COPPE
UFRJ
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia

CONTRIBUTIONS TO THE FPGA AND CPU IMPLEMENTATION OF FREQUENCY-DEPENDENT NETWORK EQUIVALENTS FOR REAL-TIME AND OFFLINE ELECTROMAGNETIC TRANSIENT POWER SYSTEM SIMULATORS

Felipe Novaes Francis Dicler

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Mauricio Aredes

Rio de Janeiro
Dezembro de 2021

CONTRIBUTIONS TO THE FPGA AND CPU IMPLEMENTATION OF FREQUENCY-DEPENDENT NETWORK EQUIVALENTS FOR REAL-TIME AND OFFLINE ELECTROMAGNETIC TRANSIENT POWER SYSTEM SIMULATORS

Felipe Novaes Francis Dicler

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Mauricio Aredes

Aprovada por: Prof. Mauricio Aredes
Prof.ª Maria Cristina Dias Tavares
Prof. Antonio Carlos Siqueira de Lima

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2021

Dicler, Felipe Novaes Francis

Contributions to the FPGA and CPU Implementation of Frequency-Dependent Network Equivalents for Real-Time and Offline Electromagnetic Transient Power System Simulators/Felipe Novaes Francis Dicler. – Rio de Janeiro: UFRJ/COPPE, 2021.

XV, 130 p.: il.; $29,7$cm.

Orientador: Mauricio Aredes

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2021.

Referências Bibliográficas: p. 106 – 113.

1. Power Systems Simulation. 2. Frequency Dependent Network Equivalent. 3. EMTP. 4. Model Reduction. I. Aredes, Mauricio. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

To my beloved family

Acknowledgements

I would like to express my sincerest appreciation to my advisor Prof. Mauricio Aredes for for his continuous encouragement, guidance, and valuable suggestions throughout this work.

I would also like to thank my examining committee members Dr. Maria Cristina Tavares and Dr. Antonio Carlos Siqueira de Lima for carefully reading my thesis and providing insightful questions, comments, and suggestions.

Special thanks should be given to all my colleagues at ONS, the Brazilian Independent System Operator, for supporting and appreciating this research. I am very thankful to Adriano de Andrade, Luiz Filipe, Alex de Castro, João José de Oliveira and Henildo Medeiros de Barros, without the support of whom this thesis would not have been concluded.

I also thanks Prof. Heraldo Almeida for having developed the first version of the SINTEQV tool employed in this research and Prof. Robson Dias and Prof. Guilherme Rolim for their valuable advices since my graduation studies.

I am also grateful to all my fellow colleagues at UFRJ, including Luiz Felipe, Matheus Sotero, Gonçalo Fontenele, Gabriel Antero, Gabriel Provenzano, Gustavo Gontijo, Marcello Neves and Matheus Soares for their sincere friendship and generous help.

I want to express my deepest thanks to my parents for their understanding, encouragement, and unconditional support all the time.

Specially, I want to thank my wife, Annycia and my children. Without their care and love, I wouldn't even have started this work.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CONTRIBUIÇÕES PARA A IMPLEMENTAÇÃO DE EQUIVALENTES DE REDE NA FREQUÊNCIA (FDNE) PARA SIMULADORES DE TRANSITÓRIOS ELETROMAGNÉTICOS OFFLINE E EM TEMPO REAL

Felipe Novaes Francis Dicler

Dezembro/2021

Orientador: Mauricio Aredes

Programa: Engenharia Elétrica

O objetivo deste trabalho foi o desenvolvimento de um algoritmo de solução de espaço de estados customizado aplicado para aumentar a performance computacional de equivalentes elétricos multiporta dependentes da frequência (FDNE) baseados no método de ajuste Vector Fitting. Os fundamentos da interface do equivalente com simuladores de transitórios eletromagnéticos (EMT) foram detalhados, mostrando as diferenças de implementação entre equivalentes ajustados a partir de dados de impedância e admitância e entre os domínios de fases e modos. Um esquema de inicialização foi implementado para execução do FDNE em simuladores que realizam solução fasorial de regime permanente, como o EMTP e ATP. Além dessas aplicações offline, o modelo foi implementado nos simuladores em tempo real RTDS e OPAL-RT e em hardware dedicado FPGA, tendo sido realizada uma análise da viabilidade do FDNE em termos do número de portas e polos para obtenção de passos de integração típicos em cada uma das plataformas mencionadas. A implementação em FPGA foi apresentada em detalhes, examinando a abordagem denominada *High Level Synthesis* e também a interface por fibra ótica com o simulador RTDS. Além disso, um algoritmo foi desenvolvido para a síntese automática de equivalentes FDNE seccionados permitindo sua execução paralelizada no domínio do tempo. Finalmente, um método que desloca o eixo de frequência das respostas em frequências utilizadas como entrada do Vector Fitting foi proposto para eliminar o erro de discretização associado aos métodos de integração trapezoidal e Backward Euler.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

CONTRIBUTIONS TO THE FPGA AND CPU IMPLEMENTATION OF FREQUENCY-DEPENDENT NETWORK EQUIVALENTS FOR REAL-TIME AND OFFLINE ELECTROMAGNETIC TRANSIENT POWER SYSTEM SIMULATORS

Felipe Novaes Francis Dicler

December/2021

Advisor: Mauricio Aredes

Department: Electrical Engineering

The objective of this work was to develop a custom state space solution algorithm applied to increase the computational performance of multiport frequency-dependent network equivalents (FDNE) based on the Vector Fitting method. The fundamentals of the equivalent interface with electromagnetic transient simulators (EMT) were detailed, showing the implementation differences between equivalents fitted from impedance and admittance values and between phase and mode domains. An initialization scheme was implemented for FDNE execution in simulators that perform steady-state phasor solution, such as EMTP and ATP. In addition to these offline applications, the model was implemented in RTDS and OPAL-RT real-time simulators and in dedicated FPGA hardware, with an analysis of the FDNE's feasibility in terms of the number of ports and poles to obtain typical time-steps in each of the mentioned platforms. The FPGA implementation was presented in detail, examining the approach called High Level Synthesis and also the optical fiber interface with the RTDS simulator. Furthermore, an algorithm was developed for the automatic synthesis of sectioned FDNE equivalents, allowing its parallel execution in time domain. Finally, a method that shifts the frequency axis of the frequency responses used as input to the Vector Fitting was proposed to eliminate the discretization error associated with the trapezoidal and Backward Euler integration methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

According to the Oxford English Dictionary, *simulation* is "the technique of imitating the behavior of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus, esp. for the purpose of study or personnel training". Despite being a previously disseminated concept, the word "simulation" with its present meaning was not found in scientific literature until 1926, when it appeared in a Transactions of the American Institute of Electrical Engineers' paper, in the context of miniaturized, scaled-down transmission lines and electrical apparatus, which were later called Transient Network Analyzers (TNA)[1]. These systems, besides simulators, were also "Real-Time" simulators, since being analogs, their outputs were synchronized with the clock time.

Network Analyzers were first applied to experimentally validate transmission line equations. Then, they were used in a wide range of studies such as power systems designs, obtaining short circuit currents, load-flow solutions, stability analysis, switching over-voltage transients, etc. With the emergence of the High Voltage Direct Current (HVDC) transmission, Network Analyzers were extensively applied for testing control hardware, in the configuration called Hardware-in-the-Loop (HIL), shown in Figure 1.1 which is only possible due to their Real-Time nature.

In the context of HIL, the simulation outputs have to be synchronized with the real-time clock in such a way that the physical device under test behaves as if it were in its real-world application, i.e., in the field. HIL is recognized as an intermediary step between non-real-time simulation (referred to in this work as offline simulation) and field operation. Besides other applications, this approach is very suitable for: i) Factory Acceptance Testing, ii) situations in which the real system cannot go out of operation and iii) testing hardware whose algorithms are black-boxes and are not available for digital simulation.

Every HVDC link in Brazil has a HIL replica for evaluating its control and protection algorithms. The Itaipu HVDC transmission system, constructed in the 1980s, has a hybrid HIL system, comprised of a TNA interfaced with a Digital Real-Time Simulator and a replica of its control systems, located in FURNAS, Rio de Janeiro [2, 3]. The other HVDC links, namely Rio Madeira HVDC system and Xingu-Estreito HVDC system, have their HIL replicas located in the ONS - Brazilian Power System Operator, also in Rio de Janeiro [4, 5].

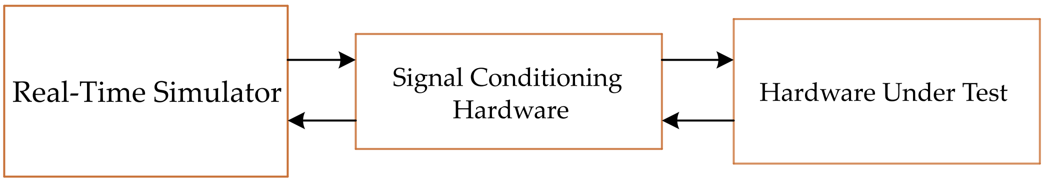


Figure 1.1: Hardware-in-the-Loop configuration.

The major drawbacks of TNAs are their high costs, changes in behavior due to component aging, longer set-up time, complex construction, operation and maintenance, due to which they have been gradually replaced by real-time digital simulators over the last three decades. Hardware and algorithm details for two major commercially available Real-Time Simulators will be given in Chapter 3.

1.2 Statement of the Problem

Power system networks are generally huge structures comprised of thousands of interconnected buses, transmission lines, transformers, generators, etc., which operate together to provide electrical energy to residential and industrial consumers. Due to their electro-mechanical and electromagnetic nature, power devices are continuously exchanging energy. During normal operation, under constant load and topology, the system behavior can be described by voltage and current phasors in the frequency domain [6]. However, under disturbances caused by unpredictable faults, lightning strikes, switching events, etc., the system components are subjected to transients that may propagate excessive currents or voltages in the network and lead to unsafe situations. Consequently, the understanding and mathematical description of the power system transients are fundamental for a wide range of applications in the electric industry, such as the planning, design and validation of control and protection schemes, designing new devices, setting devices ratings, insulation levels and diagnose causes of failures. The modeling and description of such events are the main purposes of the power system transient simulation.

1.2.1 Power Systems Simulation

Finding an analytical solution for the differential-algebraic system of equations that describes a realistic-sized electrical network is an infeasible task, then numerical methods are used instead. The modeling approach depends on the phenomenon under observance.

Power system analysis can be divided into static and dynamic analysis. The static analysis, also known as Load-Flow, assuming the frequency is constant, obtains the system steady-state operating point by calculating the voltage angle and magnitude of each bus in the system for a specified load and generation condition.

Dynamic transients, in turn, cover phenomena in a wide range of frequencies, as depicted in Figure 1.2[7]. They are divided into low-frequency electromechanical transients, whose study is known as Transient Stability Analysis (TSA) and high-frequency electromagnetic transients, referred to as EMT.

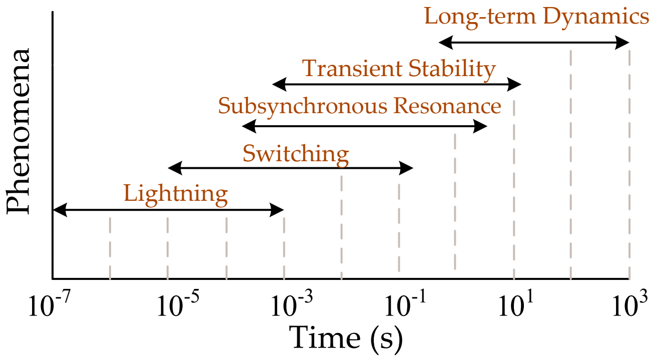


Figure 1.2: Power System Transients Time-Scale.

TSA describes the energy exchange between rotating machines and the electrical network, covering low frequencies in the range of 0 to 60Hz. It is generally used for assessing electromechanical stability by predicting if generators can return to a synchronous operation after being subjected to disturbances such as short circuits, outage of lines, transformers, generators, controls, etc. In such studies, the generator's rotor dynamics are described by the Swing Equation, which relates the net torque (electrical torque minus mechanical torque) with the rotor acceleration. Depending on the rotor's inertia constant, those oscillations can cause large energy flow variations leading to unstable conditions. For such analyzes, as the frequency deviations are small, the network can be modeled by algebraic equations in per-phase representation and phasor notation, with typical time-steps of a few milliseconds. Some commercially available TSA softwares are PSS/E and DigSilent. In the Brazilian electricity sector, the most used TSA softwares are ANATEM (developed by CEPEL, a brazillian electricity research center) and ORGANON, which has real-time capabilities and is used by the Brazilian Power System Operator as a tool for supporting the real-time system operation [8].

