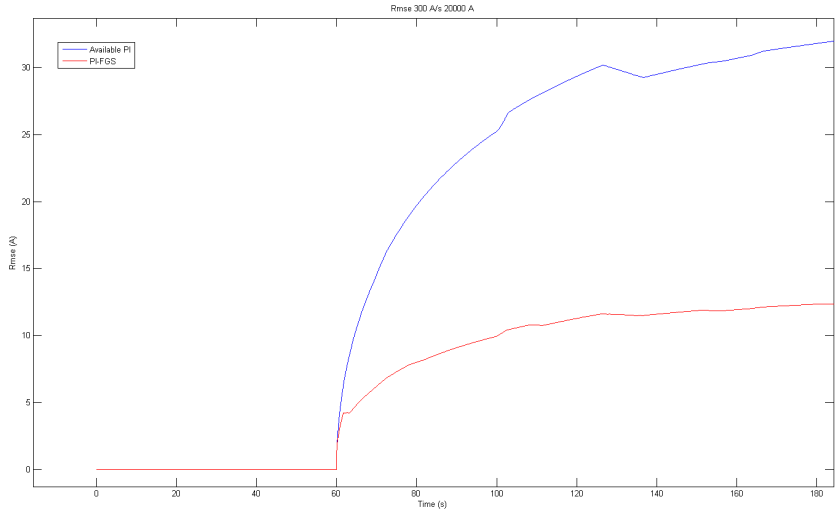

Figure 6.23: Controllers RMSE (20 kA, 300 A/s): PI (blue), PI-FGS (red).

The numerical analysis at 300 A/s ramp rate is completed by checking at the 15 kA simulations. The responses (Fig 6.24) present a known behaviour with the PI-

FGS following better the ramp with a slight overshoot and the PI following slower.

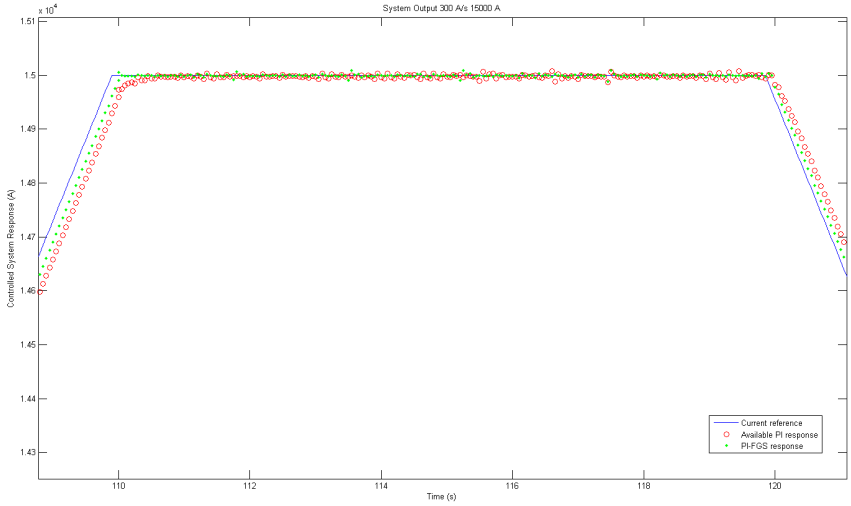

Figure 6.24: Controlled system response (15 kA, 300 A/s): Reference (blue), PI (o), PI-FGS (·).

The decreased error difference between the two controllers is still appreciable in the absolute error plot (Fig. 6.25) even with an unusual overshoot present in the PI error curve.

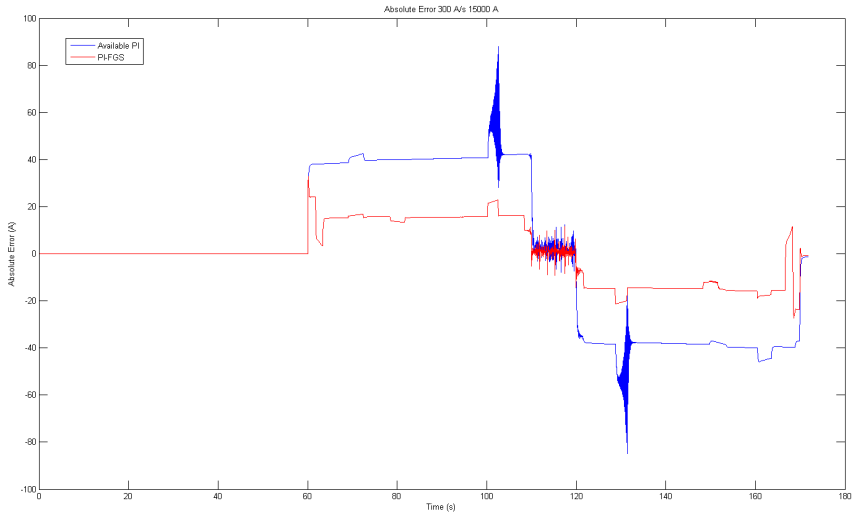

Figure 6.25: Controllers error (15 kA, 300 A/s): PI (blue), PI-FGS (red).

The decreased error difference between the two controllers is still appreciable in the absolute error plot (Fig. 6.26) even with an unusual overshoot present in the PI error curve.

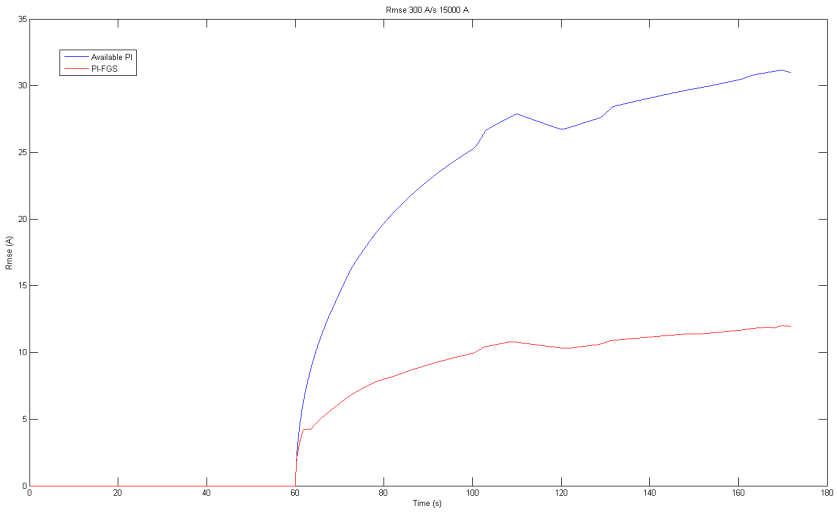

Figure 6.26: Controllers RMSE (15 kA, 300 A/s): PI (blue), PI-FGS (red).

The RMSE plot (Fig. 6.26) reports closer difference between the responses and again it is visible the unusual overshoot present in the PI error curve. The last simulations considered in this numerical analysis are the 100 A/s ramp rate reference ones. In the 10kA plot (Fig. 6.27) the responses are much closer but still keeping the same ramp-plateau transition dynamic even without the plateau ripples shown in the above considered simulations.

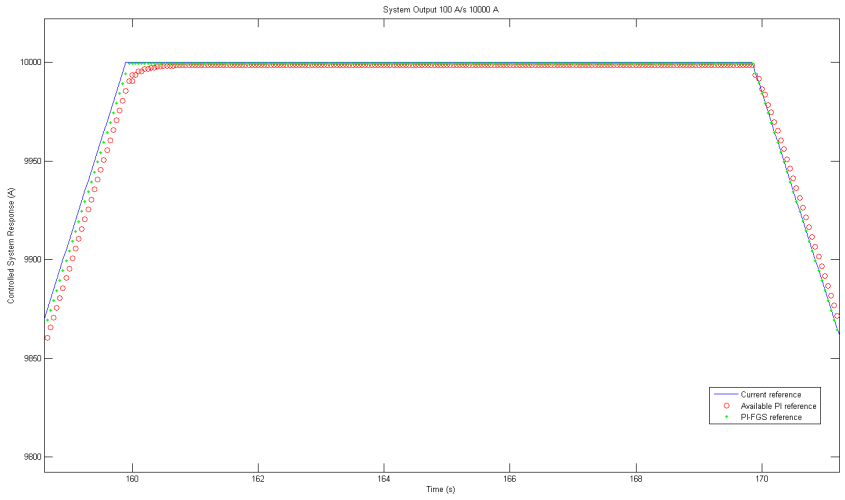

Figure 6.27: Controlled system response (10 kA, 100 A/s): Reference (blue), PI (o), PI-FGS (·).

The absolute errors (Fig. 6.28) curves are much closer even the magnitude distance of the RMSE error characteristics (Fig. 6.29) is significantly reduced.

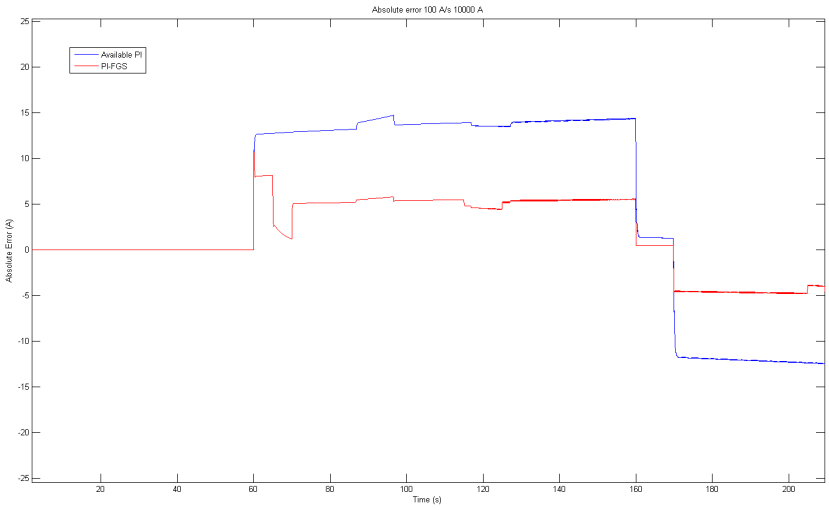

Figure 6.28: Controllers error (10 kA, 100 A/s): PI (blue), PI-FGS (red).

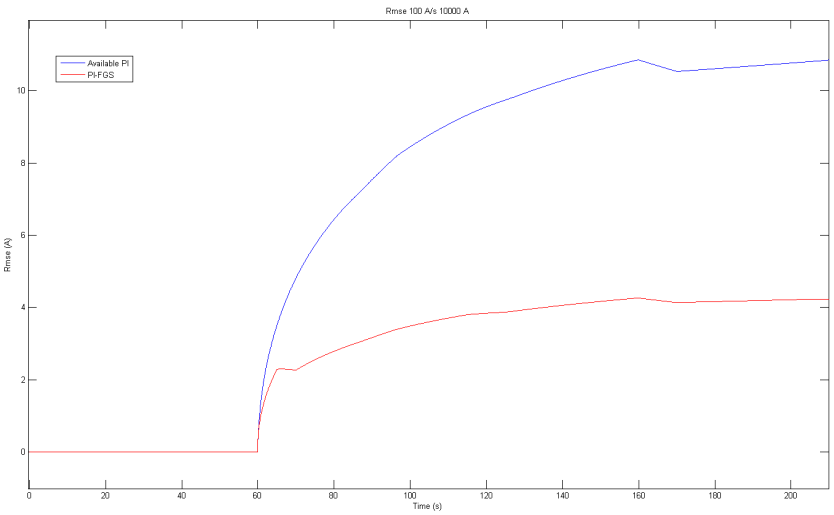

Figure 6.29: Controllers RMSE (10 kA, 100 A/s): PI (blue), PI-FGS (red).

The last simulation considered refers to the 5 kA reference at 100 A/s. Since the ramp rate is gentler with respect to the previous one considered it is not surprising to observe the two responses (Fig. 6.30) following the reference with no appreciable error on the plateau, with the usual differences on the ramps phases. Even the values shown by the absolute error plot (Fig. 6.31) and the RMSE one (Fig. 6.32) are much closer to each other.

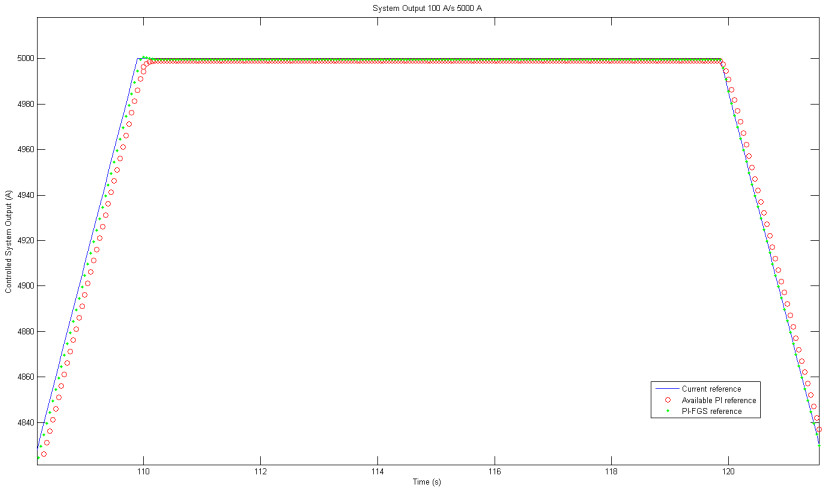

Figure 6.30: Controlled system response (5 kA, 100 A/s): Reference (blue), PI (o), PI-FGS (·).

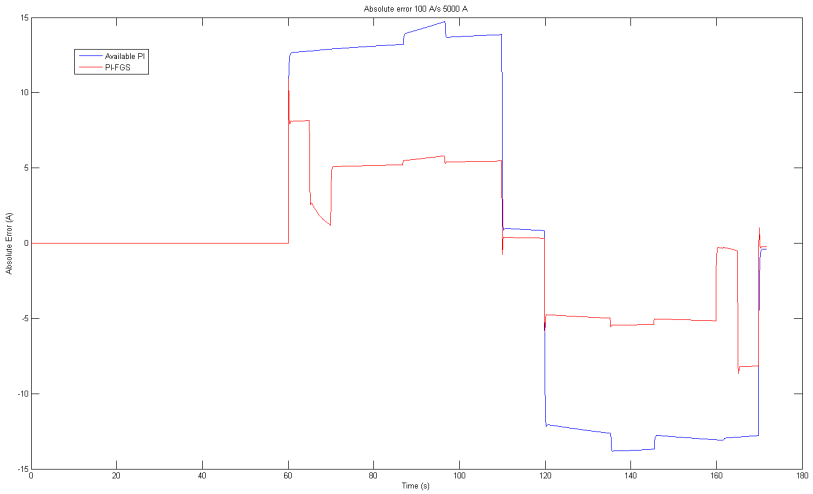

Figure 6.31: Controllers error (5 kA, 100 A/s): PI (blue), PI-FGS (red).

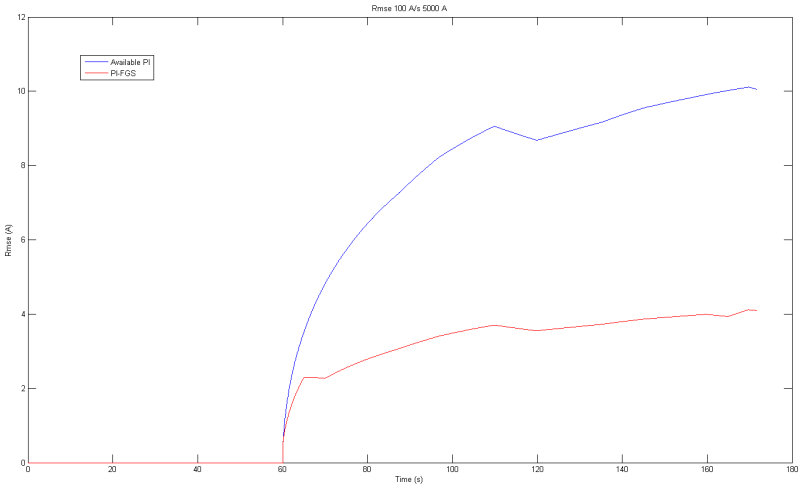

Figure 6.32: Controllers RMSE (5 kA, 100 A/s): PI (blue), PI-FGS (red).