Unlike TSA, EMT studies handle electromagnetic interaction between network inductances and capacitances, requiring more accurate electrical models described by differential equations and using smaller time-steps, typically in the range of microseconds. Instantaneous voltage and current waveforms are then obtained. In contrast, low-frequency rotor dynamics are often neglected. Typical EMT phenomena are over-voltage and over-current caused by lightning strikes, energization of transmission lines, shunt capacitor switching, interruption of inductive currents, motor starting, inrush current in transformer, linear resonance in fundamental or harmonic frequencies, series capacitor switching and subsynchronous resonance, load rejection, transient recovery voltage across circuit breakers and very fast transients in gas-insulated bus ducts caused by disconnected operations [9]. With the widespread power electronics utilization, the device-level semiconductor switches modeling has become one of the major applications of EMT simulation tools.

EMT simulation is often performed by the Nodal Analysis or by State-Space methods. In the former, the differential equations of the continuous models are discretized using a numerical method, what leads to a Norton or Thevenin Equivalent circuit representation. Then the Nodal Analysis method (or any of its extended versions such as the Modified Nodal Analysis (MNA)[10] or the Modified-Augmented Nodal Analysis (MANA) [11, 12]) is employed to solve for the node voltages of the network. In the latter, the state-space realization of the system has to be obtained and a numerical method is applied to solve for the state solution.

The most used EMT softwares are ATP, PSCAD-EMTDC, EMTP-RV and SimPowerSystems (MATLAB/Simulink).

1.2.2 Real Time Simulation

A Real-Time simulator is a device that simulates a physical system responding to external stimuli as fast as does the real system. Unlike analog simulators, whose outputs are time continuous, digital simulators compute the outputs of the model only at discrete instants. In order to achieve real-time operation, the execution time cannot exceed the time-step, which is the integration step used when solving the differential equations by means of a numerical method. Figure 1.3 shows a fixed time step simulation which meets this criterion [13].

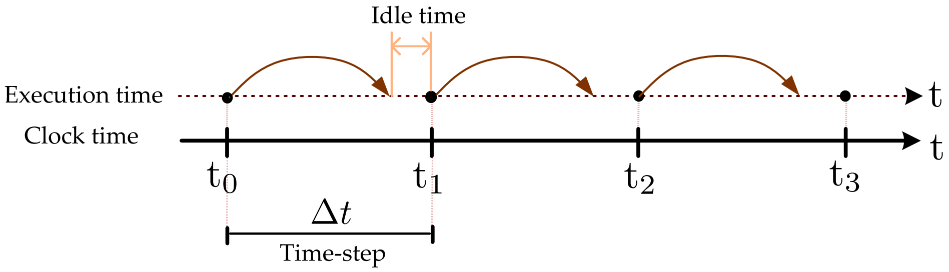


Figure 1.3: Fixed time-step simulation.

Real-time digital simulation first emerged in the late 1950s in the context of defense technology, in applications such as operational flight training[14], space telemetry systems[15] and nuclear reactors[16]. In the power systems industry, some early real-time digital simulation applications are load-flow steady-state estimation [17], TSA for operators training and evaluation of automatic generation control algorithms [18, 19].

In the late 1980s, the first real-time digital EMT simulations were made representing transmission lines with Bergeron's traveling wave model and Marti's frequency-dependent line model in a single DSP [20, 21]. This was a hybrid simulator, in which the digital part replaced the expensive part of the transmission line replica. The reference [22] reported a digital simulator for relay testing, using IBM RISC 6000 workstations and two Texas Instruments DSPs. Also for relay testing, [23] reported a custom hardware implementation based on DSPs which become commercially available under the RTDS trade name two years later. In the early 2000s, simulators based on PC-cluster architecture have been developed using clusters of standard PCs interconnected by high-speed interface cards [24–26].

The popularization of the Voltage Source Converters (VSCs) brought new challenges for real-time simulation, with the increasing demand for simulators capable of handling even smaller time-steps to validate new topologies and test different switching strategies. Venkata Dinavahi proposed a correction procedure for handling switching delays, validated in a DSP and FPGA-based platform [27]. Gustavo Parma implemented an FPGA-based induction machine drive simulator, achieving a time-step of 12.5 ns [28]. Mahmoud Mattar developed an FPGA-based simulator with VSC modeling, rotating machines and frequency dependent network equivalents (FDNEs) [29, 30]. Yuan Chen developed a complete FPGA-based EMT simulator suitable for large systems, including iterative scheme for non-linear models, frequency-dependent transmission line models in both mode and phase domains and universal machine model [31, 32].

A very recent trend is the implementation of geographically distributed real-time simulation, i.e., to execute a synchronized simulation integrating devices located in distant laboratories [33, 34], making use of the Dynamic Phasor concept [35].

Despite the extensive amount of academic research devoted to real-time simulation in recent years, there are still only a few companies dedicated exclusively to design power-systems real-time simulators, among which the following stand out:

- RTDS Technologies Inc., which was the first commercial power-systems real-time digital simulator founded in 1994 after 8 years of development. RTDS has always adopted custom hardware for its solutions. In 2017, they launched the NovaCor chassis, based on IBM's POWER8 RISC processor, containing 10 cores running at 3.5 GHz [36].

- OPAL-RT, founded in 1997, covers a wide range of applications beyond power systems. Unlike RTDS, its solutions are usually based on off-the-shelf hardware devices such as Supermicro Motherboards and Intel Xeon processors. Its simulation tools are often integrated with commercial software like MATLAB/Simulink and the HYPERSIM suite [37, 38].

- Typhoon HIL, founded in 2008 with focus on controller-Hardware-in-the-Loop solutions for power electronics, microgrids, and distribution networks [39].

System partitioning is always necessary when simulating large networks in EMT simulation. This is accomplished by creating sub-systems separated by transmission lines whose propagation delay exceeds the integration time-step. By doing so, the system admittance matrix acquires a block diagonal format. Then each processing hardware handles the calculations related to its respective subsystem.

Despite the extensive research and commercial developments that have been made in the last decades, performing real-time EMT simulations for large networks still poses a challenge. Emerging trends in power systems such as the integration of renewable resources and energy storage systems through power electronic converters bring the need for even reduced time steps. HVDC Multi-Infeed studies require large network detailed representations, covering phenomena in a wide range of frequencies.

1.2.3 Frequency Dependent Network Equivalent

One way to represent large networks while still maintaining sufficient accuracy for real-time simulation is resorting to the use of network equivalents. This approach divides the system into a study zone, where components are modeled with any level of desired accuracy and an external zone, represented by the equivalent, as shown in Fig. 1.4. The conventional power-frequency short circuit equivalent requires a careful examination of the frequency response of the studied area since it only represents the 60 Hz frequency. A useful rule-of-thumb consists in representing at least two transmission lines between the equivalent border and the studied area, as these lines can attenuate the high-frequency transients [40, 41]. But even this rule can not guarantee a good accuracy, as will be shown in Chapter 3. Hence the use of an equivalent that can represent the frequency dependency of the external models becomes a useful solution for reducing the computational burden while still preserving the original frequency response of the studied area.

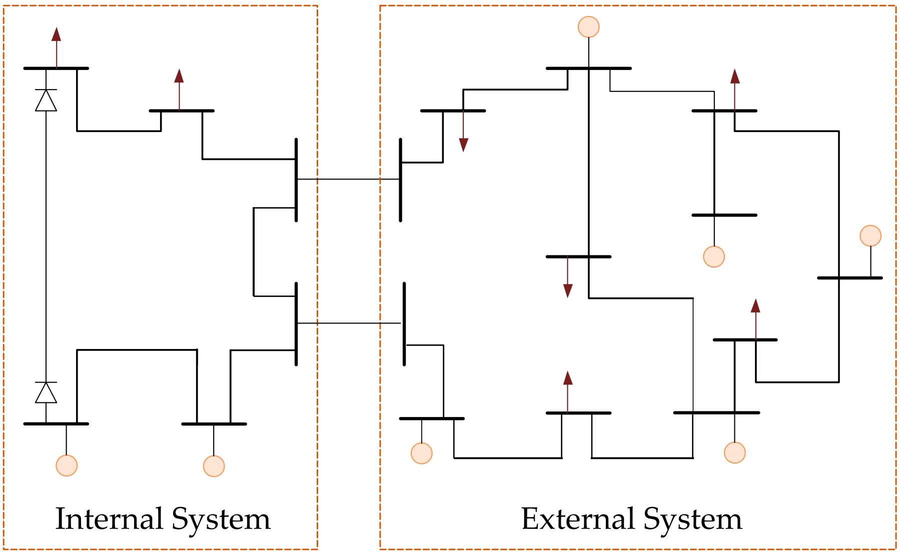


Figure 1.4: Network equivalent concept.

The early attempts to develop frequency dependent network equivalents date back to the late 1960s [42] with the synthesis of passive circuits (RLC) matching the frequency response of the external zone. Then some development has been made regarding the calculation methods of the parameters R, L and C [43]. This approach was desirable because it facilitated the equivalent's integration into the simulator, either a TNA or a digital one, but could not model arbitrary frequency responses and usually required a post-processing optimization in order to improve the fitting accuracy. The fitting of rational functions was then suggested as a more general technique. Gustavsen and Semlyen have developed the method called Vector Fitting [44–46], which has become one of the most successful techniques for creating reduced-order models for linear systems starting from samples of their response and has been used for many applications outside power systems [47]. Available as an open-source Matlab routine, this method is performed calculating a least-squares rational approximation of a vector of frequency domain response using a common set of stable poles and replacing a set of starting poles with an improved set of poles via a scaling procedure. This technique was first applied to transmission line and transformer models and has after been applied to FDNEs. Unlike the equivalent circuit method, the Vector Fitting can provide unstable time-domain responses if the resulted model is not guaranteed to be passive, i.e., it cannot generate energy in order to be stable. Thus, a passivity-enforcement post-processing step is often needed [48]. The research made in this thesis makes use of the Vector Fitting and the Passivity-Enforcement routines available at SINTEF's website [49]. Several alternative formulations and enhancements have been proposed to improve the fitting accuracy and performance, such as Frequency Partitioning[50, 51], Optimal Order Identification [52, 53], Dominant Poles [54, 55], Modal Decomposition[56, 57], Matrix Compacting [58], Multiple Time-Steps [59, 60] and Idempotent Transformation [61]. A complete and exhaustive description of the state-of-art of the VF, its enhancements and applications can be found in [62].

Despite the equivalent concept indeed decreases the computational burden of EMT simulation, the accurate representation of large networks often requires a high number of poles, which can make the traditional FDNE not suitable for CPU-based real-time simulation. Then some efforts have been made to improve the equivalent's efficiency. Mohamed Abdel-Rahman et. al. [63] observed that much of the burden of fitting the external system could be precluded by preserving some borderline transmission line with simplified modeling. The Two-Layer Network Equivalent (TLNE) concept was then proposed[64], in which the external system is split into a surface layer consisting of line models and a deep layer modeled by low-order rational functions. Xin Nie improved the deep layer fitting using the genetic algorithm to find its optimal order and implemented it in real-time within a PC-Cluster architecture [65]. Mahmoud Matar simplified the surface layer transmission line models, representing its characteristic impedance as a constant resistance rather than by a low-order rational function and obtained sufficiently good accuracy [66]. This model was validated in real-time with an FPGA platform.

In the last years, some efforts have been made to increase the FDNE time-domain performance by optimizing its state-space solver. References [67] and [68] greatly improved the FDNE performance by developing an accelerated state-space solver considering the sparsity of the state matrices. This solver was implemented in both EMTP and HYPERSIM tools. Reference [69] suggested storing the state matrix $\mathbf{A}$ as a vector, since it has a diagonal structure. This approach avoids unnecessary null calculations in the matrix-vector products when updating the state variables.