The simulation results shows a common trend indicating a significant error decrease, both absolute and RMSE, when the system is controlled by the computed PI-FGS strategy. In particular the major improvement is registered in the reference ramp phase; the PI-FGS shows a better capability to follow the ramp part of the signal both in the rising phase and in the descending phase. In response to such error decrease for all the considered conditions, is to underline a greater sensitivity of the PI-FGS controller in the transitions between the measure different phases. This sensitivity turns into more marked oscillations and a slight overshoot on the maximum current plateau to be monitored in the testing phase.

Chapter 7

Experimental results

7.1 Primary tuning

The performance analysis was carried out by two steps, first a tuning phase is executed, in order to understand the preliminary performance and eventually correct the parameters. The analysis is completed by validating the proposed system on the test chain as a whole, i.e., measurement and control loop, by measuring of the voltage-current curve of a reference superconducting cable. At this aim, a Rutherford NbTi cable with 36 strands of 0.825 mm diameter, used for the production of the outer layer of the LHC dipoles, the so called LHC cable of type 2 [1], was employed. The tuning tests performed followed the usual current reference function (figure 7.1).

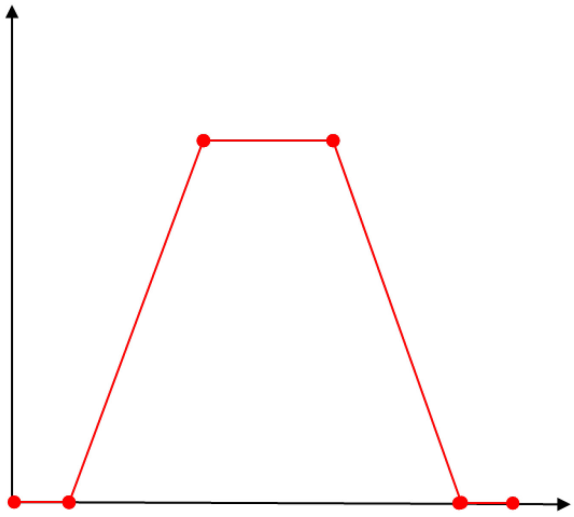

Figure 7.2: The current reference function used for the primary tests.

The low plateau phase last two seconds each while the high plateau is kept for ten seconds. The two measurements were run in this phase, the first is characterized by a 100 A/s ramp rate and a 1000 A of maximum current (figure 7.2). The second one reached the 1000 A of maximum current at 800 A/s of ramp rate (figure 7.3).

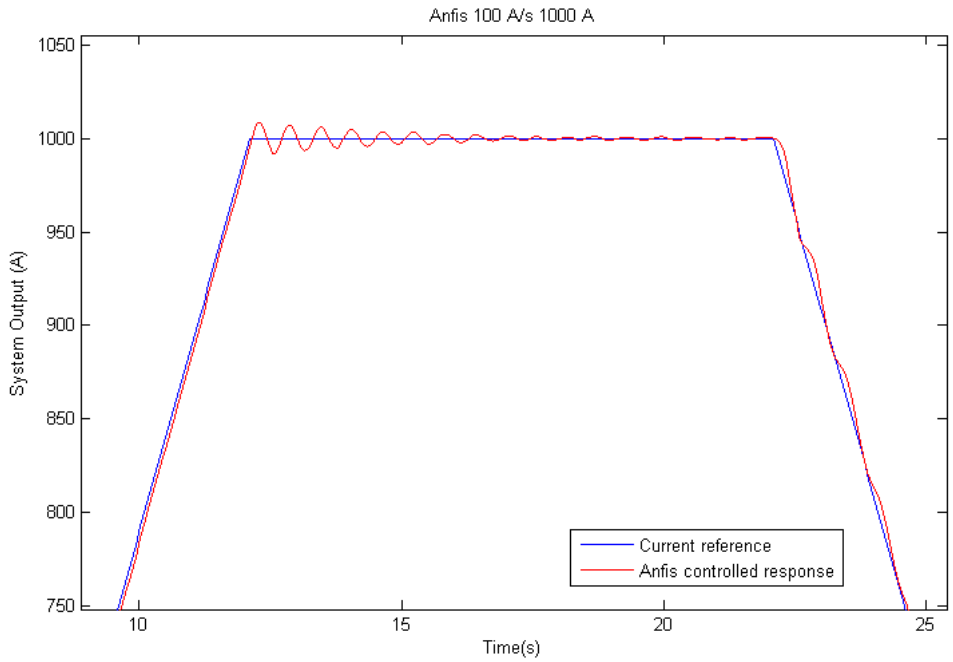

Figure 7.2: The system response for a 100 A/s ramp rate reference up to 1000 A.

Even thou the controlled system shows decent reference tracking in the ramp phase, the overshooting phenomenon evident approaching the high plateau it is not a desired phenomenon to occur at higher current.

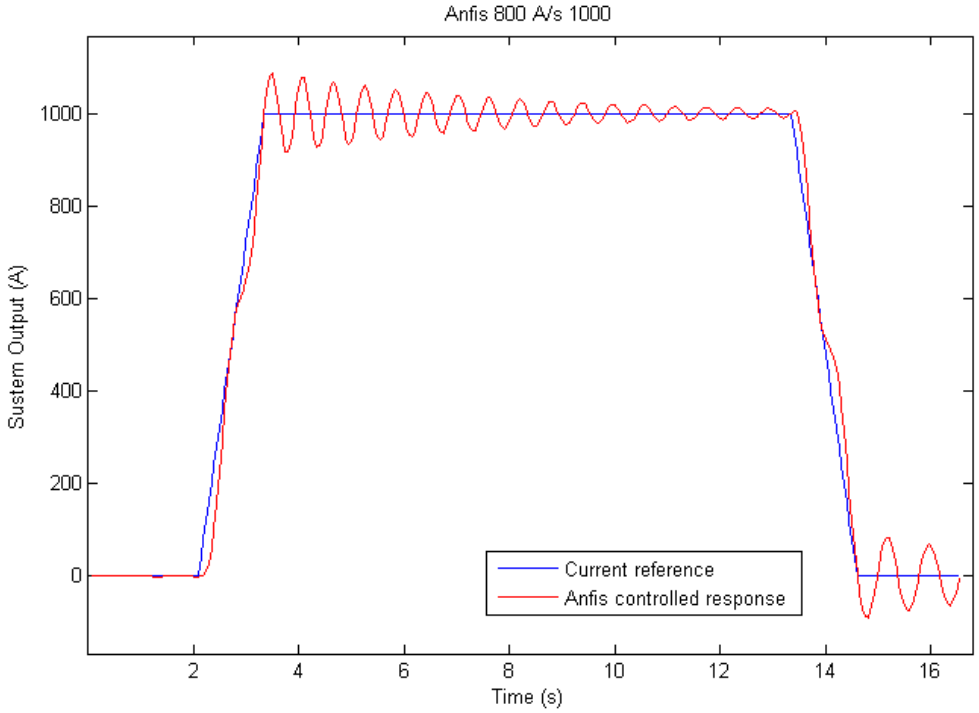

Figure 7.3: The system response for a 800 A/s ramp rate reference up to 1000 A.

To have a further point of view on the high plateau response the second test is considered (figure 7.3). The overshooting, in the response acquired in the second test, is much stronger, most likely for the steeper ramp. The simulation trend of extreme sensibility is confirmed by these preliminary experimental results. Together with further parameters tuning another solution was found to overcome the issue given by the excessive sensibility of the system. A transition time of 0.5 s between a ramp and a plateau (or conversely between a plateau and a ramp) was imposed. This expedient is negligible in a regular test, but it is very useful to help the controller to avoid the excessive overshooting. Thus the new reference signal is built by calculating the acceleration and deceleration of the ramp-plateau transition according to the ramp rate involved

$$\text{acceleration (deceleration)} = (-)\ \text{ramp rate}/\Delta t. \quad (7.1)$$

Where Δt is 0.5 s and the reference signal is built the corresponding parabolic acceleration (or deceleration) in the aforementioned transitions (figure 7.4).

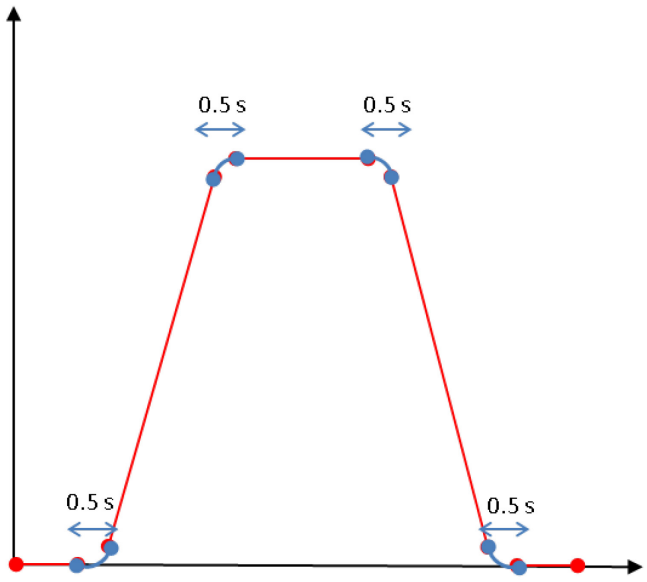

Figure 7.4: The current reference function used for the validation tests.

It has to be pointed out that the current reference signal editing, from the OP point of view, will not be affected by this new aspect, the calculation of the

acceleration (and deceleration) is embedded in the cycle definition and validation process.

7.1.1 Measurements plan

The measurements considered, taken with the new controller settings at the end of March 2013, cover different ramp rates and several maximum current are reached on the transformer secondary. The measurement plan is shown in Table 7.1 reporting for each considered ramp rate all the maximum current reached.

800 A/s	500 A/s	300 A/s	100 A/s
2000 A	2000 A	2000 A	2000 A
5000 A	5000 A	5000 A	5000 A
6000 A	6000 A	6000 A	6000 A
7000 A	7000 A	7000 A	7000 A
10000 A	10000 A	10000 A	10000 A
14000 A	14000 A	14000 A	14000 A
15000 A	15000 A	15000 A	15000 A
16000 A	16000 A	16000 A	16000 A
20000 A	20000 A	20000 A	20000 A
22000 A	22000 A	22000 A	-
23500 A	23500 A	23500 A	23500 A
24500 A	24500 A	24500 A	24500 A
26500 A	26500 A	26500 A	26500 A

Table 7.5: Set of measurement with the new reference and control configuration.

Then several system responses are considered, with particular regard to the high plateau overshooting. All the overshooting values, both as absolute value and percentage, are aggregated in Table 7.2.

Ramp Rate (A/s) / I Max A	800	500	300	100
2000	10 A - 0.5%	5 A - 0.25	3 A - 0.15%	1 A - 0.05%
5000	11 A - 0.22%	8 A - 0.16%	4 A - 0.08%	1 A - 0.02%
6000	10 A - 0.17%	6 A - 0.1%	3 A - 0.05%	-1 A - 0.016%
7000	6 A - 0.086%	4 A - 0.057%	2 A - 0.029%	-1 A - 0.014%

10000	8 A - 0.08%	4 A - 0.04%	1 A - 0.01%	-1 A - 0.01%
14000	7 A - 0.05%	4 A - 0.029%	2 A - 0.014%	-1 A - 0.007%
15000	5 A - 0.033%	3 A - 0.02%	2 A - 0.013%	-2 A - 0.013%
16000	6 A - 0.037%	3 A - 0.018%	2 A - 0.125%	-1 A - 0.006%
20000	7 A - 0.035%	3 A - 0.015%	1 A - 0.005%	-2 A - 0.01%
22000	6 A - 0.027%	4 A - 0.018%	2 A - 0.009%	-
23500	-7 A - 0.03%	2 A - 0.008%	0.5 A - 0.002%	-2 A - 0.009%
24500	8 A - 0.033%	3 A - 0.012%	2 A - 0.008%	-1 A - 0.004%
26500	7 A - 0.026%	4 A - 0.015%	3 A - 0.011%	-2 A - 0.008%

Table 7.2: Aggregated overshooting values from the validation measurements.

The overshooting phenomenon, in this configuration is still more evident when at higher ramp rate and slightly at lower current. The magnitude of the magnitude value observed keeps almost the same for all the measurement at the same ramp rate, thus the percentage value decrease drastically with the current increasing. The negative overshooting values are to be read as undershooting effect; the relative percentage value is calculated considering the absolute value of the effect. Further detail on the system response can be observed in the presented measurement data, starting from the 800 A/s ramp rate measurements at 2 kA maximum current (Fig 7.5).

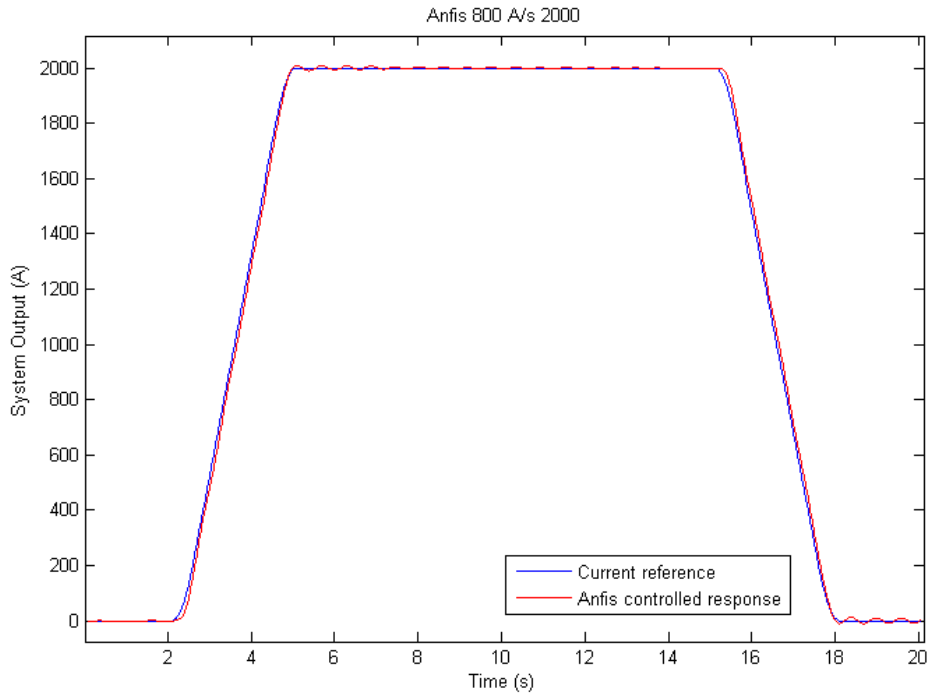

Figure 7.5: The system response for a 800 A/s ramp rate reference up to 2000 A.

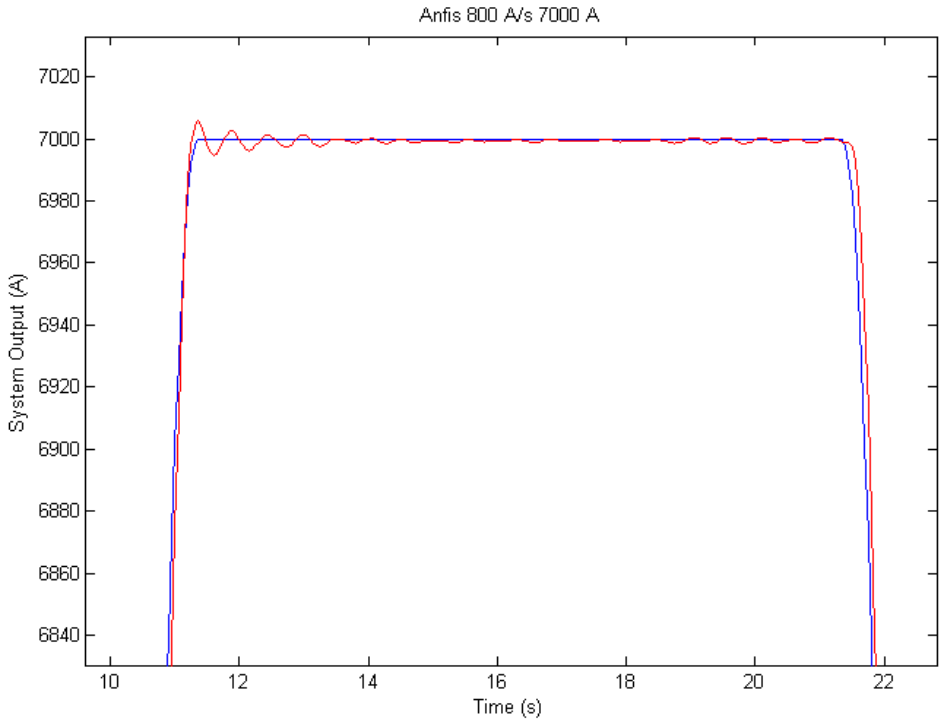

Figure 7.6: The system response for a 800 A/s ramp rate reference up to 7000 A.

It is worthy to note that for the higher current measurements the high plateau region is evidenced in order to give a better perspective on the system dynamic on that region as it shown in the 7 kA response (Fig. 7.6) and in the 22 kA one (Fig 7.7).

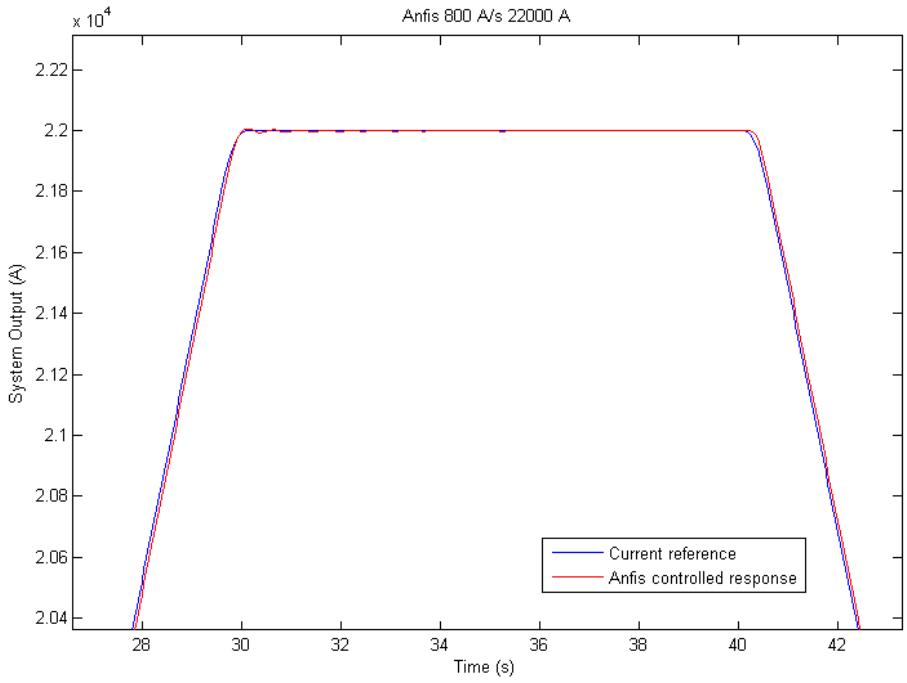

Figure 7.7: The system response for a 800 A/s ramp rate reference up to 22000 A.

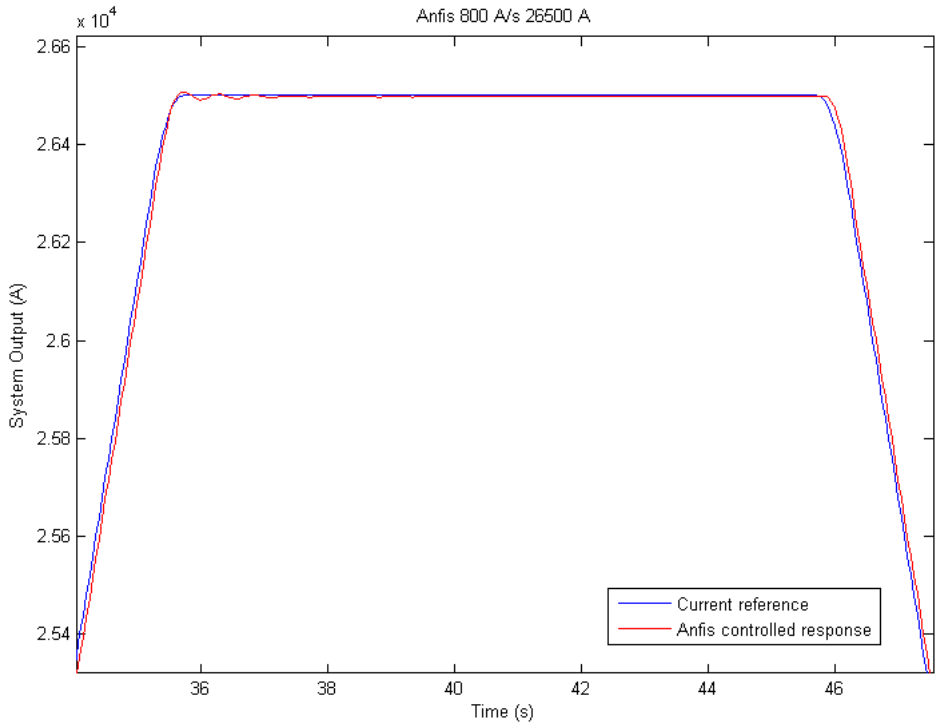

Figure 7.8: The system response for a 800 A/s ramp rate reference up to 26500 A.

The maximum current reached is 26.5 kA for 800 A/s references (Fig 7.8) and for the other ramp rates. Then three of the 500 A/s ramp rate measurements are then reported, starting from the 2 kA response (Fig. 7.9).

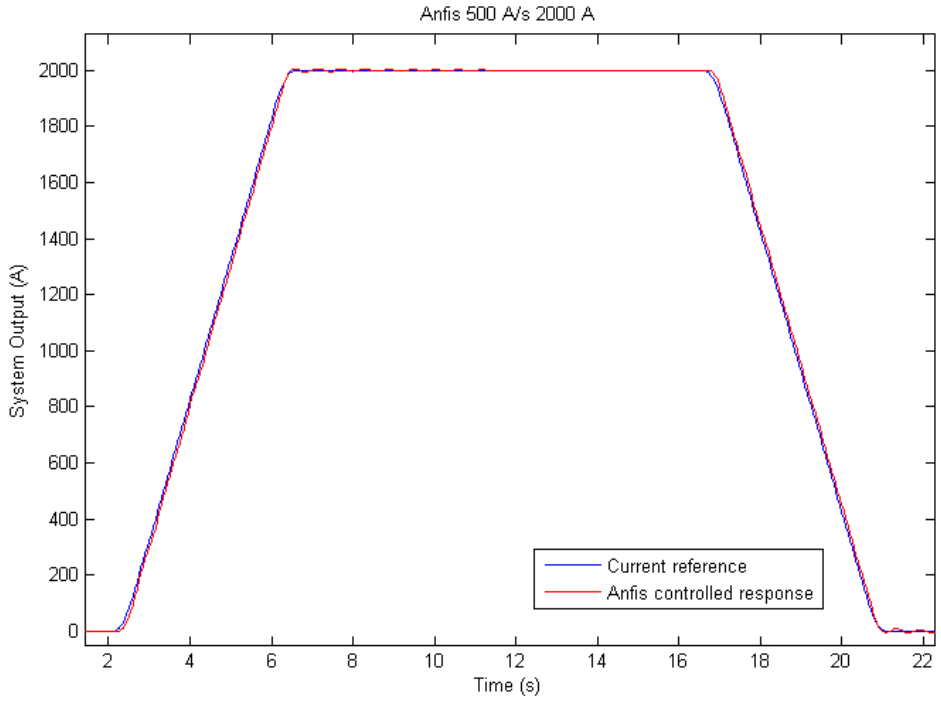

Figure 7.9: The system response for a 500 A/s ramp rate reference up to 2000 A.

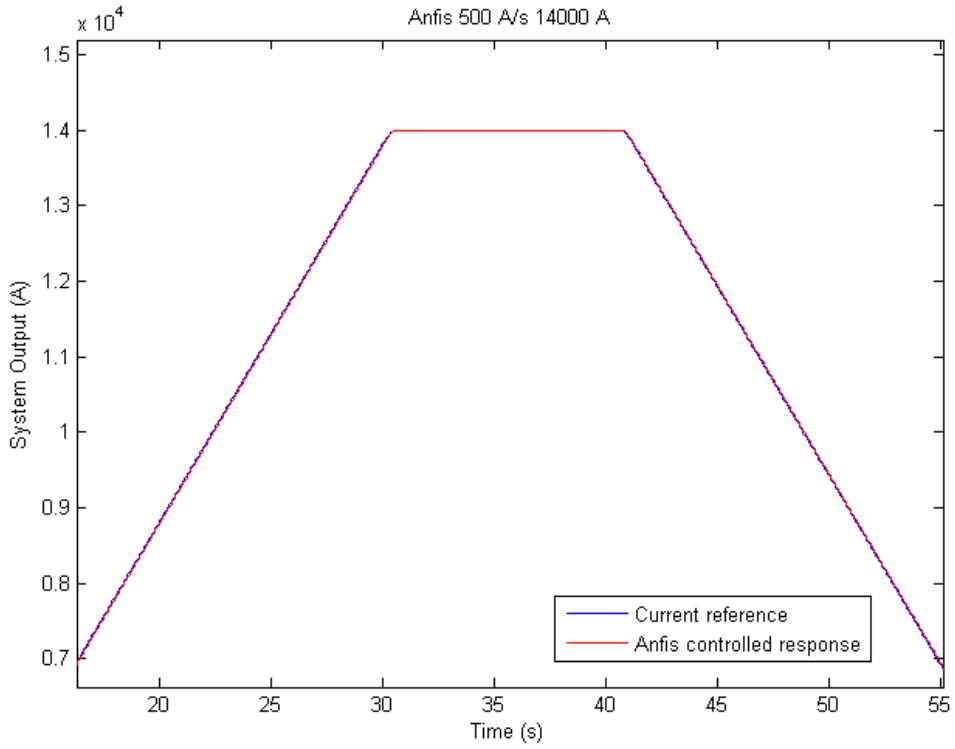

Figure 7.10: The system response for a 500 A/s ramp rate reference up to 14000 A.

.

The data of the 14 kA (Fig. 7.10) and 26.5 kA (Fig. 7.11) measurement is also showed.

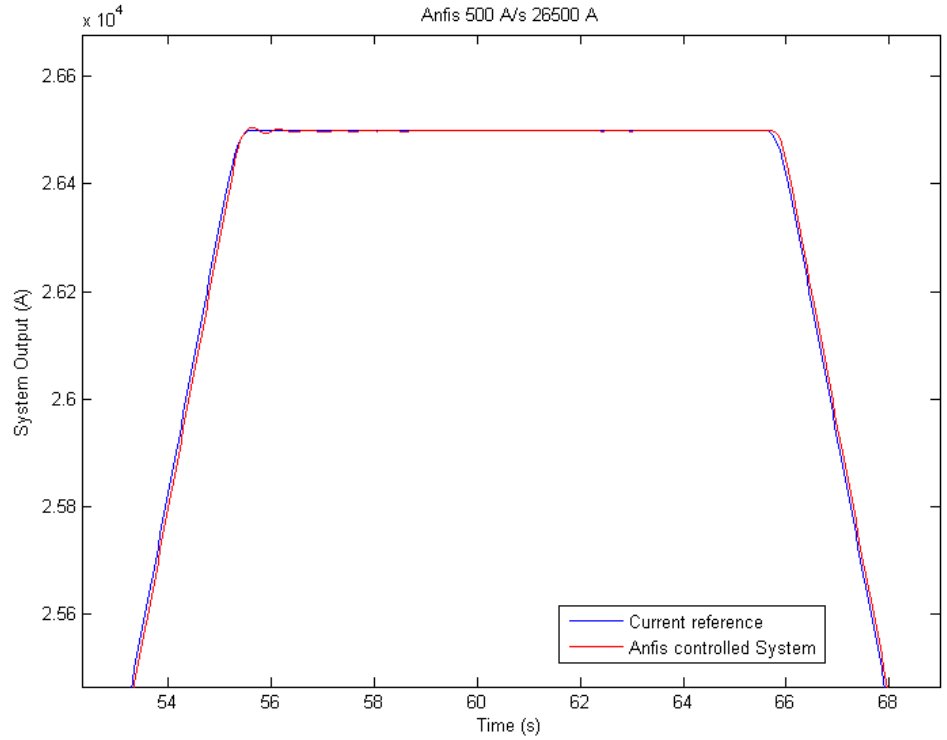

Figure 7.11: The system response for a 500 A/s ramp rate reference up to 26500 A.

More results regarding the 300 A/s ramp rate measurements are presented. In particular the 6 kA response (Fig. 7.12) and 23.5 kA one (Fig. 7.13) are showed.

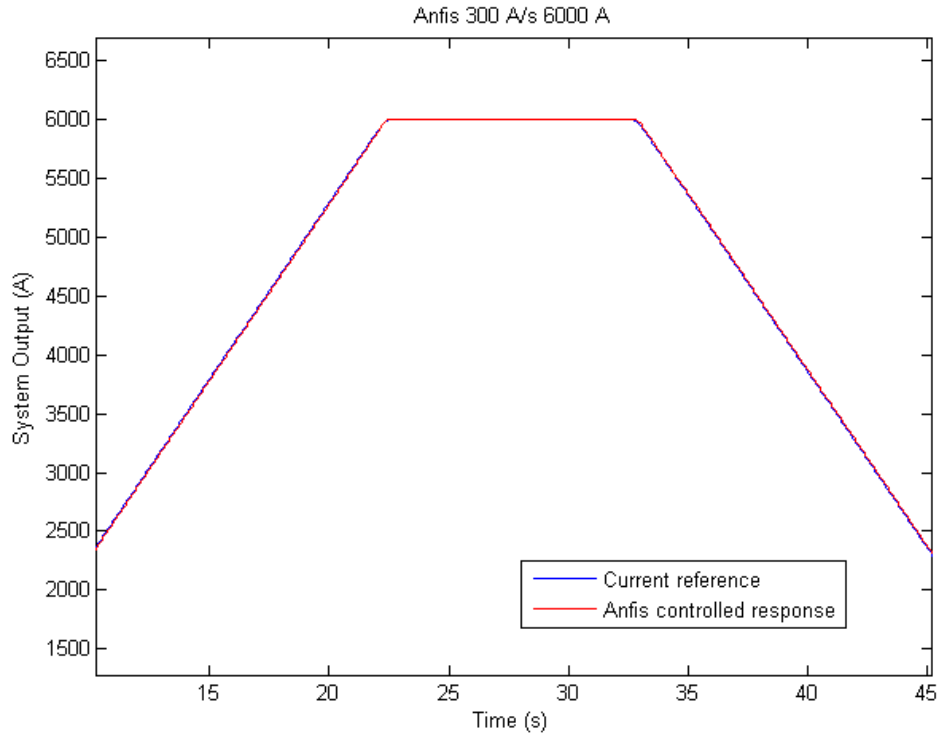

Figure 7.12: The system response for a 300 A/s ramp rate reference up to 6000 A.