Although the TLNE and its further enhancements indeed increase FDNE efficiency, they bring additional modeling difficulties by requiring the user to choose which transmission lines to keep in the surface layer. As the objective of this work is to obtain a network equivalent model as general as possible and intended for the ordinary user (who is not necessarily familiar with the FDNE development), the traditional FDNE formulation was adopted. It was implemented as an optimized solver that, instead of handling the sparsity as done in previous works, converts all input matrices into vectors, storing only the non-null terms, and takes advantage of the fixed topology of the state matrices as delivered by the VF to construct a dedicated solver applicable only for this specific topology. This approach is an extension of what is suggested in [69], where only the $\mathbf{A}$ matrix is vectorized. Furthermore, the implemented solver also works in the modal domain, what increases its numerical computational performance.

This new approach represents a significant gain in the computational performance of the FDNE model when interfaced with an EMT solver. This enhancement makes it possible to use the FDNE component with a greater number of terminals and poles, as well as allowing its application in real-time simulation, where performance is crucial. To the best of the author's knowledge, these improvements resulted in the highest time-domain performance of the FDNE so far reported for both offline and real-time EMT simulation.

1.2.4 SINTEQV Software

The algorithms developed in this work were embedded in a tool previously developed in ONS called SINTEQV, which is a C implementation of the open-source Matlab Vector Fitting routine. This program is able to automatically generate FDNE models for EMT softwares such as PSCAD-EMTDC, ATP, EMTP-RV, RSCAD/RTDS and Hypersim. Figure 1.5 presents an overview of the SINTEQV and its input/output data interfaces. The FDNE is synthesized based on the following input data:

- Frequency responses of the elements of the nodal impedance (or admittance) matrix seen by the boundary bars, obtained by reading the frequency scan from the ATP or EMTP-RV tools.

- Phasors referring to the power flow case in the boundary bars, obtained from the automatic reading of the output report from ANAREDE power-flow tool.

This is an optional feature for automatically generating the active sources of the FDNE.

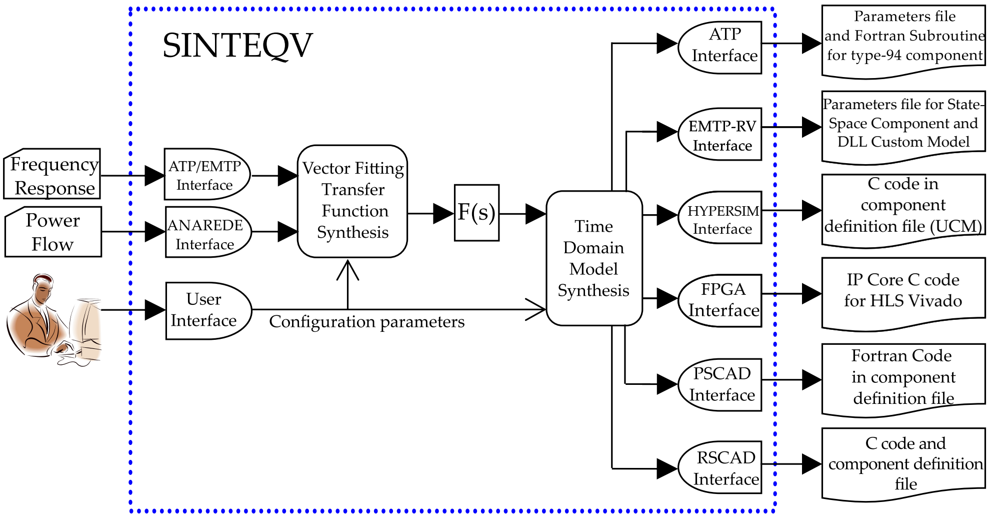
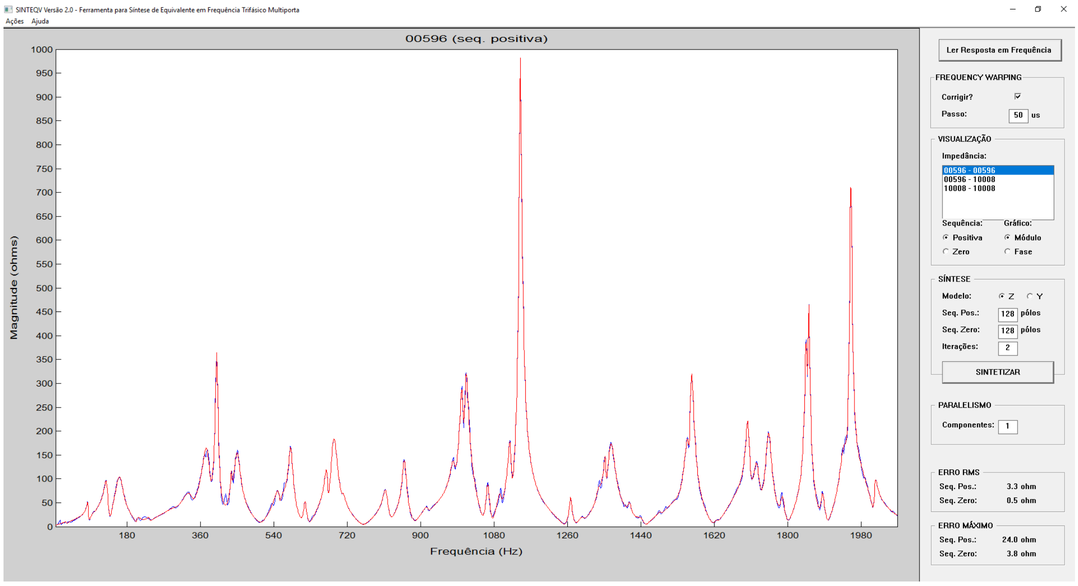

SINTEQV
Frequency Response
Power Flow
ATP/EMTP Interface
ANAREDE Interface
User Interface
Vector Fitting Transfer Function Synthesis
F(s)
Configuration parameters
Time Domain Model Synthesis
ATP Interface
EMTP-RV Interface
HYPERSIM Interface
FPGA Interface
PSCAD Interface
RSCAD Interface
Parameters file and Fortran Subroutine for type-94 component
Parameters file for State-Space Component and DLL Custom Model
C code in component definition file (UCM)
IP Core C code for HLS Vivado
Fortran Code in component definition file
C code and component definition file

Figure 1.5: SINTEQV's functional block diagram.

The SINTEQV is developed in the Microsoft Visual Studio IDE, using the Intel MKL (Math Kernel Library) CLAPACK and BLAS libraries and OpenMP for full parallelization. Its Graphical User Interface, shown in Figure 1.6, is based on the Windows API. Due to its C-based high-performance implementation of the Vector-Fitting routine, the SINTEQV is used for all FDNE synthesis performed in this work.



FREQUENCY WARPING

Corrigir? ☑

Passo: 50 us

VISUALIZAÇÃO

Impedância:

00596 - 00596
00596 - 10008
10008 - 10008

Sequência: ● Positiva ○ Zero

Gráfico: ● Módulo ○ Fase

SÍNTESE

Modelo: ● Z ○ Y

Seq. Pos.: 128 pólos

Seq. Zero: 128 pólos

Iterações: 2

PARALELISMO

Componentes: 1

ERRO RMS

Seq. Pos.: 3.3 ohm

Seq. Zero: 0.5 ohm

ERRO MÁXIMO

Seq. Pos.: 24.0 ohm

Seq. Zero: 3.8 ohm

Figure 1.6: GUI of the SINTEQV tool.

1.3 Research Contributions

The main contributions of this thesis are:

- Development of an automatically-generated fully parallelizable custom FDNE state-space solver suitable for both offline and real-time EMT simulation in both phase and modal domains, considering rational approximation for either impedance or admittance values.
- Assessment of its performance for the EMTP® offline tool, as well as the RTDS® and OPAL-RT® real-time simulators.
- The development of an FPGA model for the FDNE, with a high number of poles and ports and interfacing it with the RTDS Novacor platform.
- The proposal of a warping-error correction algorithm applied to the input frequency responses of the Vector Fitting method.
- An assessment of the performance of the FDNE component for challenging cases in both offline and real-time.
- A comparison between the RTDS and the OPAL-RT platforms regarding user custom models.

1.4 Thesis Outline

This thesis contains six chapters including this introduction. The next chapters are organized as follows:

Chapter 2: FDNE

This chapter presents the frequency response identification technique for the phase and the modal domains, the Vector Fitting algorithm, the discrete-time FDNE model synthesis and its high efficient time-domain integration algorithm for both impedance and admittance values, the initalization from steady state scheme and the post-processing passivity enforcement routine.

Chapter 3: EMTP Implementation This chapter describes the implementation of the custom FDNE component as a DLL file and its integration to the kernel solver of the EMTP software. A study case of transformer energization is demonstrated for single and multi-port reductions and a performance evaluation is made.

Chapter 4: FDNE Implementation in RTDS and OPAL-RT simulators

This chapter introduces the real-time simulation, presenting the equivalent implementation on both RTDS and OPAL-RT environments, addressing issues concerning each platform user custom model details. This chapter also presents the sectioning FDNE method and the frequency warping correction algorithm.

Chapter 5: FPGA Design Architecture for the FDNE Model

This chapter introduces some basics FPGA concepts, the High Level Synthesis approach, the FPGA-RTDS interface, the hardware description language architecture of the developed FDNE model and its results.

Chapter 6: Conclusions and Future Work

This chapter finalizes the work, highlighting the conclusions, contributions and providing some guidelines for future research.

Chapter 2

Frequency Dependent Network Equivalent

This chapter provides the fundamental concepts regarding FDNEs, including the network's frequency response identification, the Vector Fitting (VF) routine, the modal decomposition in sequence networks, the discrete-time model synthesis, the post-processing passivity enforcement routine and the interfacing approaches to time-domain EMT simulation. Developments will be described in impedance parameters for convenience.

2.1 Frequency Response Identification

The FDNE synthesis begins by determining the frequency response of the nodal impedance matrix Z_{bus} as seen from the external network terminals. Each port is associated with a self impedance and $(p-1)$ mutual impedances, being p the number of three-phase ports of the equivalent. Equation (2.1) presents the Network Equation, in which the Z_{bus} matrix relates the voltages and currents at the interface buses. A graphical representation of the p-port network equivalent can be seen in Figure 2.1.

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_p \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1p} \\ Z_{21} & Z_{22} & \cdots & Z_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{p1} & Z_{p2} & \cdots & Z_{pp} \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_p \end{bmatrix} \tag{2.1}$$

It should be noted that the impedances Z_{ij} between ports i and j do not have the physical meaning of an impedance measurement between those nodes, they are instead the components of Z_{bus}, which is the inverse of Y_{bus}, whose components indeed are physical admittances measurements between those nodes. In short,

$Z_{ij} \neq 1/Y_{ij}$.

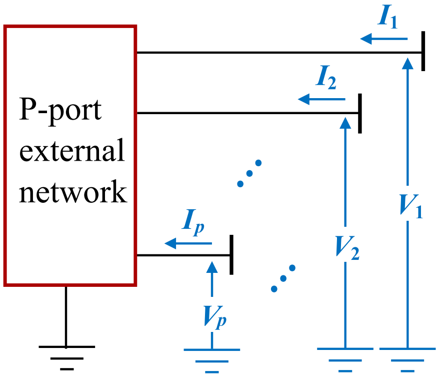


Figure 2.1: p-port equivalent circuit.

In Equation (2.1), voltage, current and impedance are three-phase quantities. The frequency response identification can be made either in phase or modal domains.

$$V_i = \begin{bmatrix} v_i^A \\ v_i^B \\ v_i^C \end{bmatrix}, \quad I_i = \begin{bmatrix} i_i^A \\ i_i^B \\ i_i^C \end{bmatrix}, \quad Z_{ij} = \begin{bmatrix} Z_{ij}^{AA} & Z_{ij}^{AB} & Z_{ij}^{AC} \\ Z_{ij}^{BA} & Z_{ij}^{BB} & Z_{ij}^{BC} \\ Z_{ij}^{CA} & Z_{ij}^{CB} & Z_{ij}^{CC} \end{bmatrix}, \quad i,j = 1,2,\cdots,p \tag{2.2}$$

2.1.1 Phase-Domain

For the phase-domain frequency response identification, a single-phase unitary current injection is successively applied in each terminal of the equivalent for each frequency in the fitting range, as illustrated in Fig. 2.2. By doing so, the voltage measured at each terminal is equal its impedance, since the current is unitary. As the Z_{BUS} matrix is symmetric, the phasor solution of each injection is used to obtain the self and mutual impedances associated with the terminal receiving that injection, i.e., the $k-th$ row and column of the matrix, being $k = 1, 2, ..., 3p$. Setting voltage sources and measuring currents would give the same result. This procedure works under the assumption that there is no power sources inside the network, so any voltage sources should be replaced by short circuits and the current sources must be opened.

In EMT tools that perform phasor solution, like ATP and EMTP, this frequency scan is automatically made, but as most real-time simulators do not perform steady-state solution, a different approach is necessary. Instead of applying one unitary injection for each frequency, a summation of all currents is obtained and inject once for each terminal. Then, the Fast Fourier Transform (FFT) is applied to the voltage waveforms, what yields the frequency response. The major drawbacks of this method are the error introduced by the time-domain discretization and the long time it consumes. This approach will be explained in Section 4.1.

Usually, the network to be reduced is modeled in offline tools, then the straightforward phasor frequency scan method can be applied. But for evaluating the FDNE model in real-time simulators, the FFT approach is used.

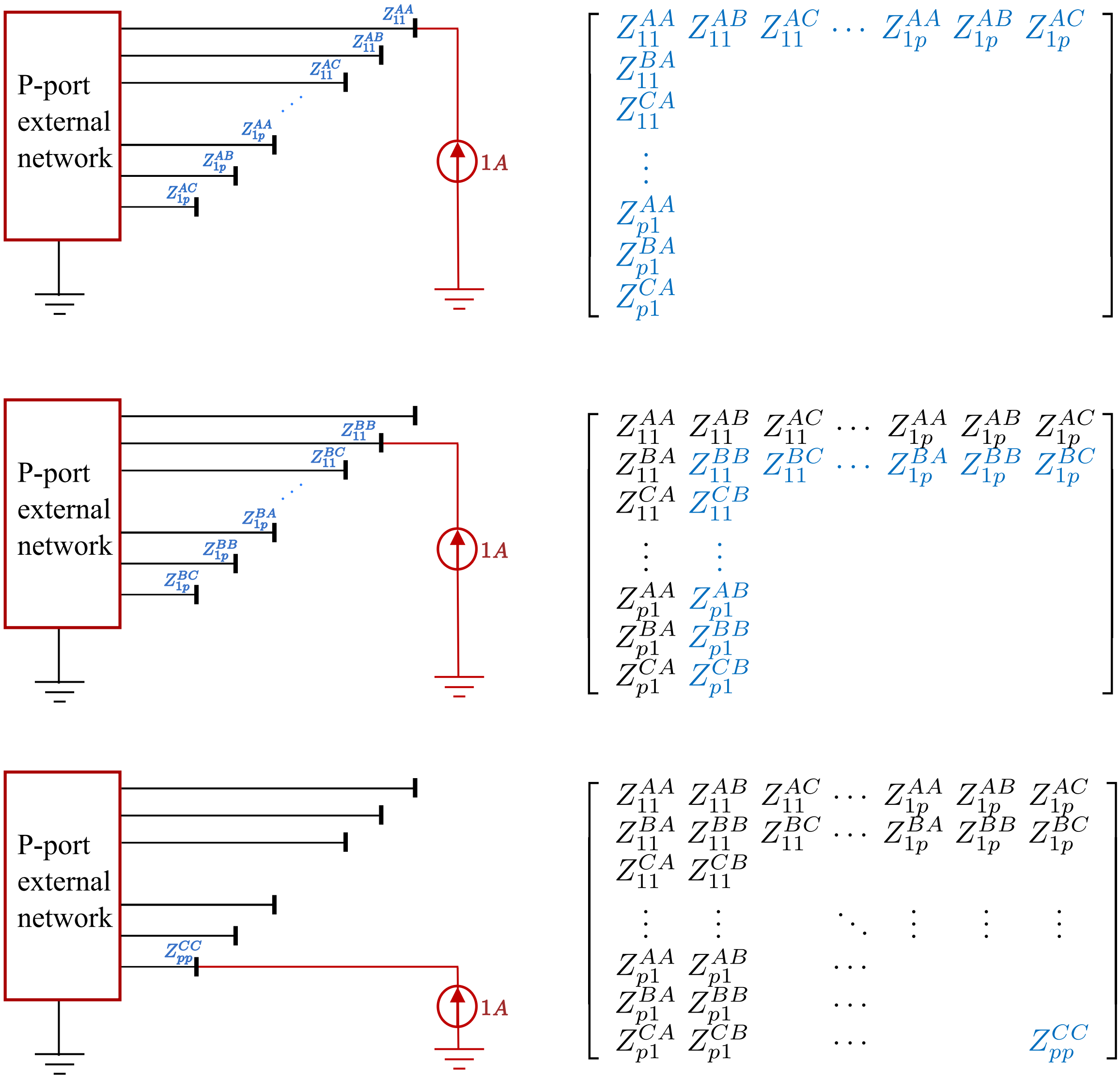


Figure 2.2: Obtaining Z_{BUS} performing unitary current injections in every bus for every required frequency.

2.1.2 Modal-Domain

For balanced networks, a great performance increase can be obtained using the modal decomposition [70], which is detailed in Appendix B. Applying the modal decomposition (B.8) to 2.1, the three-phase network is decomposed into three completely independent single-phase networks corresponding to two non-homopolar modes and one homopolar mode.

$$\begin{bmatrix} v_i^A \\ v_i^B \\ v_i^C \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} v_i^0 \\ v_i^1 \\ v_i^2 \end{bmatrix}, \quad i = 1, 2, \cdots, p \tag{2.3}$$

$$\begin{bmatrix} i_i^0 \\ i_i^1 \\ i_i^2 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} i_i^A \\ i_i^B \\ i_i^C \end{bmatrix}, \quad i = 1, 2, \cdots, p \tag{2.4}$$

$$\begin{bmatrix} v_1^m \\ v_2^m \\ \vdots \\ v_p^m \end{bmatrix} = \begin{bmatrix} Z_{11}^m & Z_{12}^m & \cdots & Z_{1p}^m \\ Z_{21}^m & Z_{22}^m & \cdots & Z_{2p}^m \\ \vdots & \vdots & \ddots & \vdots \\ Z_{p1}^m & Z_{p2}^m & \cdots & Z_{pp}^m \end{bmatrix} \cdot \begin{bmatrix} i_1^m \\ i_2^m \\ \vdots \\ i_p^m \end{bmatrix}, \quad m = 0, 1, 2 \tag{2.5}$$

where p is the number of three-phase ports and m is the sequence.

For three-phase balanced networks composed exclusively of passive models, the impedance matrix is symmetric and the same for both non-homopolar modes, i.e:

$$Z_{ij}^m = Z_{ji}^m, \quad i, j = 1, 2, \cdots, p, \quad m = 0, 1, 2$$

$$Z_{ij}^1 = Z_{ij}^2, \quad i, j = 1, 2, \cdots, p$$

Hence, only the upper triangular part of impedance matrices in the modal domain must be obtained, which gives a total of $p(p+1)$ frequency responses to be fitted.

The same current injection method shown before can be used here, but injecting three-phase currents of positive and zero sequences, instead of single-phase quantities, as depicted in Figure 2.3.

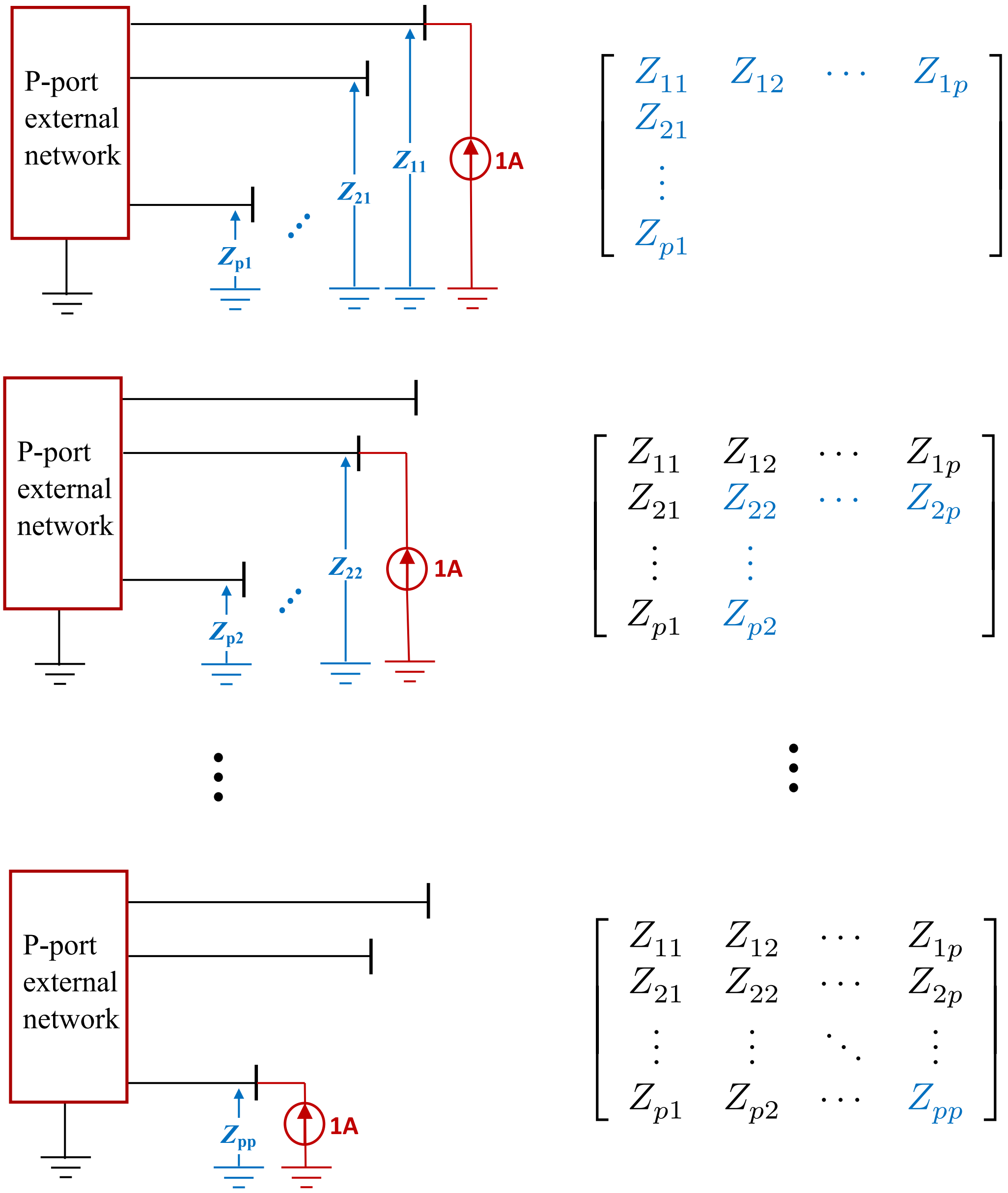


Figure 2.3: Obtaining Z_{BUS} performing unitary current injections in every bus for every required frequency.

2.2 Vector Fitting

The objective of the VF algorithm in the FDNE synthesis is to obtain rational functions in the pole-residue form that approximate the input frequency responses, i.e, the system's admittance (or impedance) matrices. Being k the number of frequency samples and p the number of ports, the inputs of the VF routine obtained from the frequency scan are

$$Z_{\mathrm{ij}}\left(j\omega_f\right) = R_{\mathrm{ij}\,f} + jX_{\mathrm{ij}\,f}, \quad i, j = 1, 2, \cdots, p; \quad f = 1, 2, \cdots, k \tag{2.6}$$

or

$$\mathbf{Z}(s) = \begin{bmatrix} Z_{11}(s) & Z_{12}(s) & \cdots & Z_{1p}(s) \\ Z_{21}(s) & Z_{22}(s) & \cdots & Z_{2p}(s) \\ \vdots & \vdots & \ddots & \vdots \\ Z_{p1}(s) & Z_{p2}(s) & \cdots & Z_{pp}(s) \end{bmatrix} \tag{2.7}$$

The synthesis algorithm finds rational functions with n poles, in the form of (2.8), which fit the set of input frequency responses minimizing the mean squared error, employing the same set of poles for all components of the admittance matrix.

$$\hat{Z}_{ij}(s) = \sum_{k=1}^{n} \frac{c_{ijk}}{s - a_k} + d_{ij}, \quad i, j = 1, 2, \cdots, p \tag{2.8}$$

or equivalently

$$\hat{\mathbf{Z}} = \sum_{k=1}^{n} \frac{\mathbf{C}_k}{s - a_k} + \mathbf{D} = \begin{bmatrix} \sum_{k=1}^{n} \frac{c_{11k}}{s - a_k} & \cdots & \sum_{k=1}^{n} \frac{c_{1pk}}{s - a_k} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} \frac{c_{p1k}}{s - a_k} & \cdots & \sum_{k=1}^{n} \frac{c_{ppk}}{s - a_k} \end{bmatrix} + \begin{bmatrix} d_{11} & \cdots & d_{1p} \\ \vdots & \ddots & \vdots \\ d_{p1} & \cdots & d_{pp} \end{bmatrix}$$

where coefficients a_k and c_{ijk} are the poles and residues, respectively, and coefficients d_{ij} are the constant terms responsible for the asymptotic response of the system. The pole-residue model in (2.8) can be converted to its state-space formulation (2.9).

$$\begin{cases} \dot{X}(s) = \mathbf{A}X(s) + \mathbf{B}U(s) \\ Y(s) = \mathbf{C}X(s) + \mathbf{D}U(s) \end{cases} \tag{2.9}$$

where X is the state vector, U the input vector, Y the output vector and **A**, **B**, **C** and **D** are the system matrices.