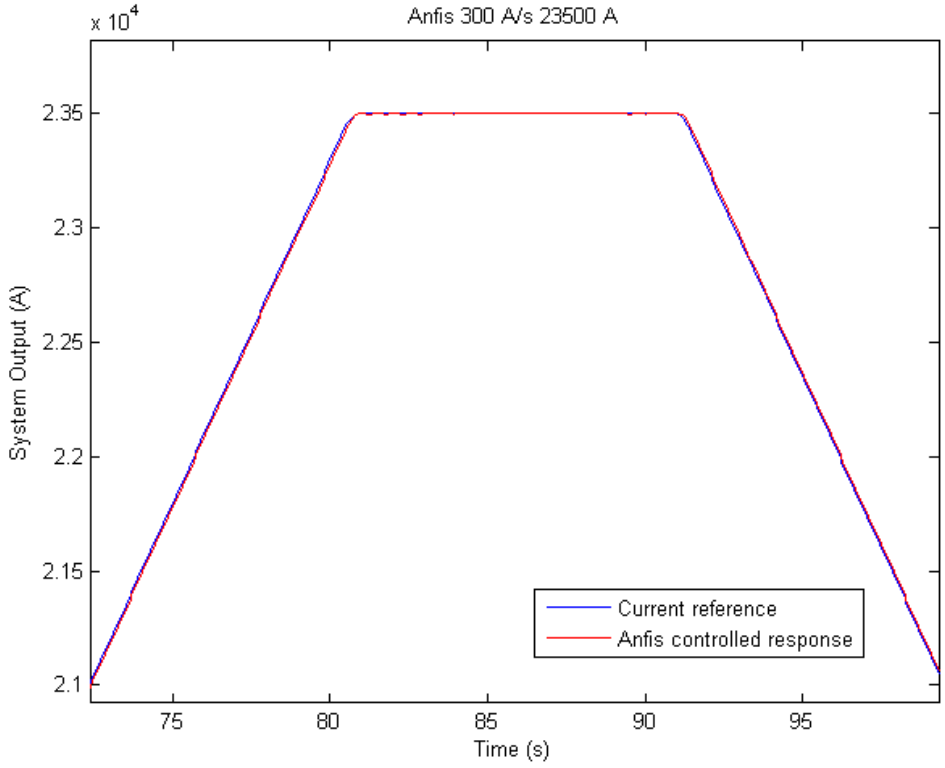

Figure 7.13: The system response for a 300 A/s ramp rate reference up to 23500 A.

Finally two from the 100 A/s ramp rate system responses are presented, namely the response at 2 kA (Fig. 7.14) and the 16 kA characteristic (Fig. 7.15).

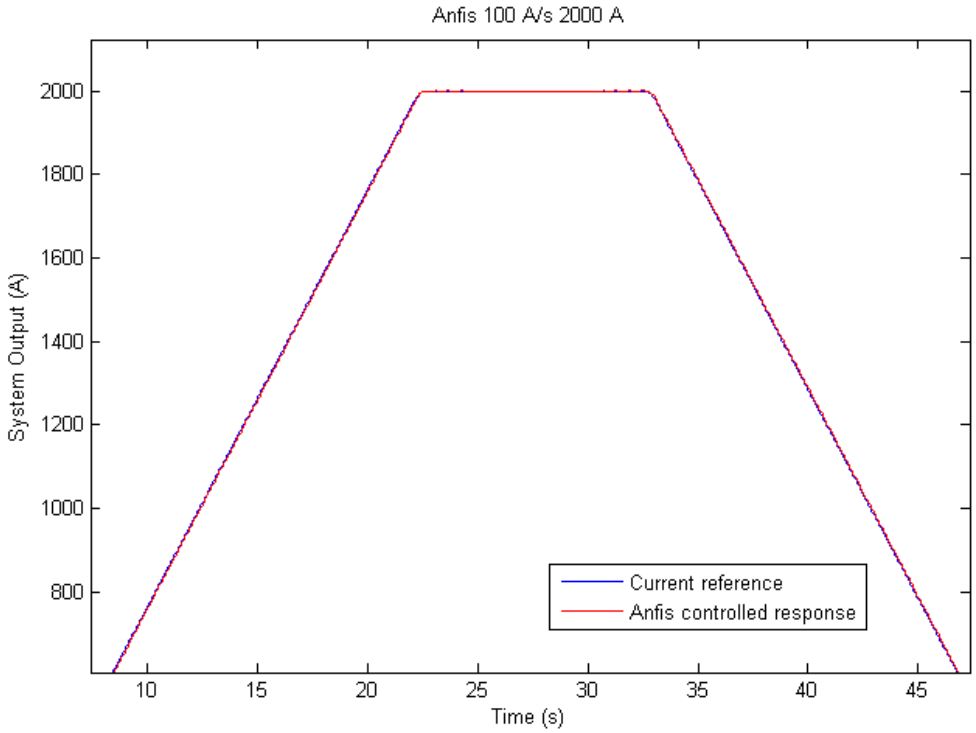

Figure 7.14: The system response for a 100 A/s ramp rate reference up to 2000 A.

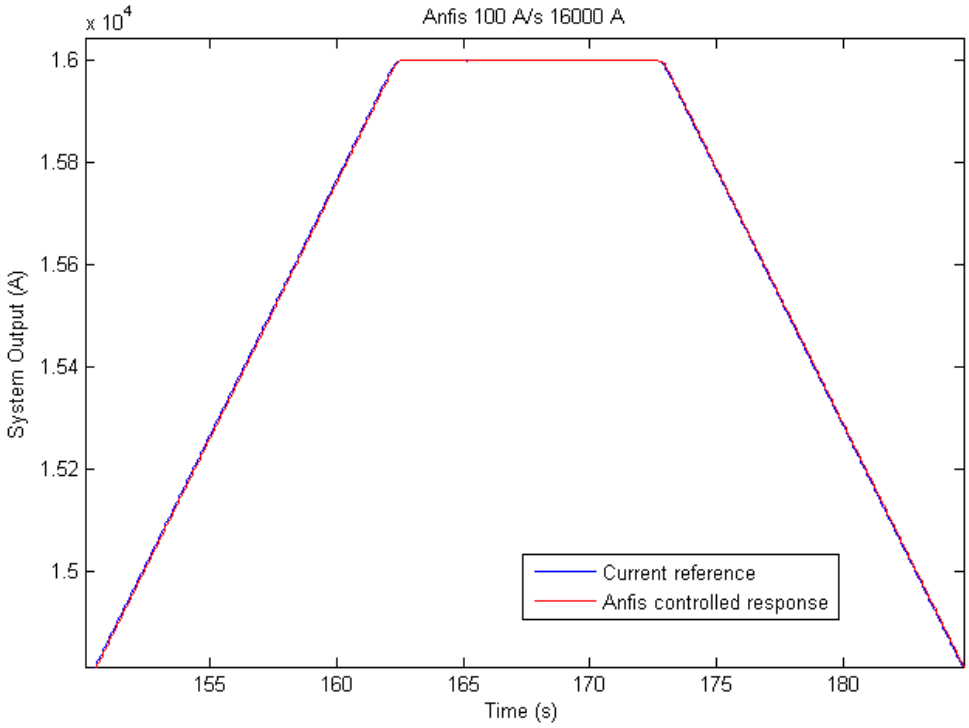

Figure 7.15: The system response for a 100 A/s ramp rate reference up to 16000 A.

7.2 State of the art comparison

The effective performance of the proposed system can be better evaluated if compared with the state of the art reference. For this purpose several measurements were performed with the available control system [2] and then compared with the proposed system ones. The data provided includes the system response, the absolute error and the RMSE for both systems. The current references considered for the comparison are summarized in Table 7.3.

800 A/s	500 A/s	300 A/s	100 A/s
5000 A	5000 A	5000 A	5000 A
10000 A	10000 A	10000 A	10000 A
15000 A	15000 A	15000 A	15000 A
20000 A	20000 A	20000 A	20000 A
24500 A	24500 A	24500 A	24500 A

Table 7.3: Set of measurement for the state of the art control comparison.

First the 800 A/s ramp rate measurements are showed, starting from the 5 kA responses (Fig. 7.16). Here the simulation results are confirmed regarding the slight overshoot for the proposed control but not for the higher PI one.

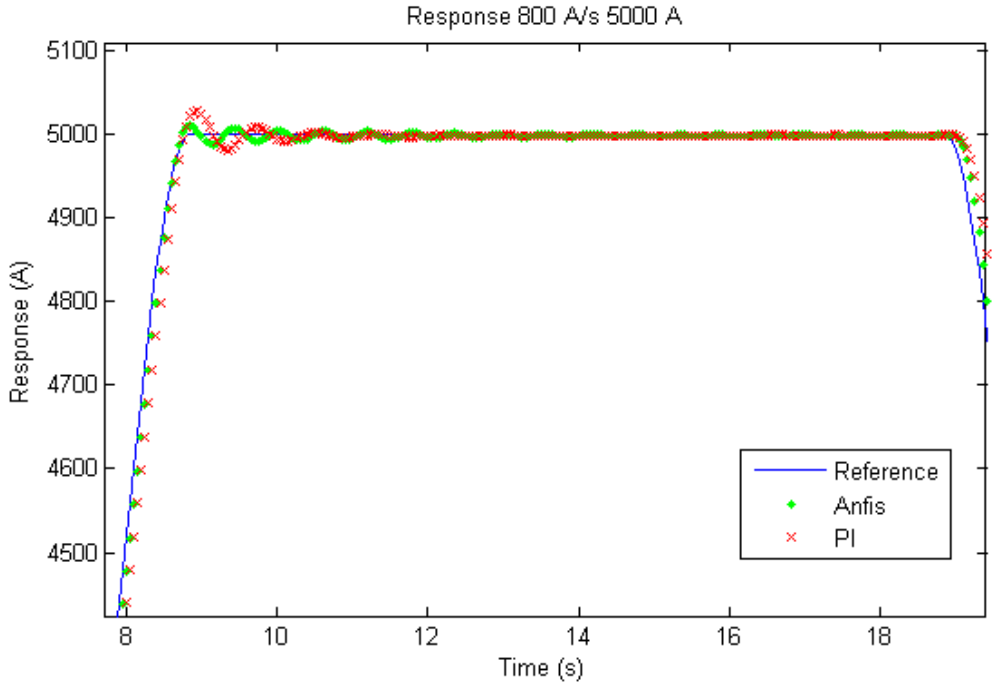

Figure 7.16: ANFIS and PI controlled system responses for a 800 A/s ramp rate reference up to 5000 A.

The higher overshooting as well as the almost doubled error is observable both from the absolute error plot (Fig 7.17) and from the RMSE curves (Fig. 7.18).

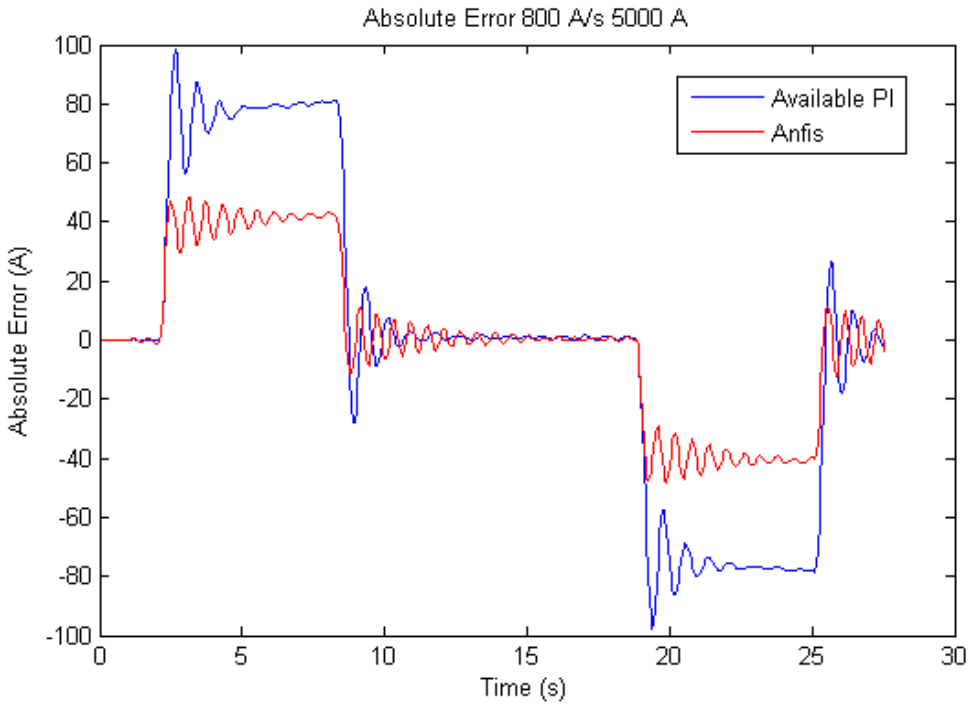

Figure 7.17: ANFIS and PI controlled system abs. errors for a 800 A/s ramp rate reference up to 5000 A.

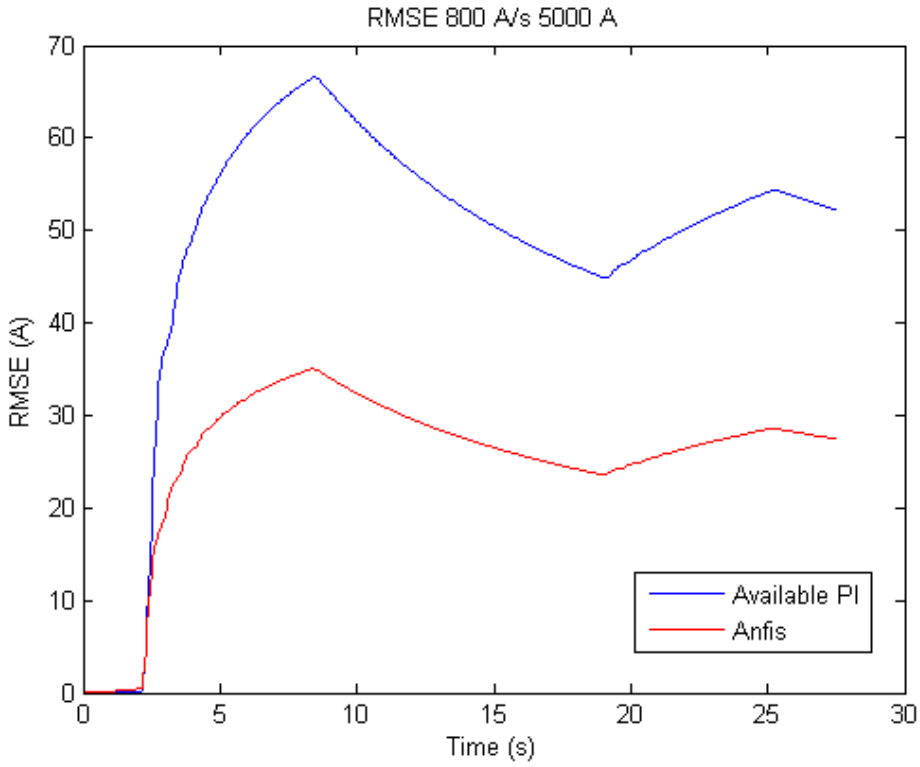

Figure 7.18: ANFIS and PI controlled system RMSE for a 800 A/s ramp rate reference up to 5000 A.

The behaviour highlighted in the previous responses is possibly more stressed in the 15 kA responses (Fig 7.19) where the PI overshoot seems much higher.

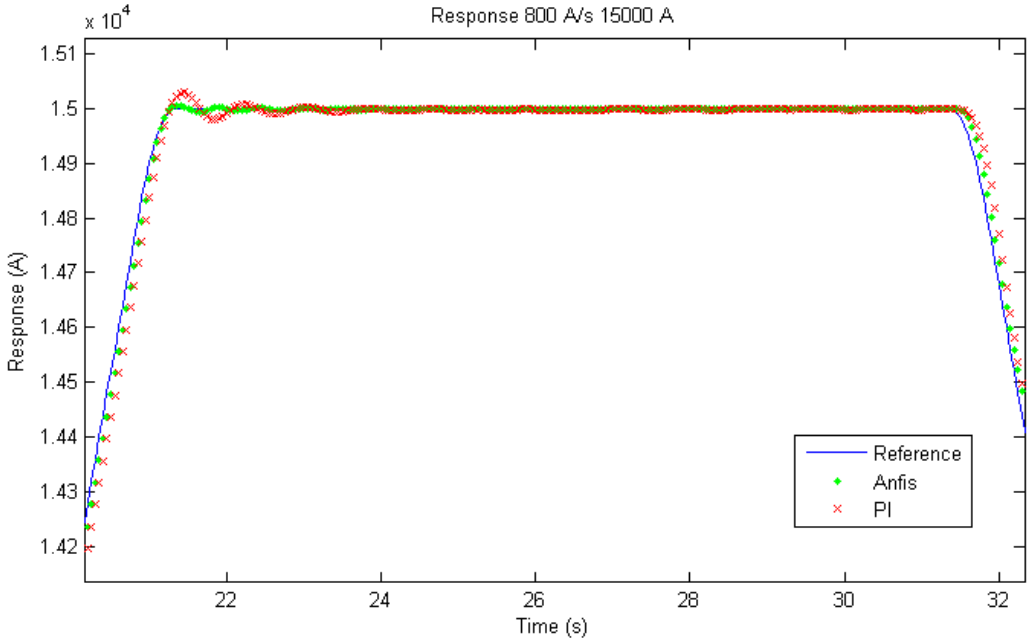

Figure 7.19: ANFIS and PI controlled system responses for a 800 A/s ramp rate reference up to 15000 A.

Even the absolute error difference (Fig. 7.20) is wider.

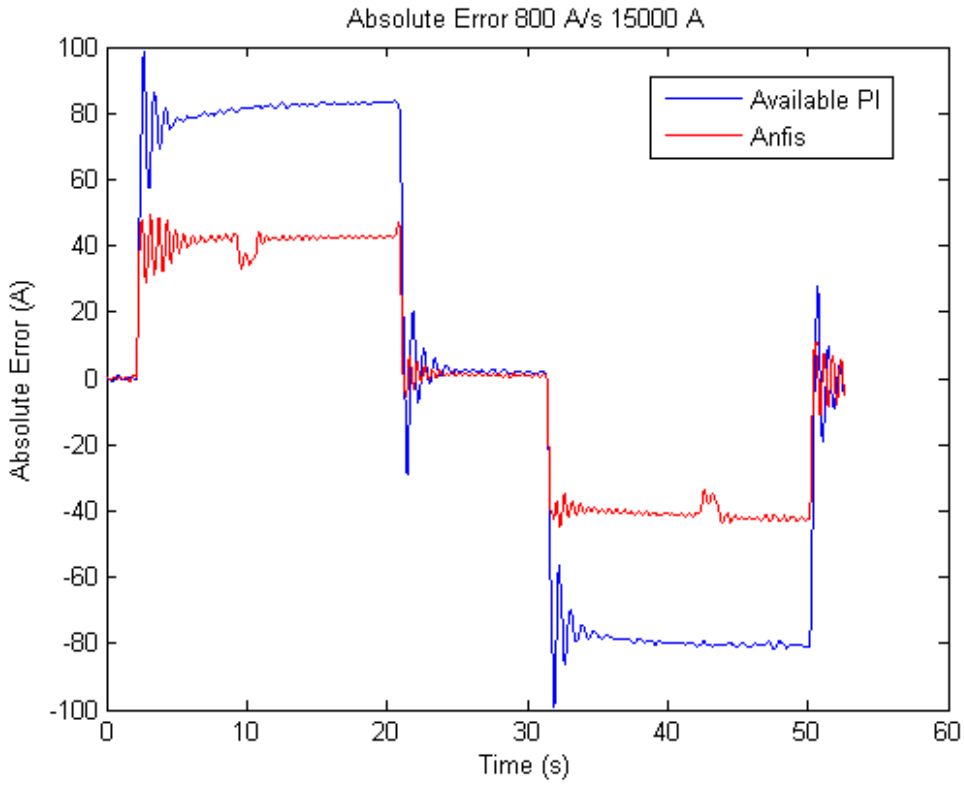

Figure 7.20: ANFIS and PI controlled system abs. error for a 800 A/s ramp rate reference up to 15000 A.

The RMSE plot (Fig. 7.21) for this measure confirm the almost fifty per cent of difference between the PI error and the proposed controller one.

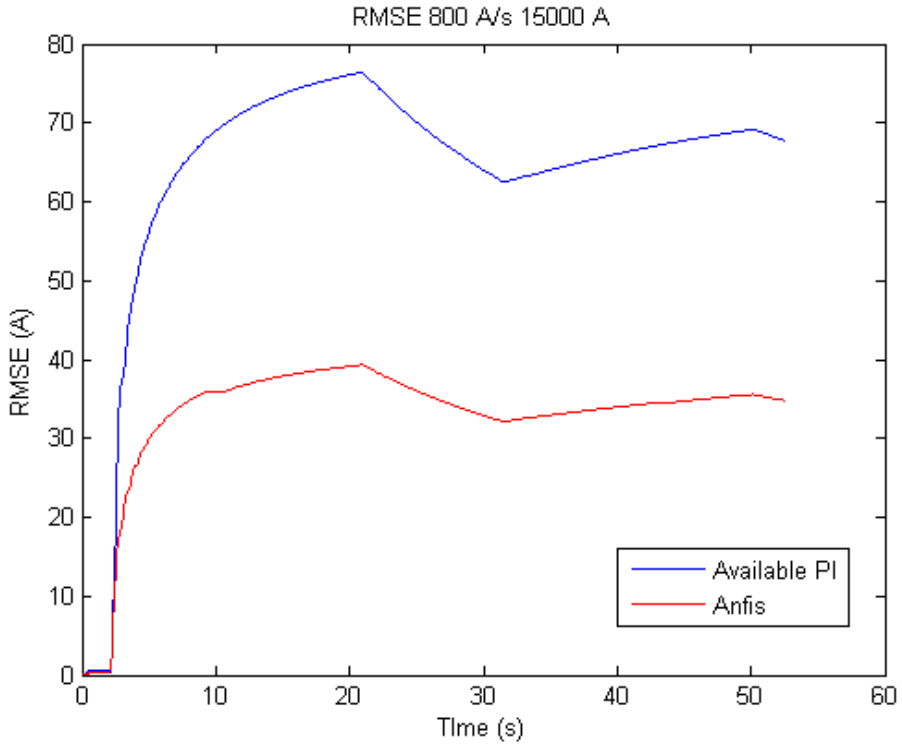

Figure 7.21: ANFIS and PI controlled system RMSE for 800 A/s ramp rate reference up to 15000 A.

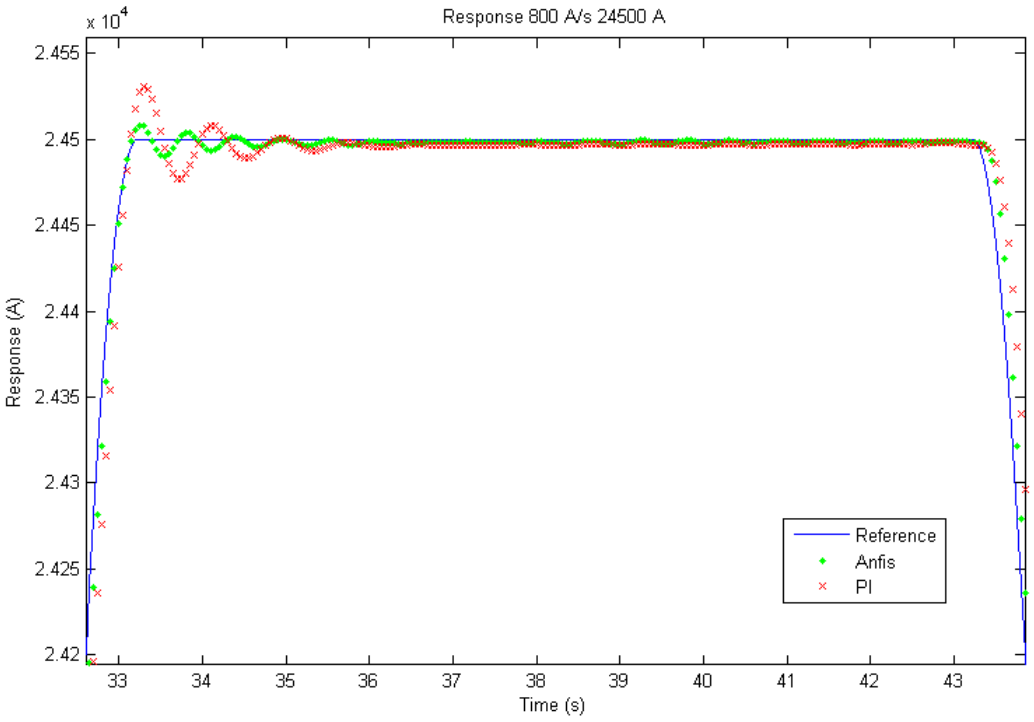

Figure 7.22: ANFIS and PI controlled system responses for a 800 A/s ramp rate reference up to 24500 A.

The higher current responses comparison is given at 24.5 kA (Fig. 7.22) where the overshoot difference is much more evident. Such a difference can be appreciated also in the absolute error plot (Fig. 7.23), while the RMSE plot (Fig. 7.24) provides the same indication given by the previous measurement.

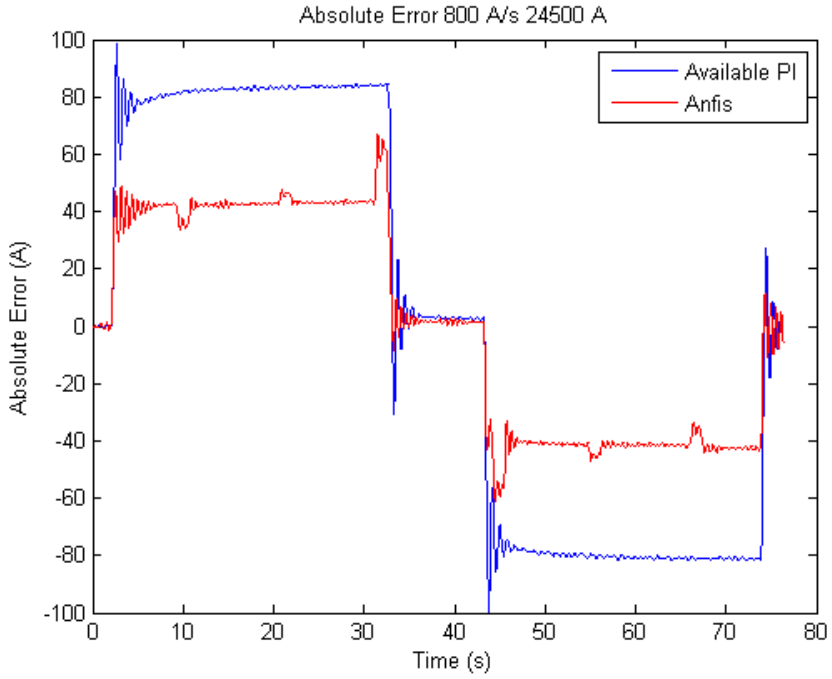

Figure 7.23: ANFIS and PI controlled system abs. error for a 800 A/s ramp rate reference up to 24500 A

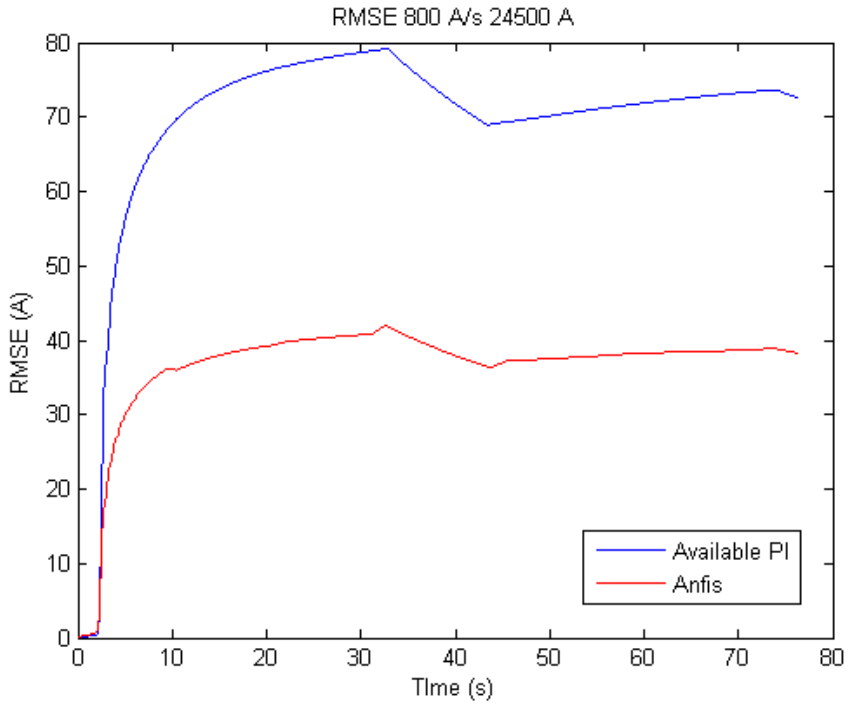

Figure 7.24: ANFIS and PI controlled system RMSE for 800 A/s ramp rate reference up to 24500 A.

The comparison measurements shown are with the 500 A/s ramp rate starting from the 10 kA responses (Fig. 7.25). The PI response does provide yet a gentle response while the Anfis has slight overshoot and follows closely the ramp. This consideration are assessed by the absolute error plot (Fig. 7.26) and the RMSE comparison (Fig. 7.27).

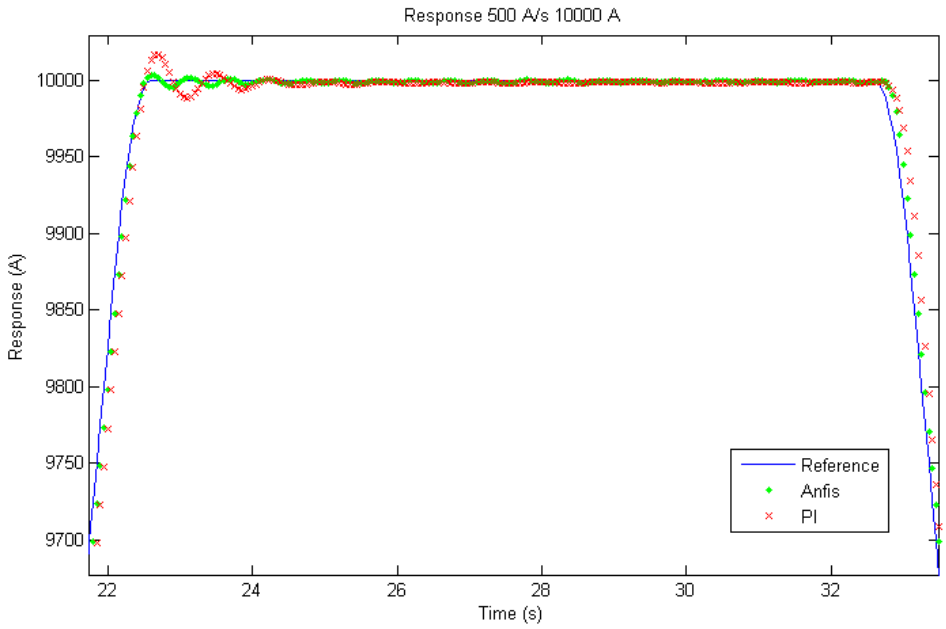

Figure 7.25: ANFIS and PI controlled system responses for a 500 A/s ramp rate reference up to 10000 A.