The poles are stored in the **A** matrix, whose order is np, being n the number of poles and p the order of the **Z** matrix to be fitted, i.e, it is the number of terminals in the phase domain or ports in the modal domain. The real poles are placed in the main diagonal. The complex poles appear as 2x2 blocks, where α represents the real part, and β the complex part, as shown in (2.10). The blocks A_1 to A_p are equal since the same set of poles is used to fit all frequency responses (this was suggested in the closure to [71], in order to increase the time-domain computational performance of the model). The B matrix is $[np \text{ x } p]$ normalized to contain the number 1 for a real pole, 2 for the real part of a complex pole and 0 for its imaginary part. The C matrix is $[p \text{ x } np]$, carrying the residues of the rational transfer function, which can be either real or complex. Finally, the D matrix is [p x p] having only real components.

The state-space equation (2.9) can be converted to its transfer function representation (2.10).

$$Z(s) = C(sI - A)^{-1}B + D \tag{2.10}$$

where

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_p \end{bmatrix} \quad , \quad B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_p \end{bmatrix}$$

$$A_i = \begin{bmatrix} a_1 & & & & & & & & & & & \\ & a_2 & & & & & & & & & & \\ & & \ddots & & & & & & & & & \\ & & & a_{n_r} & & & & & & & & \\ & & & & \alpha_1 & \beta_1 & & & & & & \\ & & & & -\beta_1 & \alpha_1 & & & & & & \\ & & & & & & \alpha_2 & \beta_2 & & & & \\ & & & & & & -\beta_2 & \alpha_2 & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & & \ddots & & \\ & & & & & & & & & & \alpha_{n_c} & \beta_{n_c} \\ & & & & & & & & & & -\beta_{n_c} & \alpha_{n_c} \end{bmatrix} \quad , \quad B_i = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 2 \\ 0 \\ 2 \\ 0 \\ \vdots \\ \vdots \\ 2 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1p} \\ C_{21} & C_{22} & \cdots & C_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ C_{p1} & C_{p2} & \cdots & C_{pp} \end{bmatrix} \quad ; \quad D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1p} \\ d_{21} & d_{22} & \cdots & d_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ d_{p1} & d_{p2} & \cdots & d_{pp} \end{bmatrix}$$

$$C_{ij} = \begin{bmatrix} c_{ij1} & c_{ij2} & \cdots & c_{ijn} \end{bmatrix} \quad ; \quad i, j = 1, 2, \cdots, p$$

As an example to better visualize the topology of the state-space as obtained from the VF routine, Figure 2.4 presents the realization resulted from a 2-port fit in the modal domain, with two real and four complex poles. Since this system results from an impedance fit, the inputs are currents and the outputs are voltages ($Z(s) = \frac{V(s)}{I(s)}$). The complete FDNE model of this hypothetical case would have three state-space representations (one for each sequence), as the impedance matrices are in the modal domain.

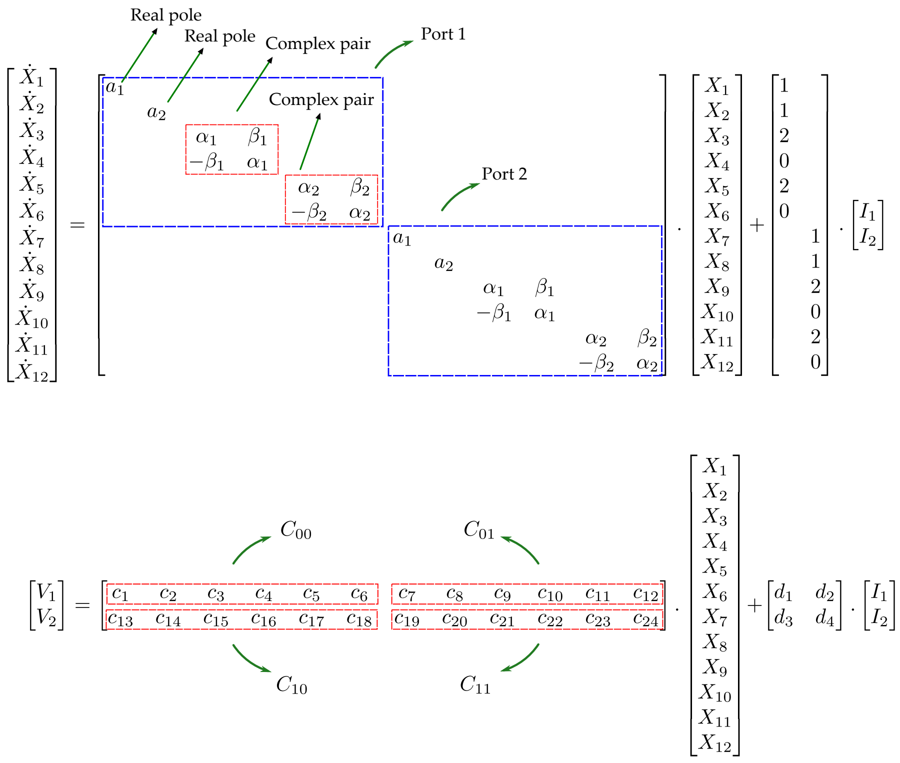


Figure 2.4: State-space system of a multi-port frequency-dependent Thevenin Equivalent.

2.3 Discrete-Time Model Synthesis

Applying the trapezoidal integration method to (2.9) yields the associated discrete state-space system represented in (2.11a) to (2.11c). The complete development is described in Appendix D.

$$X(k) = \hat{A}X(k-1) + \hat{B}U(k-1) + \hat{B}U(k) \tag{2.11a}$$

$$Y_H(k) = -Y_H(k-1) + \hat{C}X(k-1) \tag{2.11b}$$

$$Y(k) = Y_H(k) + \hat{D}U(k) \tag{2.11c}$$

where

$$\hat{A} = \left(I - \frac{\Delta t}{2}A\right)^{-1}\left(I + \frac{\Delta t}{2}A\right) \tag{2.12a}$$

$$\hat{B} = \left(I - \frac{\Delta t}{2}A\right)^{-1}\frac{\Delta t}{2}B \tag{2.12b}$$

$$\hat{C} = C\hat{A} + C \tag{2.12c}$$

$$\hat{D} = C\hat{B} + D \tag{2.12d}$$

Equation $(2.11\mathrm{c})$ can be interpreted as a companion circuit, where the output Y depends on a historic value Y_H from the previous step plus the input scaled by the factor $\hat{D}$, which is a resistance matrix for an impedance fit or a conductance matrix for an admittance one.

2.3.1 Admittance Fitting

For the sake of illustration, for the admittance adjustment, Figure 2.5 represents $(2.11\mathrm{c})$ as a two port Norton Equivalent, where the input $U(k)$ is the vector $[v_1\ v_2]^T$, the historic term $Y_H(k)$ is the vector $[i_{H_1}\ i_{H_2}]^T$ and $\hat{D}$ is the conductance matrix related to the interface nodes 1 and 2.

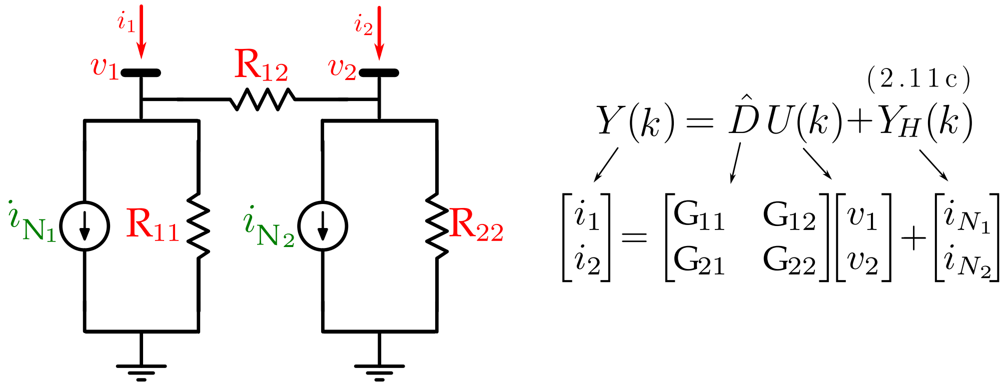

i_1
i_2
v_1
v_2
R_{12}
i_{N_1}
R_{11}
i_{N_2}
R_{22}
(2.11c)
$Y(k) = \hat{D}\,U(k) + Y_H(k)$
$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} i_{N_1} \\ i_{N_2} \end{bmatrix}$

Figure 2.5: Multi-Port Norton Equivalent.

2.3.2 Impedance Fitting

For the case of an impedance adjustment, Figure 2.6 represents $(2.11\mathrm{c})$ as a two port Thevenin Equivalent, where the input $U(k)$ is the vector of nodal current injections $[i_1\ i_2]^T$, the historic term $Y_H(k)$ is the vector of Thevenin Sources $[v_{th_1}\ v_{th_2}]^T$ and $\hat{D}$ is the resistance matrix related to the interface nodes 1 and 2. The resistances in the matrix **R** are primed to emphasize that they do not correspond directly to the circuit values. For example, $R'_{11} = G_{11}^{-1} = (\frac{1}{R_{11}} + \frac{1}{R_{12}})^{-1}$.

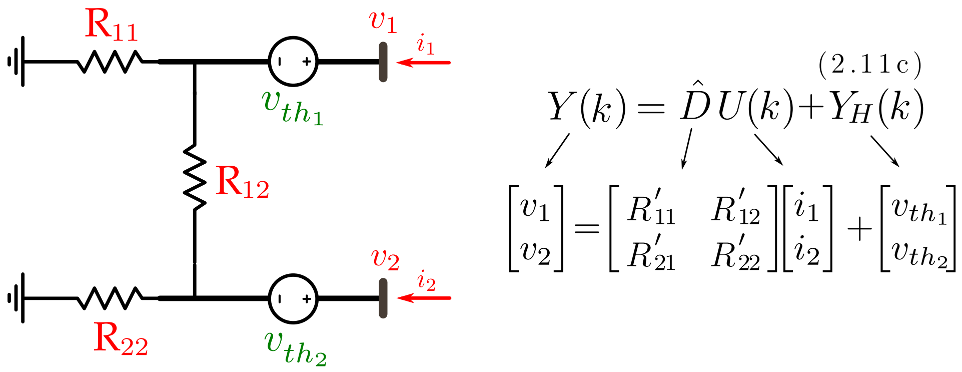

R11
v1
i1
vth1
R12
v2
i2
R22
vth2
(2.11c)
$Y(k) = \hat{D}\,U(k) + Y_H(k)$
$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} R'_{11} & R'_{12} \\ R'_{21} & R'_{22} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} + \begin{bmatrix} v_{th_1} \\ v_{th_2} \end{bmatrix}$

Figure 2.6: Multi-Port Thevenin Equivalent.

The Multi-Port Thevenin Equivalent can not be directly interfaced to the time-domain solver, since the integration of custom components in EMT simulation is usually done by Norton Equivalents. This is due the fact that the Nodal Analyses performed by most simulators calculate nodal voltages and expect to receive current injections to form the global vector of currents, which multiplied by the inverse of the conductance matrix yields the global vector of nodal voltages for the next time-step. Also, the conductance matrix related to the interface nodes has to be incorporated into the global conductance matrix. But when performing an impedance fitting, the historic term Y_H and the factor $\hat{D}$ have voltage and resistance quantities, respectively. Therefore a Thevenin to Norton transformation, as shown in Figure 2.7, is required to i) obtain the conductance matrix related with the terminal nodes before the simulation time-loop begins; ii) translate the terminal nodal voltages into current injections at every time-step (state-space input) and iii) translate the Thevenin sources into Norton sources to be injected at the terminal nodes at every time-step.

It is worth noting that the Norton Sources after the transformation are entering the interface nodes, instead of leaving them (as in the Norton Equivalent of the admittance fitting). So one should be careful about the convention adopted by the simulator, i.e, if a current injection leaving a node is positive or negative. All simulators for which FDNE models were developed in this dissertation (EMTP, RSCAD and HYPERSIM) adopt the convention of positive signal for the current leaving a node, so the sign of Norton Sources had to be changed in the impedance fitting after the transformation from Thevenin Sources.

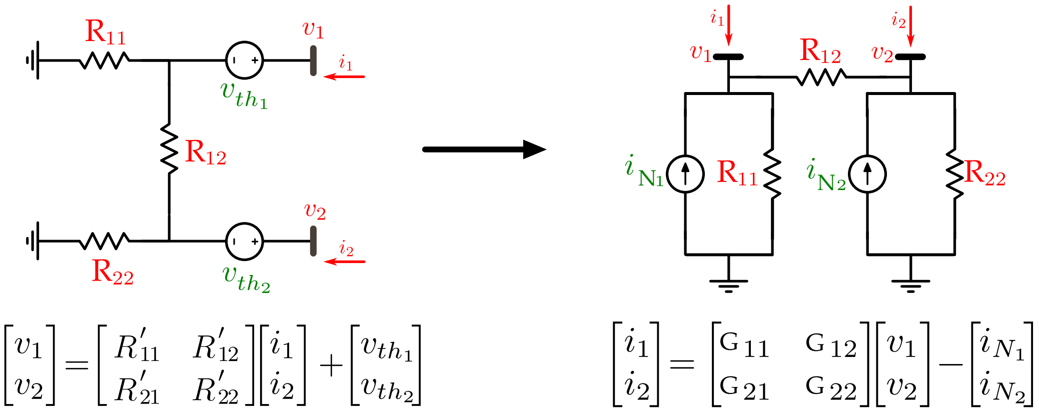

R11
v1
i1
vth1
R12
v2
i2
R22
vth2
i1
i2
v1
R12
v2
iN1
R11
iN2
R22
$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} R'_{11} & R'_{12} \\ R'_{21} & R'_{22} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} + \begin{bmatrix} v_{th_1} \\ v_{th_2} \end{bmatrix}$
$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} - \begin{bmatrix} i_{N_1} \\ i_{N_2} \end{bmatrix}$

Figure 2.7: Multiport Thevenin to Norton transformation.

The Norton Current sources are given by (2.13).

$$\begin{bmatrix} i_{N_1} \\ i_{N_2} \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} v_{th_1} \\ v_{th_2} \end{bmatrix} \tag{2.13}$$

and the conductance matrix is the inverse of the resistance matrix

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} = \begin{bmatrix} R'_{11} & R'_{12} \\ R'_{21} & R'_{22} \end{bmatrix}^{-1}. \tag{2.14}$$

In short, for the admittance-based fitting, the historic term Y_H is a current source vector and $\hat{D}$ is a conductance matrix, as expected by the network solver, so the integration is straightforward. On the other hand, if the fitting is made in impedance quantities, Y_H is a voltage source vector and $\hat{D}$ is a resistance matrix, as a Thevenin Equivalent, so these terms must be converted with (2.13) and (2.14) before being integrated to the network solver.

2.4 Time-Domain Interface Algorithm

The overall FDNE algorithm to be implemented in any simulator's user custom model can be divided into two sections:

1. Part 1: Discrete model parameters and Norton Conductance Matrix calculation. This step precedes the time-domain simulation.
2. Part 2: Historic current sources that are calculated at every time-step. This step solves the state-space discrete system and is performed within the time-domain simulation.

Both steps are summarized and illustrated in Figures 2.8 and 2.9. The algorithm may vary according to the following: 1) the fitting domain, i.e, whether in phase or modal coordinates; 2) the type of the fitted matrix, i.e, admittance or impedance and 3) the initial conditions, i.e, if steady-state or zero initialization. The algorithm in modal coordinates is shown here, highlighting the different procedures for both impedance and admittance fits. One should note that the negative sequence parameters are not calculated, since the supposition of balanced networks for which $Z^1 = Z^2$ was made. Consequently, the negative-sequence state-space is solved employing the positive-sequence parameters. Also, in the following algorithm, the discretized state-space matrices are denoted by a hat, as in (2.11a) to (2.11c). The initialization from steady-state will be covered in the section 2.6.

Part 1: Discrete Model parameters and Norton Conductance Matrix calculation.

1. Input data:
   - Number of ports: p
   - Number of poles (both modes): n_0 and n_1
   - Number of real poles (both modes): n_{r0} and n_{r1}
   - Matrices A_0 and A_1 (block-diagonal $[np \text{ x } np]$)
   - Matrices C_0 and C_1 (full $[p \text{ x } np]$)
   - Matrices D_0 and D_1 (full $[p \text{ x } p]$)
   - Integration time-step: Δt
2. Matrix $\hat{A}_0$ and $\hat{A}_1$ (Eq. 2.12a)
   - $\hat{A}_0 = \left(I - \frac{\Delta t}{2} A_0\right)^{-1} \left(I + \frac{\Delta t}{2} A_0\right)$
   - $\hat{A}_1 = \left(I - \frac{\Delta t}{2} A_1\right)^{-1} \left(I + \frac{\Delta t}{2} A_1\right)$
3. Matrix $\hat{B}^0$ and $\hat{B}^1$ (Eq. 2.12b)
   - $\hat{B}_0 = \left(I - \frac{\Delta t}{2} A_0\right)^{-1} \frac{\Delta t}{2} B_0$
   - $\hat{B}_1 = \left(I - \frac{\Delta t}{2} A_1\right)^{-1} \frac{\Delta t}{2} B_1$
4. Matrix $\hat{C}^0$ and $\hat{C}^1$ (Eq. 2.12c)
   - $\hat{C}_0 = C_0 \hat{A}_0 + C_0$
   - $\hat{C}_1 = C_1 \hat{A}_1 + C_1$
5. Matrix $\hat{D}^0$ and $\hat{D}^1$ (Eq. 2.12d)

- $\hat{D}_0 = C_0\hat{B}_0 + D_0$
- $\hat{D}_1 = C_1\hat{B}_1 + D_1$

6. Nodal Conductance Matrix in modal components (Eq. 2.14. If the fitting is in admittance values, $\hat{D}$ is already the Nodal Conductance Matrix)

   - $G_0 = (\hat{D}_0)^{-1}$
   - $G_1 = (\hat{D}_1)^{-1}$

7. Calculate self and mutual conductance matrix:

   - $G^{\text{s}} = \frac{1}{3}(G_0 + 2G_1)$
   - $G^{\text{m}} = \frac{1}{3}(G_0 - G_1)$

8. Construct the Nodal Conductance Matrix in phase components:

   - Firstly, each [3x3] sub-matrix is assembled.

$$G_{ij} = \begin{bmatrix} G^{\text{s}}_{ij} & G^{\text{m}}_{ij} & G^{\text{m}}_{ij} \\ G^{\text{m}}_{ij} & G^{\text{s}}_{ij} & G^{\text{m}}_{ij} \\ G^{\text{m}}_{ij} & G^{\text{m}}_{ij} & G^{\text{s}}_{ij} \end{bmatrix} \quad ; \quad i,j = 1,2,\cdots p$$

   - The full matrix [3px3p] is then constructed.

$$G_{\text{norton}} = \begin{bmatrix} G_{11} & \cdots & G_{1p} \\ \vdots & \ddots & \vdots \\ G_{p1} & \cdots & G_{pp} \end{bmatrix}$$

Part 2: Norton Current Sources calculated at every time-step.

1. Read the voltage node vector V_{node} given by the network solver ([3p] entries in phase coordinates).

2. Calculate the current node vector I_{node} entering the Thevenin Equivalent (If in admittance fiting, this is not necessary, since in this case V_{node} is already the state-space input.):

   - v_{th} is Y_H from the last time-step (Eq. 2.11c).
   - $I_{\text{norton}} = G_{\text{norton}}v_{\text{th}}$ (Eq. 2.13).
   - $I_{\text{node}} = I_{\text{norton}} + G_{\text{norton}}V_{\text{node}}$ (Fig. 2.7 - Norton Sources' sign is reversed, as explained in 2.3.2).

3. Convert state-space inputs (voltage or currents) from phase $(U^A, U^B\text{e}U^C)$ to modal components (U^0, U^1 e U^2), for each port p (Eq. B.7).

- $$\begin{bmatrix} U_i^0 \\ U_i^1 \\ U_i^2 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix}\begin{bmatrix} U_i^A \\ U_i^B \\ U_i^C \end{bmatrix}, \quad i = 1, 2, \cdots, p$$

4. Update state vectors X_0, X_1 e X_2 (Eq. 2.11a).

   - $X_0(k) = \hat{A}_0 X_0(k-1) + \hat{B}_0 U_0(k-1) + \hat{B}_0 U_0(k)$
   - $X_1(k) = \hat{A}_1 X_1(k-1) + \hat{B}_1 U_1(k-1) + \hat{B}_1 U_1(k)$
   - $X_2(k) = \hat{A}_2 X_2(k-1) + \hat{B}_2 U_2(k-1) + \hat{B}_2 U_2(k)$

5. Calculate the modal historic components Y_0^H, Y_1^H e Y_2^H (Eq. 2.11b) .

   - $Y_{H0}(k) = -Y_{H0}(k-1) + \hat{C}_0 X_0(k-1)$
   - $Y_{H1}(k) = -Y_{H1}(k-1) + \hat{C}_1 X_1(k-1)$
   - $Y_{H2}(k) = -Y_{H2}(k-1) + \hat{C}_2 X_2(k-1)$

6. Convert historic values Y_0^H, Y_1^H e Y_2^H from modal to phase components Y_A^H,Y_B^H e Y_C^H, for each port p (Eq. B.7).

   - $$\begin{bmatrix} (Y_A^H)_i \\ (Y_B^H)_i \\ (Y_C^H)_i \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix}\begin{bmatrix} (Y_0^H)_i \\ (Y_1^H)_i \\ (Y_2^H)_i \end{bmatrix}, \quad i = 1, 2, \cdots, p$$

7. Calculate the new Norton Current vector I_{norton}, (Thevenin to Norton Transformation that is only necessary in impedance fitting. If in admittance fitting, the Y^H vector is already the state-space current historic output. Eq. 2.13):

   - $$\begin{bmatrix} i_{N_1} \\ i_{N_2} \\ \vdots \\ i_{N_p} \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1p} \\ G_{21} & \cdots & G_{2p} \\ \vdots & \ddots & \vdots \\ G_{p1} & \cdots & G_{pp} \end{bmatrix}\begin{bmatrix} Y_1^H \\ Y_2^H \\ \vdots \\ Y_p^H \end{bmatrix}$$