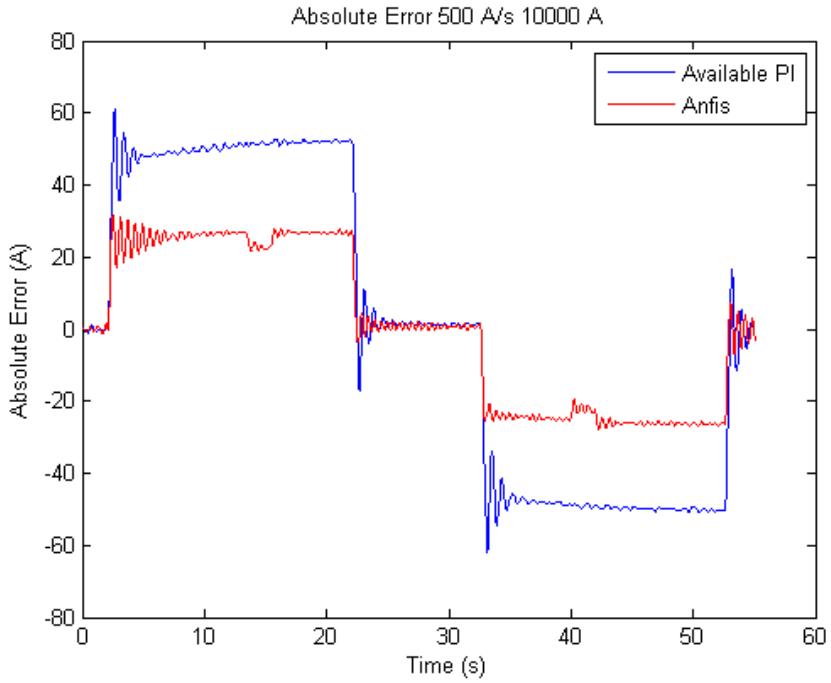

Figure 7.26: ANFIS and PI controlled system abs. error for a 500 A/s ramp rate reference up to 10000 A.

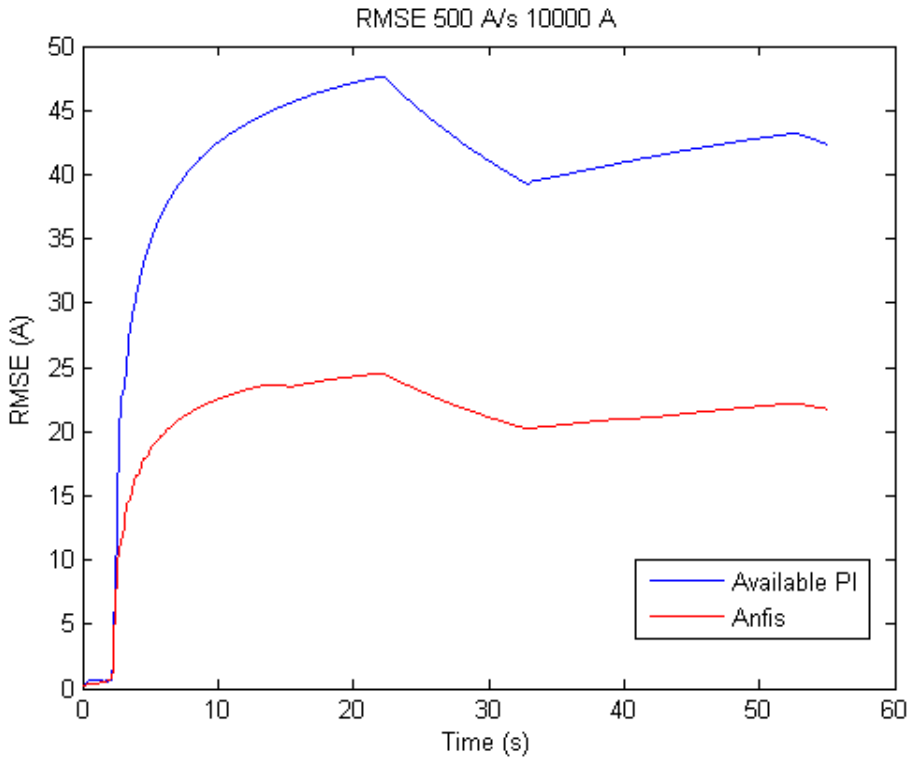

Figure 7.27: ANFIS and PI controlled system RMSE for 500 A/s ramp rate reference up to 10000 A.

The second comparison at 500 A/s proposed is at 20 kA and the responses (Fig. 7.28) shows again the same behavior.

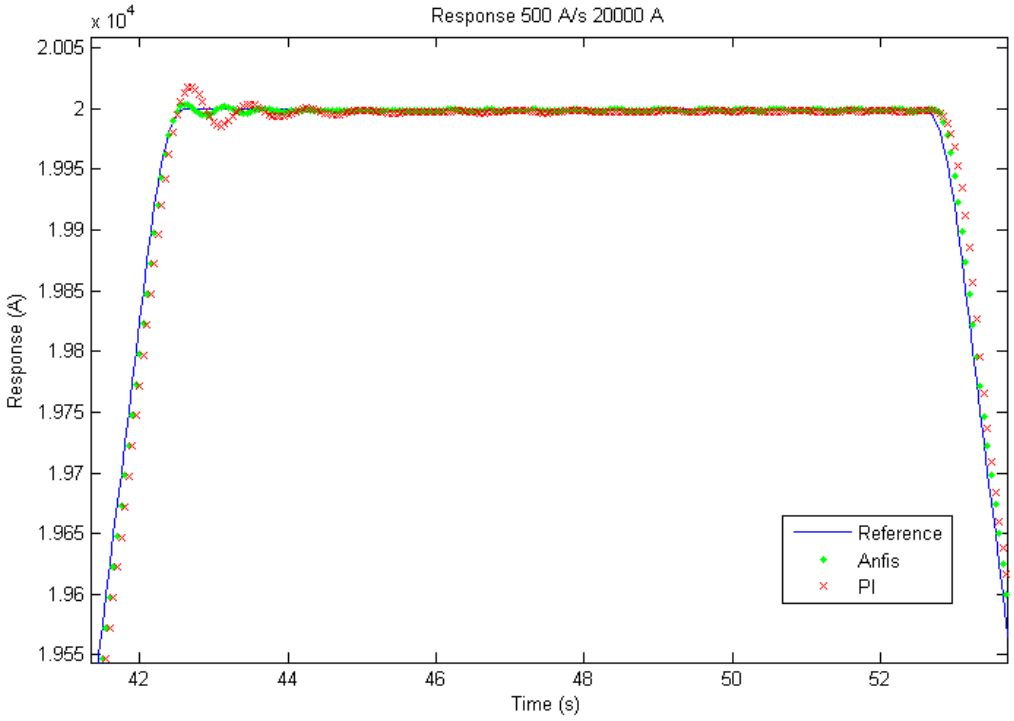

Figure 7.28: ANFIS and PI controlled system responses for a 500 A/s ramp rate reference up to 20000 A.

The ramp-plateau transition differences are appreciable also from the absolute error (Fig. 7.29) data, while the ratio between the errors is evident in the RMSE comparison (Fig. 7.30)

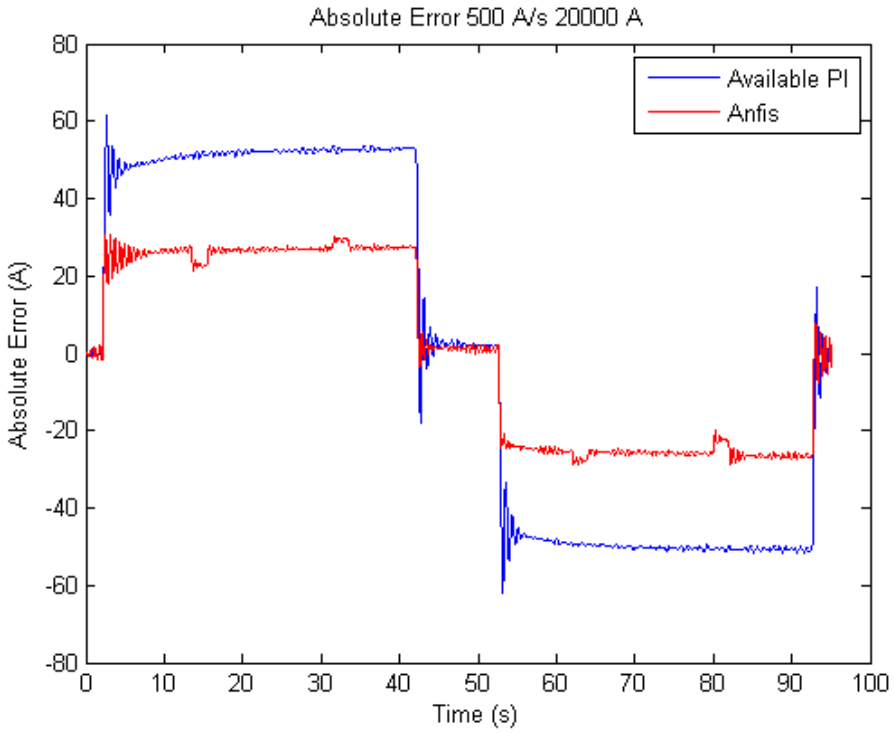

Figure 7.29: ANFIS and PI controlled system abs. error for a 500 A/s ramp rate reference up to 20000 A.

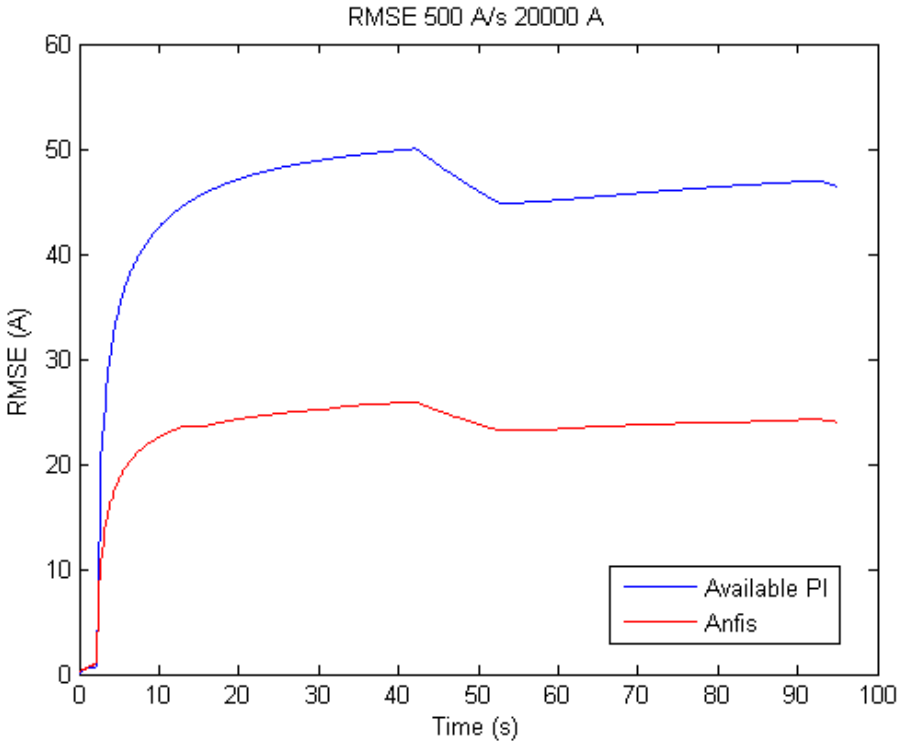

Figure 7.30: ANFIS and PI controlled system RMSE for 500 A/s ramp rate reference up to 20000 A.

Two more comparisons of measurements at 300 A/s ramp rate are provided. The former is between responses at 5 kA (Fig. 7.31) where the difference is slightly decreased due to the gentler ramp rate.

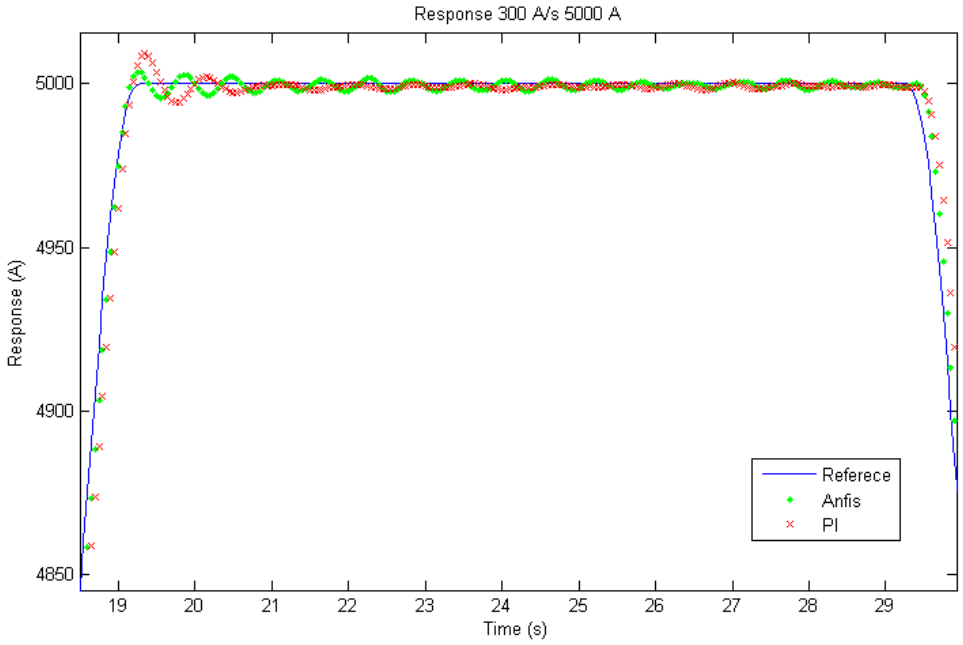

Figure 7.31: ANFIS and PI controlled system responses for a 300 A/s ramp rate reference up to 5000 A.

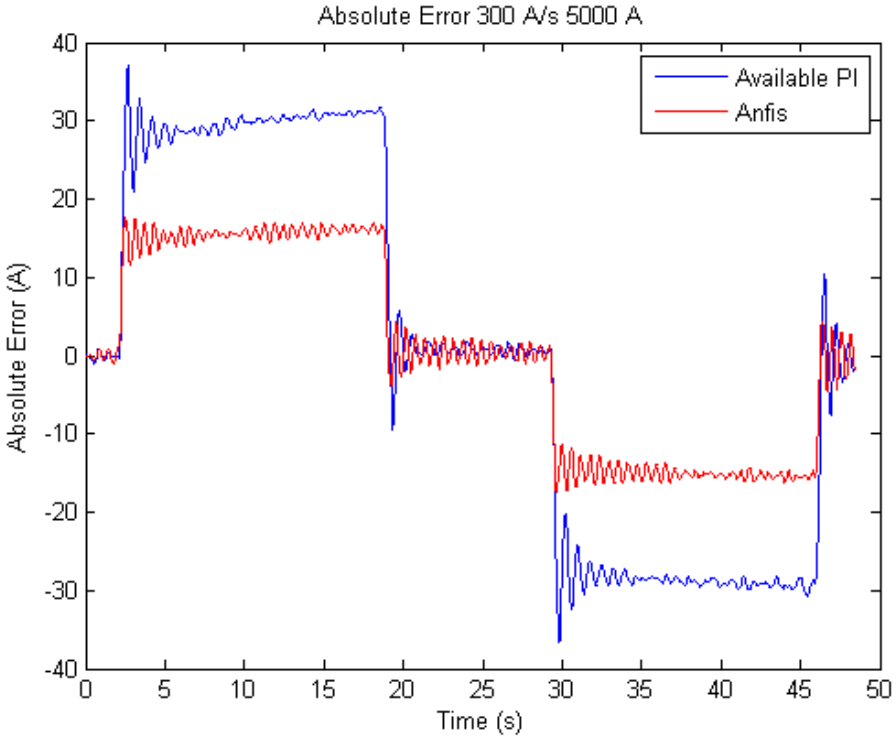

Figure 7.32: ANFIS and PI controlled system abs. error for a 300 A/s ramp rate reference up to 5000 A.