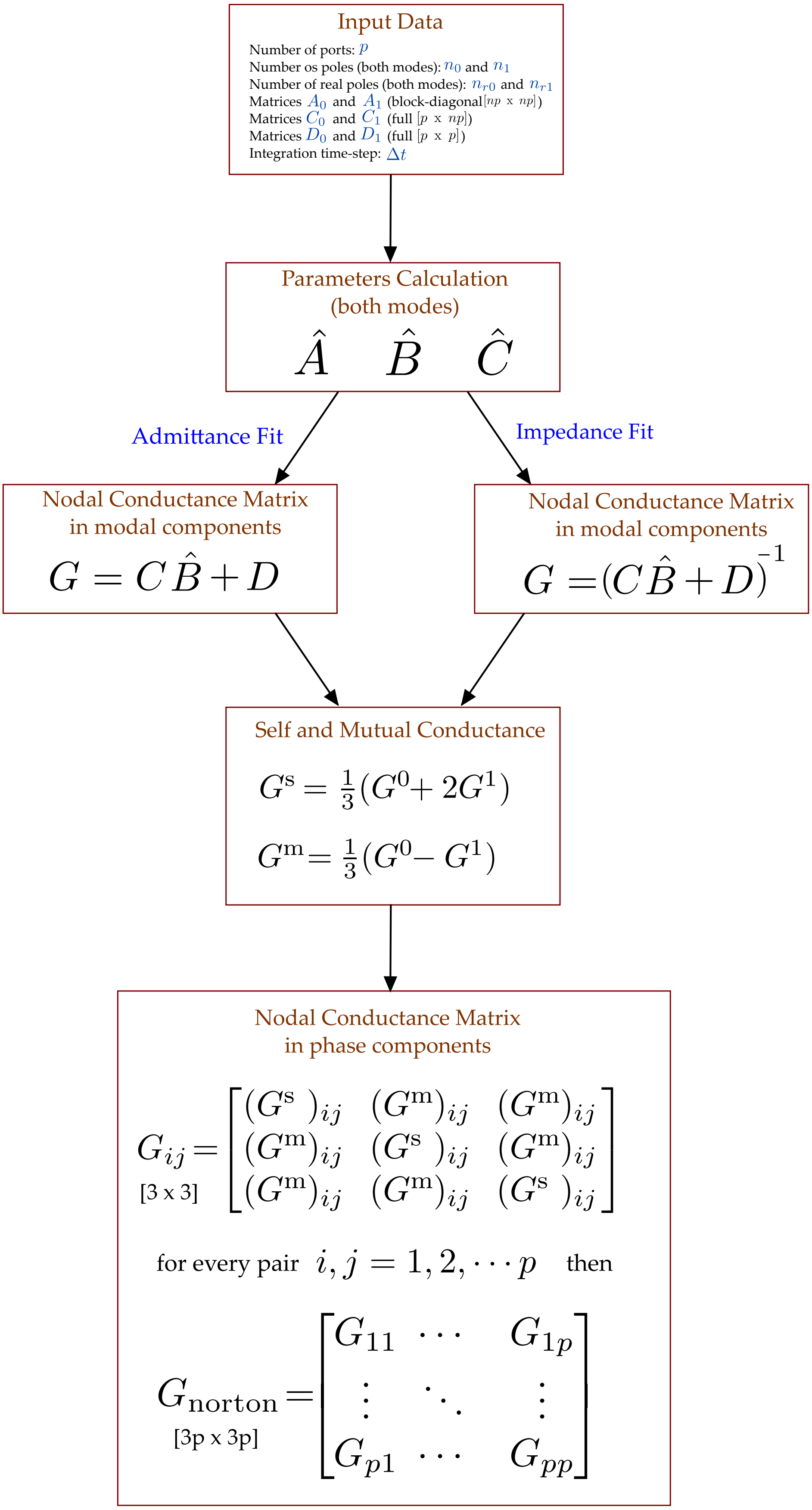

Input Data
Number of ports: p
Number os poles (both modes): n_0 and n_1
Number of real poles (both modes): n_{r0} and n_{r1}
Matrices A_0 and A_1 (block-diagonal$[np$ x $np]$)
Matrices C_0 and C_1 (full $[p$ x $np]$)
Matrices D_0 and D_1 (full $[p$ x $p]$)
Integration time-step: Δt
Parameters Calculation (both modes)
$\hat{A}$ $\hat{B}$ $\hat{C}$
Admittance Fit
Impedance Fit
Nodal Conductance Matrix in modal components
$G = C\hat{B} + D$
Nodal Conductance Matrix in modal components
$G = (C\hat{B} + D)^{-1}$
Self and Mutual Conductance
$G^{\mathrm{s}} = \frac{1}{3}(G^0 + 2G^1)$
$G^{\mathrm{m}} = \frac{1}{3}(G^0 - G^1)$
Nodal Conductance Matrix in phase components
$\underset{[3 \times 3]}{G_{ij}} = \begin{bmatrix} (G^{\mathrm{s}})_{ij} & (G^{\mathrm{m}})_{ij} & (G^{\mathrm{m}})_{ij} \\ (G^{\mathrm{m}})_{ij} & (G^{\mathrm{s}})_{ij} & (G^{\mathrm{m}})_{ij} \\ (G^{\mathrm{m}})_{ij} & (G^{\mathrm{m}})_{ij} & (G^{\mathrm{s}})_{ij} \end{bmatrix}$
for every pair $i, j = 1, 2, \cdots p$ then
$\underset{[3p \times 3p]}{G_{\mathrm{norton}}} = \begin{bmatrix} G_{11} & \cdots & G_{1p} \\ \vdots & \ddots & \vdots \\ G_{p1} & \cdots & G_{pp} \end{bmatrix}$

Figure 2.8: Flowchart of the Discrete Model parameters and Norton Conductance Matrix calculation.

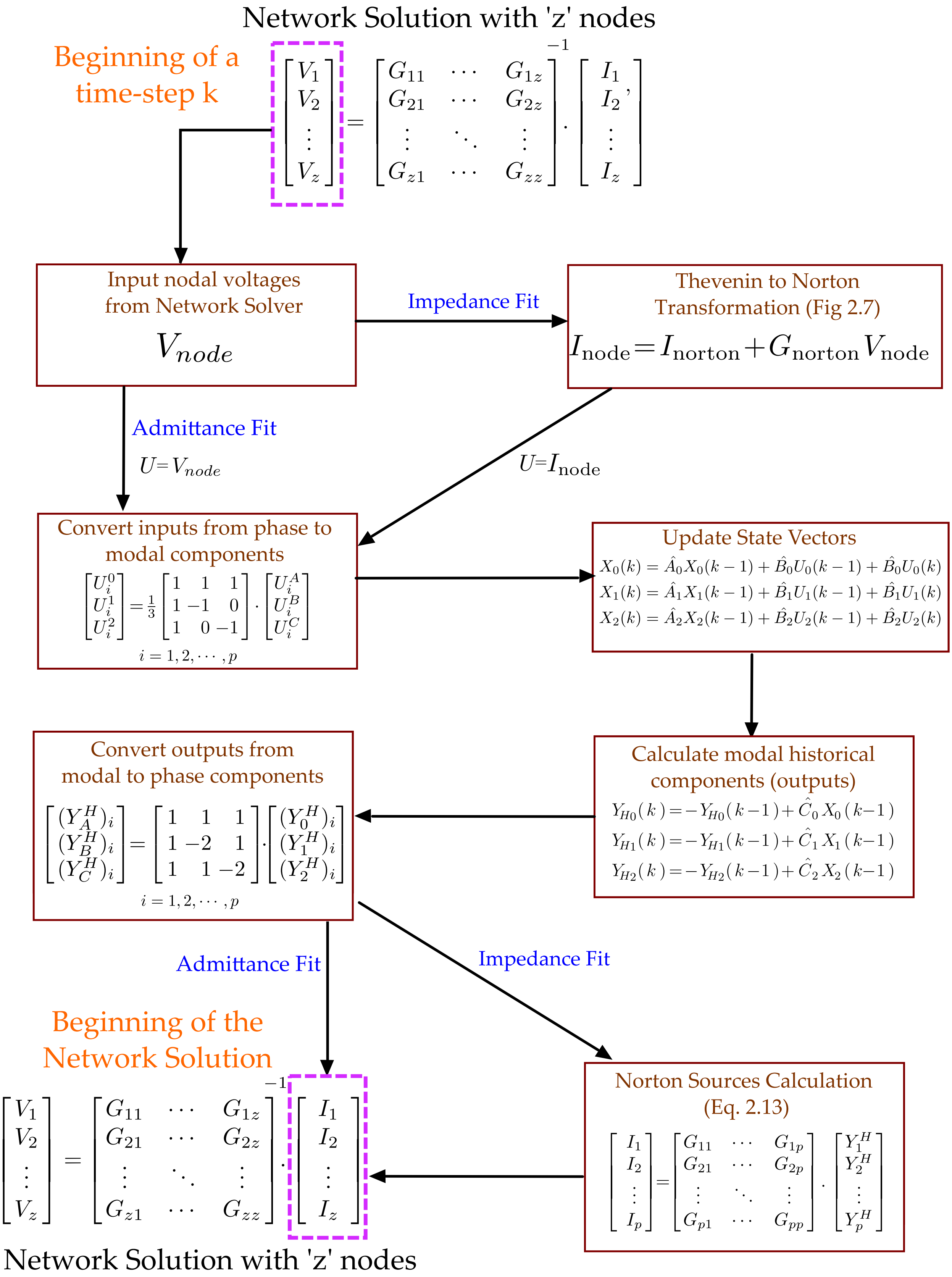

Network Solution with 'z' nodes
Beginning of a time-step k
$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_z \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1z} \\ G_{21} & \cdots & G_{2z} \\ \vdots & \ddots & \vdots \\ G_{z1} & \cdots & G_{zz} \end{bmatrix}^{-1} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_z \end{bmatrix},$
Input nodal voltages from Network Solver
V_{node}
Impedance Fit
Thevenin to Norton Transformation (Fig 2.7)
$I_{\text{node}} = I_{\text{norton}} + G_{\text{norton}} V_{\text{node}}$
Admittance Fit
$U = V_{node}$
$U = I_{\text{node}}$
Convert inputs from phase to modal components
$\begin{bmatrix} U_i^0 \\ U_i^1 \\ U_i^2 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} U_i^A \\ U_i^B \\ U_i^C \end{bmatrix}$
$i = 1, 2, \cdots, p$
Update State Vectors
$X_0(k) = \hat{A}_0 X_0(k-1) + \hat{B}_0 U_0(k-1) + \hat{B}_0 U_0(k)$
$X_1(k) = \hat{A}_1 X_1(k-1) + \hat{B}_1 U_1(k-1) + \hat{B}_1 U_1(k)$
$X_2(k) = \hat{A}_2 X_2(k-1) + \hat{B}_2 U_2(k-1) + \hat{B}_2 U_2(k)$
Calculate modal historical components (outputs)
$Y_{H0}(k) = -Y_{H0}(k-1) + \hat{C}_0 X_0(k-1)$
$Y_{H1}(k) = -Y_{H1}(k-1) + \hat{C}_1 X_1(k-1)$
$Y_{H2}(k) = -Y_{H2}(k-1) + \hat{C}_2 X_2(k-1)$
Convert outputs from modal to phase components
$\begin{bmatrix} (Y_A^H)_i \\ (Y_B^H)_i \\ (Y_C^H)_i \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} (Y_0^H)_i \\ (Y_1^H)_i \\ (Y_2^H)_i \end{bmatrix}$
$i = 1, 2, \cdots, p$
Admittance Fit
Impedance Fit
Beginning of the Network Solution
$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_z \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1z} \\ G_{21} & \cdots & G_{2z} \\ \vdots & \ddots & \vdots \\ G_{z1} & \cdots & G_{zz} \end{bmatrix}^{-1} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_z \end{bmatrix}$
Network Solution with 'z' nodes
Norton Sources Calculation (Eq. 2.13)
$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_p \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1p} \\ G_{21} & \cdots & G_{2p} \\ \vdots & \ddots & \vdots \\ G_{p1} & \cdots & G_{pp} \end{bmatrix} \cdot \begin{bmatrix} Y_1^H \\ Y_2^H \\ \vdots \\ Y_p^H \end{bmatrix}$

Figure 2.9: Interface of an FDNE component represented by Norton Equivalent into the Network Solution.

2.5 Initialization

When initialization is not requested, a first network solution is made without the contribution from the FDNE component, i.e, its first nodal injection Y_h is null. Then, the resulting voltages from the network solution are used to start the time-loop simulation. The state-space equations (2.11a) to (2.11c) requires variables from the previous iteration. These values are initialized as zeroes in the first time-loop iteration.