The controller errors are decreased proportionally as the absolute error curves (Fig. 7.32) and the RMSE characteristics (Fig. 7.33).

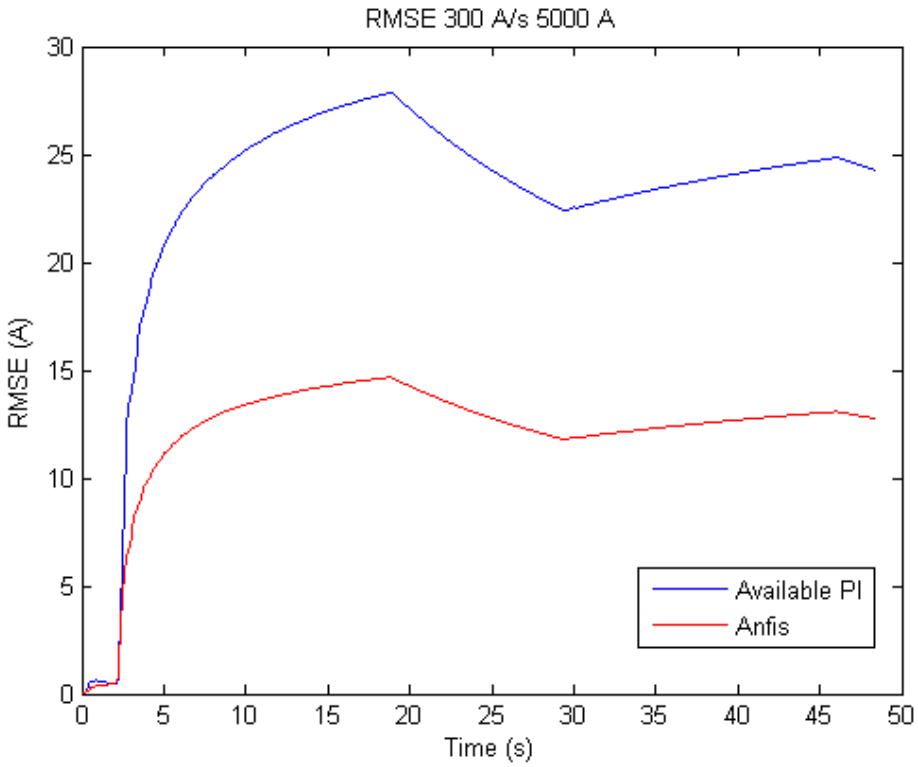

Figure 7.33: ANFIS and PI controlled system RMSE for 300 A/s ramp rate reference up to 5000 A.

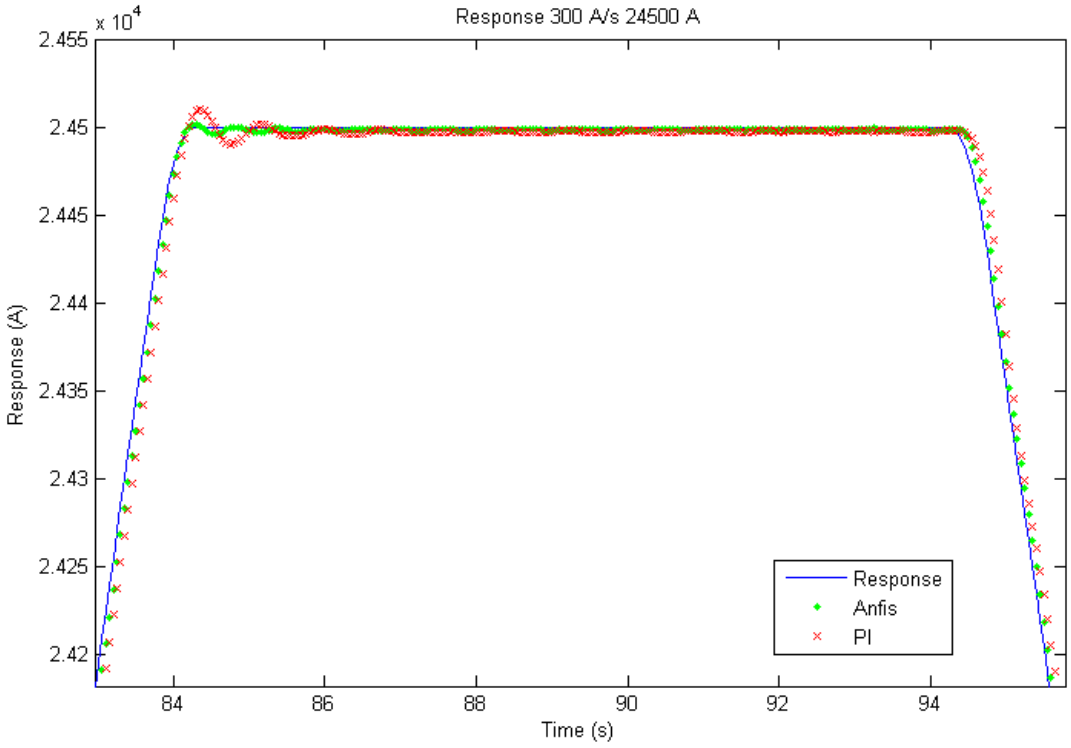

Figure 7.34: ANFIS and PI controlled system responses for a 300 A/s ramp rate reference up to 24500 A.

The latter shows the responses at 24.5 kA (Fig 7.34) where a strong overshoot difference is still appreciable even at a lower ramp rate. The error oscillations (Fig. 7.35) on the other hand are tighter. A smoother trend in the error can be observed in the RMSE plot (Fig. 7.36).

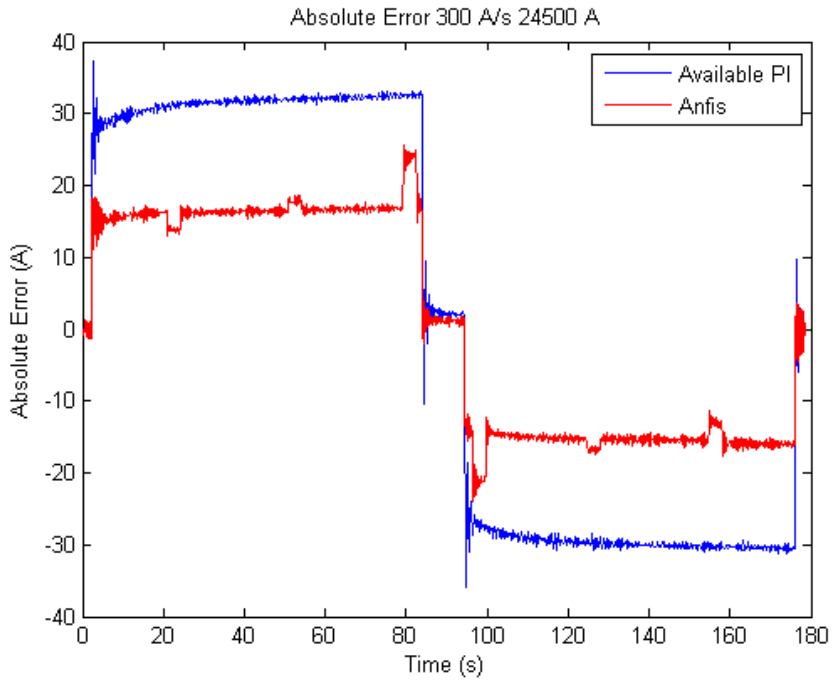

Figure 7.35: ANFIS and PI controlled system abs. error for 300 A/s ramp rate reference up to 24500 A.

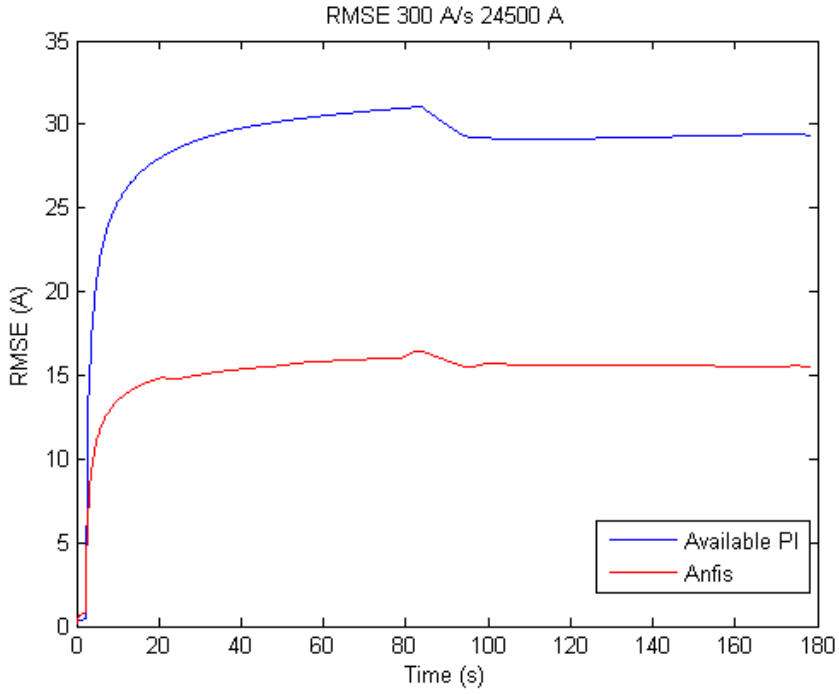

Figure 7.36: ANFIS and PI controlled system RMSE for 300 A/s ramp rate reference up to 24500 A.

The comparison is then completed with two measurements at 100 A/s ramp rate. The first one carried out at 10 kA (Fig. 7.37) presents a more oscillating behavior of the PI on the plateau while the Anfis keeps the overshoot low. The errors ratio still on the same value can be evaluated on the absolute error plot (Fig. 7.38) and on the RMSE characteristic (Fig. 7.39).

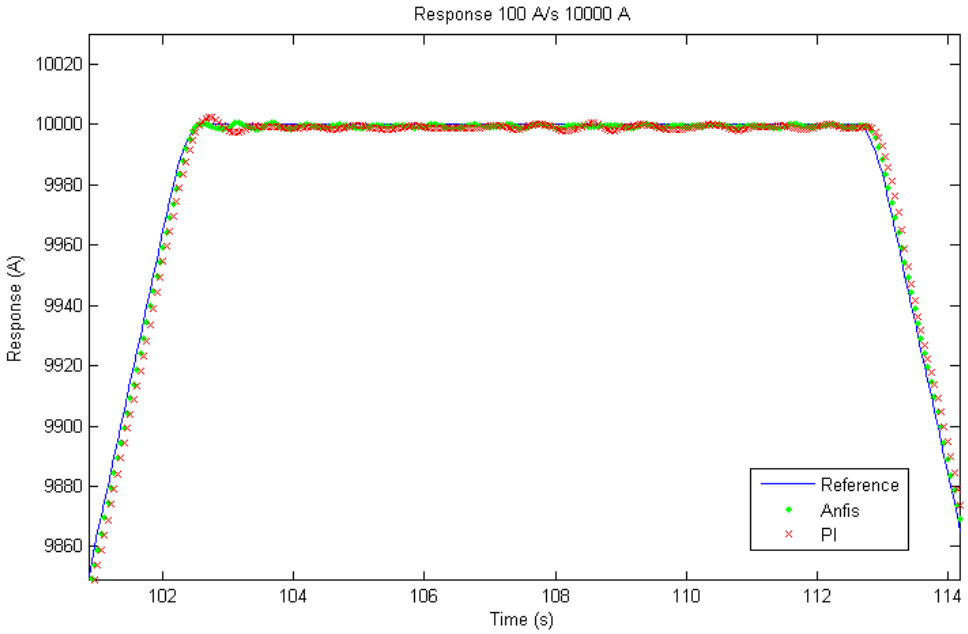

Figure 7.37: ANFIS and PI controlled system responses for a 100 A/s ramp rate reference up to 10000 A.

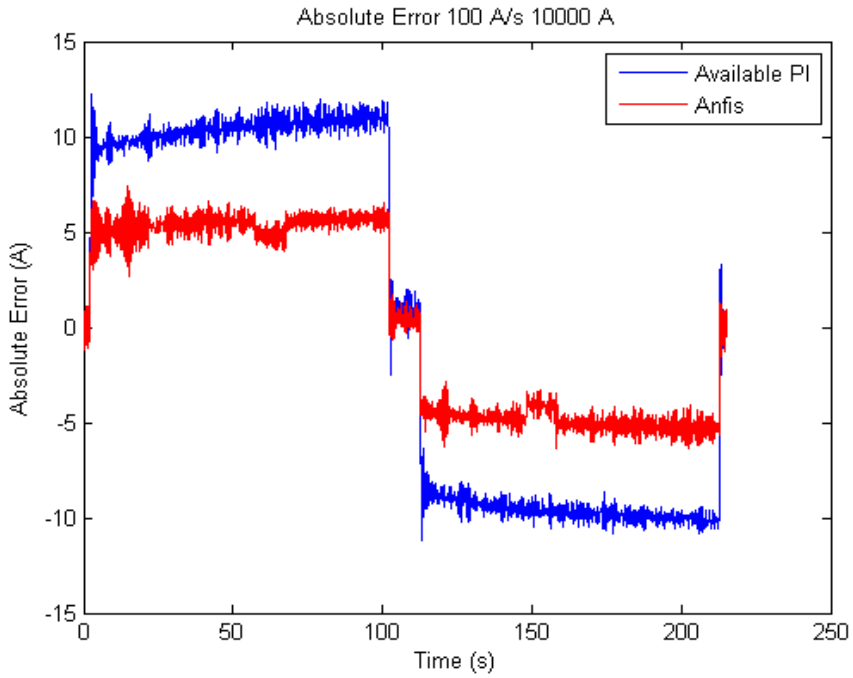

Figure 7.38: ANFIS and PI controlled system abs. error for 100 A/s ramp rate reference up to 10000 A.

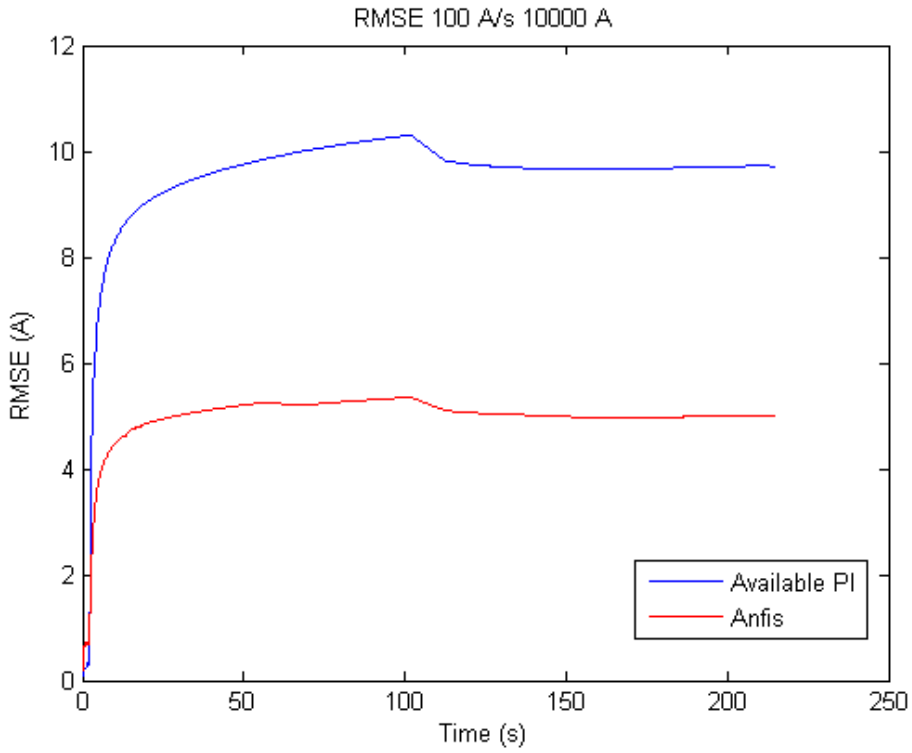

Figure 7.39: ANFIS and PI controlled system RMSE for 100 A/s ramp rate reference up to 10000 A.

The last response comparison provided is the one at 20 kA (Fig. 7.40) with the corresponding absolute error (Fig. 7.41) and RMSE (Fig. 7.42) plots confirming the consideration pointed out in the above measurements.

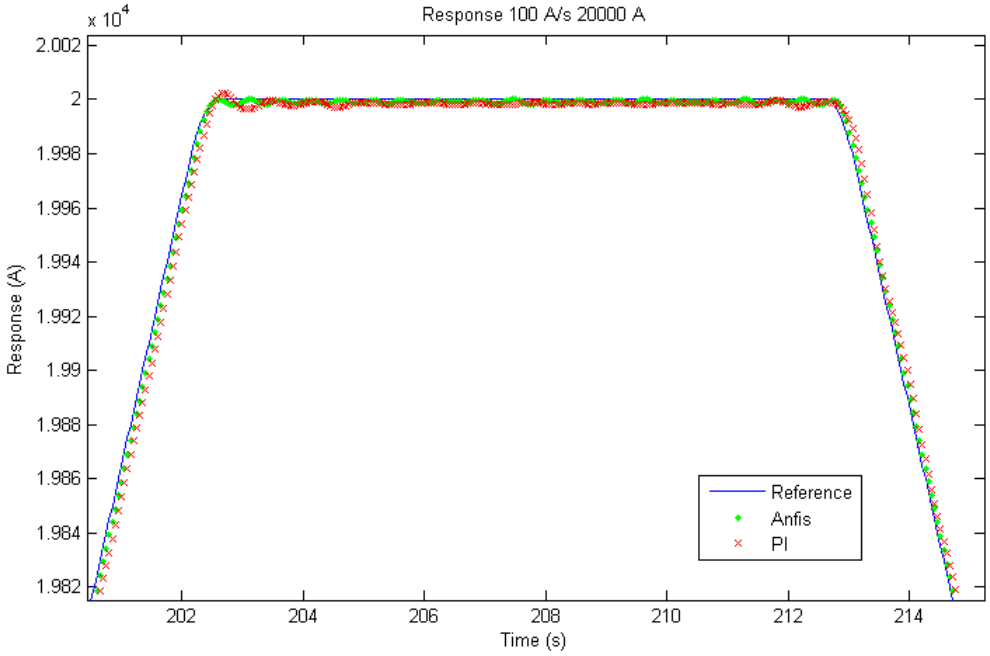

Figure 7.40: ANFIS and PI controlled system responses for a 100 A/s ramp rate reference up to 20000 A.

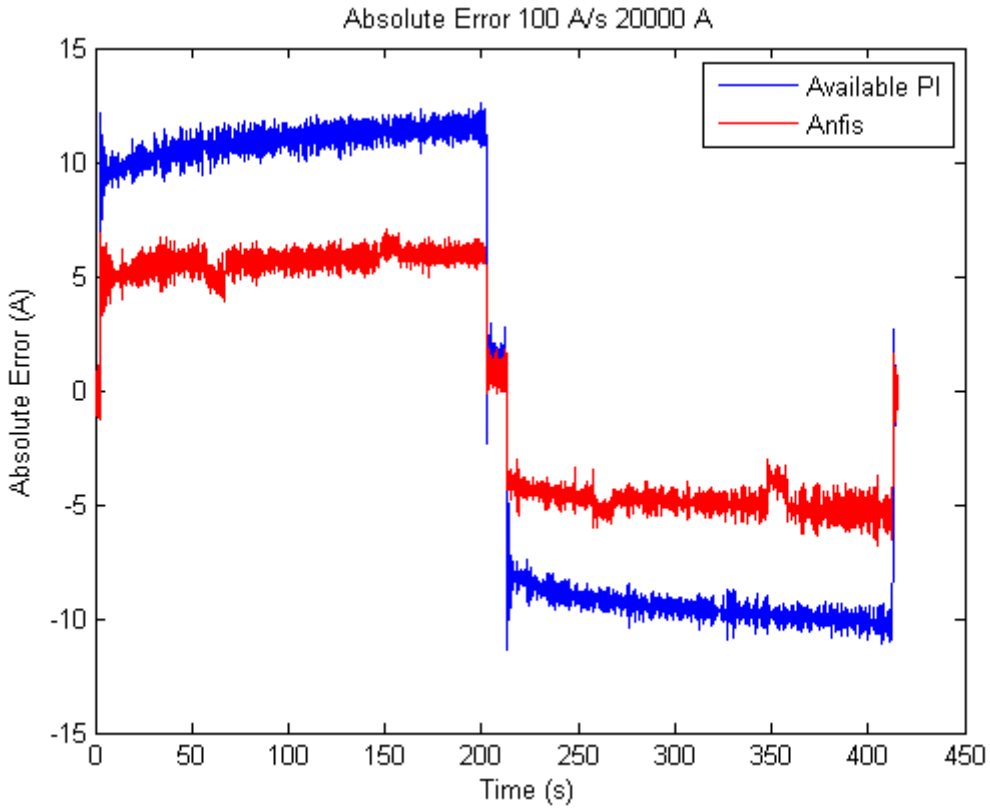

Figure 7.41: ANFIS and PI controlled system abs. error for 100 A/s ramp rate reference up to 20000 A.

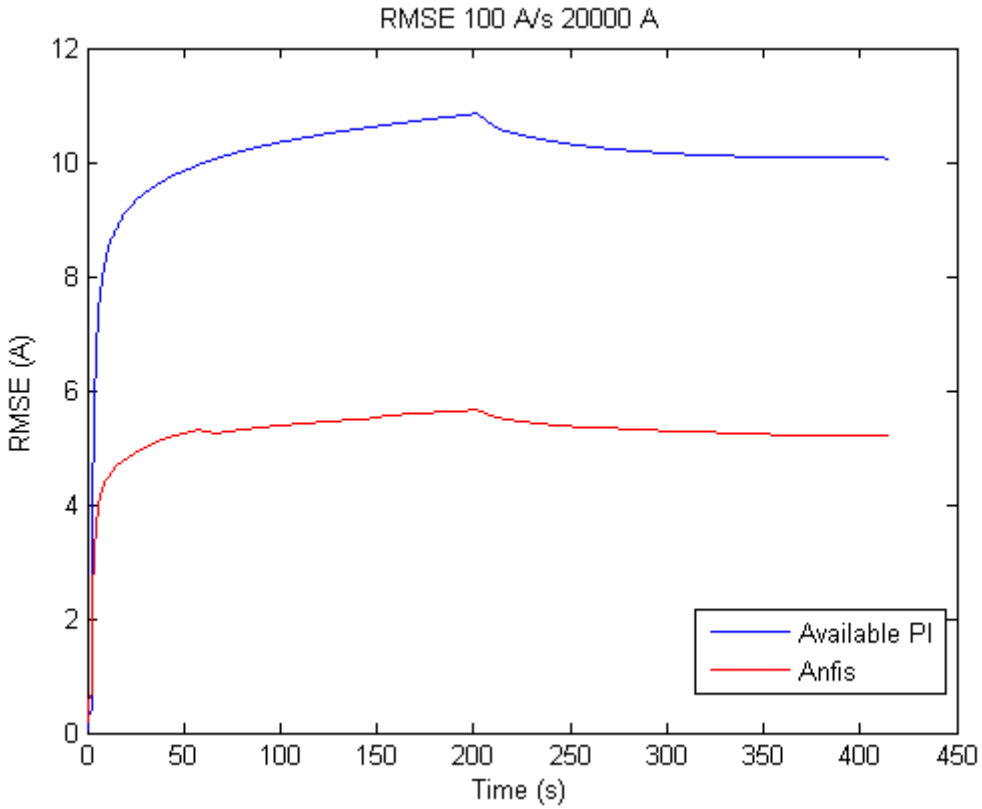

Figure 7.42: ANFIS and PI controlled system RMSE for 100 A/s ramp rate reference up to 20000 A.

The presented data show a common trend regardless the ramp rate and the maximum current of the measurement considered. Under the same conditions the proposed control strategy error is considerably less, sometimes more than 50%, than the state of the art ones. Such an error is accumulated mainly in the ramp phases; in these circumstances the proposed controller shows satisfactory rapidity in following the ramp and negligible oscillations during transitions between ramp and plateau. The information regarding the average error accumulated by the two control strategies is summarized in Table 7.4 and Table 7.5.

Ramp Rate (A/s) / I Max A	800	500	300	100
5000	25.52	18.24	14.15	4.74
10000	30.31	20.77	13.41	4.98
15000	33.08	22.24	14.13	5.14
20000	34.93	23.2	14.62	5.28
24500	36.6	24.15	15.15	5.43

Table 7.4: Aggregated RMSE mean values from the ANFIS comparison measurements.

Ramp Rate (A/s) / I Max A	800	500	300	100
5000	48.36	34.28	22.99	8.85
10000	58.69	40.1	25.92	9.52
15000	64.03	42.96	27.29	9.86
20000	67.38	44.72	28.15	10.11
24500	69.52	45.83	28.72	10.29

Table 7.5: Aggregated RMSE mean values from the PI comparison measurements.

References

[32] T. Shimada, M. Sugimoto, Y. Nagasu, A. Takagi, K. Wada, H. Shimizu, A.Kimura, and S. Meguro, IEEE Trans. Appl. Supercond. 12(1), 1075 (2002).

[33] P. Arpaia, L. Bottura, G. Montenero, S. Le Naour, "Performance improvement of a measurement station for superconducting cable test", *Review of Scientific Instruments*, vol. 83, 095111, 2012.

Appendix A - Lexicon

In this section the lexicon used in the thesis is outlined. The special purpose semantic of the terms is described in order to give further information on the operating domain.

Operator (OP) Is the role of the final user. The OP will start the application and run the test cycles.

Cycle Sequence of segments given as reference to follow from the user to the system.

Segment The smallest unity that an OP can set in Cycle definition phase.

Quench A particular state that the system can reach. There are two kinds of Quench states, according to the events that lead to each of them.

- **Quench_1** Is the border state reached by the system, during a test Cycle, when the superconducting conditions do not apply anymore. Particular recovery routine has to be started to face this issue.
- **Quench_E** Is a soft-error state. The system comes to this state if, after a control of the acquisition boards, a *Quench signal* is read even no test Cycle is running.

Quench signal Signal read from the board when occurs either a Quench_1, or a Quench_E.

Iref Current reference value defined by the OP in the test Cycle.

Vref Voltage value synthesized by the control system, used to drive the current on the superconductor.

Measurement Includes all the action taken and the routines invoked during the voltage measure on the *Secondary* on the *Transformer*.

Control Stands for the control logic that drives the V_{Ref} in order to follow the current reference indicated in the Cycle.

System The whole software system whose requirements are presented in this document.

Technical Network (TN) Particular cabled network installed in CERN's laboratories.

Front End Computer (FEC) Special purpose computer for developing *FESA* application.

Front End Software Architecture (FESA) CERN's interface for devices control and management applications.

Acknowledgment (ACK) Success message sent by the System to the OP during a work session.

Secondary Current (Imes) Is the current value coming from the Measurement on the Secondary of the Transformer.

Light Hardware Control (LHwC) Short fault error check performed by the System during a test Cycle session.

Fast Digital Integrator (FDI) Hardware board that performs the Secondary voltage value integration in order to get the Imes.

Conclusions

At the European Organization for Nuclear Research (CERN) superconductivity is widely used in the development of particle accelerators. In this context the Facility for the Reception of Superconducting Cables (FReSCa) were created to perform test on superconducting cables and determine their properties in particular the critical current. There a measurement system based on a superconducting transformer was introduced. To fulfil the need for this measurement system to be integrated in the FReSCa station system a remote monitoring system is proposed. To provide performance improvements in the sample current control an adaptive current control strategy is designed. Thus the proposed remote monitoring system for cable testing aims to allow the measurement system to be remotely monitored through the FESA class interface. Furthermore, it aims to overcome the limitations shown by the available measurement system related to the transformer secondary current control. The remote monitoring system design was divided in two main parts: the FESA remote interface and the FReSCa server. The FESA class was developed following the FESA standard to give full abstraction of the transformer measurement station to the operator in charge for the test running. For this purpose a FEC was obtained and the FESA class FTS01 was tested and run on it. The FReSCa server layer was designed to manage all the data and the atomic operations, and drive the system along all its possible state. It takes care for the system initialization from the initial device check to the connection establishment. All the request are then processed such as a test cycle request, from the current reference building to the test run, or the status update request. The error handling and the Quench management are also task accomplished by the FReSCa server system. The current control strategy was designed by following the Gain Scheduling logic which suggests changing in the controller parameters according to the working condition changes. The variable chosen for driving the changes in the controller was the reference current I_{Ref} whose domain has been divided in twelve subspaces according to the system behavior. By

identifying each local domain dynamic, a model of the chain composed by the current source and the transformer was synthetized, and the subspaces linked to each other by means of Fuzzy logic. The system input chosen was voltage reference driving the current source while the secondary current was the system output. The scheduling function that allows navigating among the different subdomains was implemented again through the Fuzzy Logic. Thus a PI-FGS system was designed by tuning twelve PI controllers, one associated to each subdomain, and implementing each controller in the consequent of the Sugeno type inference system chosen for the FGS development. The parameters estimated were comparable as magnitude order with the state of the art controller ones even if different. The simulation results of the PI-FGS controller strategy showed a decent attitude of the controller to follow the reference particularly in the ramp phase. Another aspect come out from the simulation result was the excessive overshooting especially in the steep ramp rate references, showing a non-negligible sensitivity of the controller to the ramp-plateau transitions. The last particular taken into account derived from the observation of the absolute error characteristic. Even if it is much less on average with respect to the state of the art, it showed several oscillations afferent to the transition between controllers of adjacent subdomains. The data collected from the PI-FGS simulation where used to train the Neuro-fuzzy network parameters through the back-propagation algorithm starting from the fuzzy inference system defined for the PI-FGS. The network was then implemented in C++ and included in the FReSCa server system in the module for the test run management. The preliminary test confirmed the excessive overshooting trend showed in the simulations, requiring further tuning. To avoid abrupt ramp-plateau transitions, especially at higher ramp rates, and at the same time to not slow down the rising phase a constraint on the transition concerned was introduced. It consists in a minimum transition time of 0.5 s between a ramp and a plateau. This turned in to acceleration (deceleration) between a ramp and a plateau calculated dividing the ramp rate of the ramp interested by the transition time of 0.5 s. This operation is embedded in the reference builder and completely transparent to the OP, furthermore the transition

time chosen is negligible with respect to a normal test session period. The system performances reported in the experimental result improved significantly. The response for the most critical current and ramp rates showed a short overshoot keeping the same capability to follow the reference ramp. In terms of error both absolute and RMS in the compared results showed a reduction up to 52% with respect to the state of the art control strategy. The significant improvement achieved in the ramp phase together with the ability to settle within the given transition time are the main reasons for the strong error decreasing. To perform further step forward two main paths should be followed: a more detailed system model definition and the exploration of control techniques involving the system model. The computation of a more accurate model allows a finer tuning of the controller resulting in a further performance improvement. Furthermore, a more accurate model gives the chance to exploit a model-based control strategy, where a fine system model is the major requirement. Techniques such as Model Predictive Control or Model Reference Adaptive Control can provide significant improvements given a reliable model of the system to be controlled.