Conversely, initializing from a steady-state point of operation requires the FDNE model to participate in the phasor solution before the time-domain simulation. This is accomplished by calculating the admittance matrix of the equivalent and stamping it into the global admittance matrix of the system, what is done by evaluating the fitting expression (2.8) or its related transfer function (2.10) for the frequency $s = j\omega = j2\pi f$. If the fitting is based on impedance values, it is necessary to invert the resulting complex matrix. After the phasor solution, it is necessary to initialize the state vector **X** and the historic current sources. The state-vector is initialized evaluating its frequency domain expression (2.15) at the desired frequency, and taking its real part.

$$\dot{X}(s) = AX(s) + BU(s) \Rightarrow sX(s) = AX(s) + BU(s) \tag{2.15}$$

$$X(s) = [s - A]^{-1}BU(s)\Big|_{s=j2\pi f}$$

$$X_{k=0} = \Re(X)$$

Supposing an admittance-based fitting (naming the state-space inputs **U** as voltages **V** and the state-space outputs **Y** as currents **I**), the current source initialization is performed by (2.16), which is taken from the discretization development of the state-space equations, described in Appendix D.

$$I_H(k) = -I(k-1) + \hat{C}X(k-1) + GV(k-1) \tag{2.16}$$

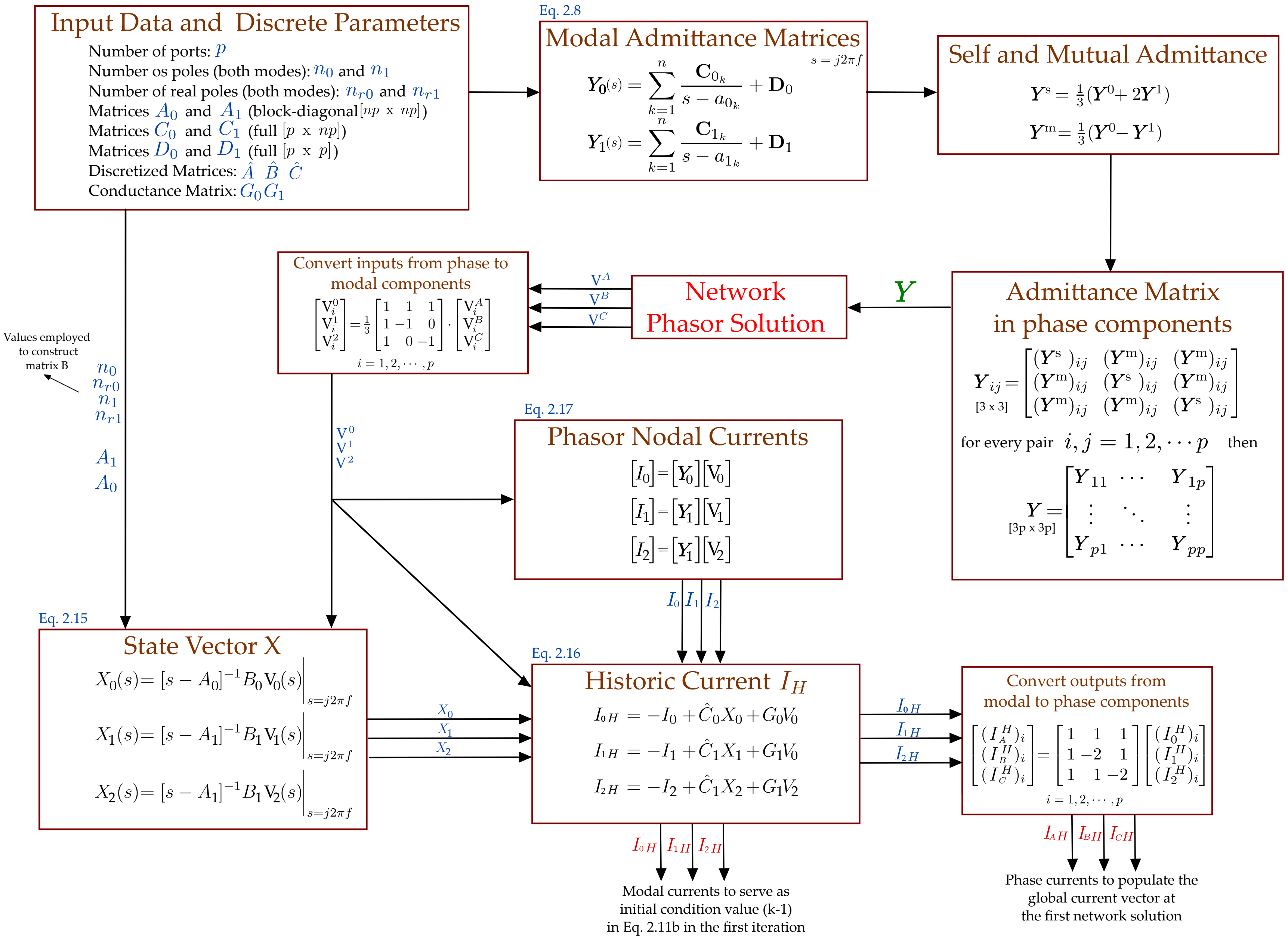
Considering the iteration k = 0 is the phasor solution, evaluating (2.16) at k = 1 yields

$$I_H(k=1) = -I_{phasor} + \hat{C}X_{phasor} + GV_{phasor}$$

where I_{phasor} is the steady-state nodal current entering the equivalent, X_{phasor} is the steady-state state-vector calculated in (2.15), V_{phasor} is the steady-state voltage at the boundary buses, $\hat{C}$ is the discretized C matrix defined in (2.12c) and G is the conductance matrix defined in (2.12d). As the FDNE component generally does not have access to the nodal currents (they are not calculated by default in EMT simulation), it is necessary to calculate them as

$$I_{phasor} = \mathbf{Y} \times V_{phasor} \tag{2.17}$$

where **Y** is the FDNE complex admittance matrix stamped at the global admittance matrix for the phasor solution of the complete system. Only the real part of the phasors is required for computation related to the initialization process. The above initialization scheme is depicted in Figure 2.10 supposing an admittance fitting in modal coordinates. A similar approach can be applied for impedance fitting, phase domain and other numerical methods.



Input Data and Discrete Parameters
Number of ports: p
Number os poles (both modes): n_0 and n_1
Number of real poles (both modes): n_{r0} and n_{r1}
Matrices A_0 and A_1 (block-diagonal $[np \text{ x } np]$)
Matrices C_0 and C_1 (full $[p \text{ x } np]$)
Matrices D_0 and D_1 (full $[p \text{ x } p]$)
Discretized Matrices: $\hat{A}$ $\hat{B}$ $\hat{C}$
Conductance Matrix: $G_0 G_1$
Eq. 2.8
Modal Admittance Matrices
$s = j2\pi f$
$Y_0(s) = \sum_{k=1}^{n} \frac{\mathbf{C}_{0_k}}{s - a_{0_k}} + \mathbf{D}_0$
$Y_1(s) = \sum_{k=1}^{n} \frac{\mathbf{C}_{1_k}}{s - a_{1_k}} + \mathbf{D}_1$
Self and Mutual Admittance
$Y^{s} = \frac{1}{3}(Y^0 + 2Y^1)$
$Y^{m} = \frac{1}{3}(Y^0 - Y^1)$
Convert inputs from phase to modal components
$\begin{bmatrix} V_i^0 \\ V_i^1 \\ V_i^2 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} V_i^A \\ V_i^B \\ V_i^C \end{bmatrix}$
$i = 1, 2, \cdots, p$
V^A
V^B
V^C
Network
Phasor Solution
Y
Admittance Matrix in phase components
$\underset{[3 \text{ x } 3]}{Y_{ij}} = \begin{bmatrix} (Y^{s})_{ij} & (Y^{m})_{ij} & (Y^{m})_{ij} \\ (Y^{m})_{ij} & (Y^{s})_{ij} & (Y^{m})_{ij} \\ (Y^{m})_{ij} & (Y^{m})_{ij} & (Y^{s})_{ij} \end{bmatrix}$
for every pair $i, j = 1, 2, \cdots p$ then
$\underset{[3p \text{ x } 3p]}{Y} = \begin{bmatrix} Y_{11} & \cdots & Y_{1p} \\ \vdots & \ddots & \vdots \\ Y_{p1} & \cdots & Y_{pp} \end{bmatrix}$
Values employed to construct matrix B
n_0
n_{r0}
n_1
n_{r1}
A_1
A_0
V^0
V^1
V^2
Eq. 2.17
Phasor Nodal Currents
$[I_0] = [Y_0][V_0]$
$[I_1] = [Y_1][V_1]$
$[I_2] = [Y_1][V_2]$
I_0
I_1
I_2
Eq. 2.15
State Vector X
$X_0(s) = [s - A_0]^{-1} B_0 V_0(s) \Big|_{s=j2\pi f}$
$X_1(s) = [s - A_1]^{-1} B_1 V_1(s) \Big|_{s=j2\pi f}$
$X_2(s) = [s - A_1]^{-1} B_1 V_2(s) \Big|_{s=j2\pi f}$
X_0
X_1
X_2
Eq. 2.16
Historic Current I_H
$I_{0H} = -I_0 + \hat{C}_0 X_0 + G_0 V_0$
$I_{1H} = -I_1 + \hat{C}_1 X_1 + G_1 V_0$
$I_{2H} = -I_2 + \hat{C}_1 X_2 + G_1 V_2$
I_{0H}
I_{1H}
I_{2H}
Convert outputs from modal to phase components
$\begin{bmatrix} (I_A^H)_i \\ (I_B^H)_i \\ (I_C^H)_i \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} (I_0^H)_i \\ (I_1^H)_i \\ (I_2^H)_i \end{bmatrix}$
$i = 1, 2, \cdots, p$
I_{0H}
I_{1H}
I_{2H}
Modal currents to serve as initial condition value (k-1) in Eq. 2.11b in the first iteration
I_{AH}
I_{BH}
I_{CH}
Phase currents to populate the global current vector at the first network solution

Figure 2.10: Flowchart of the initialization scheme.

2.6 Efficient Implementation

The FDNE performance depends mainly on how the state-space equations are solved in the time-domain. The less efficient approach would be performing the full matrix operations. Since the $\mathbf{A}$ matrix and its corresponding discretized $\hat{\mathbf{A}}$ matrix are very sparse (block diagonal with blocks of order 2), a sparsity handling approach is beneficial. Nevertheless, it would still be necessary to store the original, full-size matrix data, which for some challenging cases can reach several Gigabytes, as will be shown in the next chapter. In order to circumvent this limitation, the proposed approach stores only the non-null terms of the matrices. This leads to a great performance increase and a reduction in the size of data entry files. The VF routine was programmed to print an output data file with the state-space parameters as vectors, instead of matrices.

Since the topology of the state matrices depends only on three parameters (the number of ports, the number of poles and of these, how many are real), the state-space solver can be programmed to handle only that specific topology, what decreases its generality, but improves its performance. Since the state matrices delivered by the VF have always the same topology, the time-domain state-space solver does not need to solve generic state matrices, but only those with that expected topology. The implemented algorithm performs only basic arithmetic operations and does not require any linear algebra library. This approach was initially thought to simplify the component integration to real-time simulators since they generally do not provide linear algebra libraries to be used in their custom user components.

Every matrix equation presented in the last section will be performed using only arithmetic operations and appropriate indexing that represents the expected topology of the matrices.

2.6.1 Data pre-processing

The algorithm begins by calculating the discretized matrices. Both $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ can be calculated within only two for-loops statements, as they share the common factor $(I - \frac{\Delta t}{2}A)^{-1}$ which must be inverted. Since this matrix is block diagonal, its inverse can easily be calculated by taking the inverse of the scalar elements related with the real poles, and the inverse of the 2x2 blocks related with the complex poles, because the inverse of a block-diagonal matrix is the inverse of each block separately.

Assuming that Δt is the integration step and $\mathbf{A}$ is a vector containing the poles, ordering the real poles first, and storing only one pole of each complex pair, the following pseudo-code summarizes the calculation of the $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ matrices, both of them stored as vectors.

```
Result: $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ Matrices
for $k = 1;\ k <= n_r;$ do
    $T0 = \Delta t \times A[k]/2$
    $T1 = 1 - T0$
    $\hat{A}[k] = (1 + T0)/T1$
    $\hat{B}[k] = \Delta t/(2 \times T1)$
    $k = k + 1$
end
for $k = n_r + 1;\ k < n;$ do
    $T0 = \Delta t \times A[k]/2$
    $T1 = \Delta t \times A[k+1]/2$
    $T2 = T0^2 + T1^2$
    $T3 = 1 + T2 - 2T0$
    $\hat{A}[k] = (1 - T2)/T3$
    $\hat{A}[k+1] = (2 \times T2)/T3$
    $\hat{B}[k] = (\Delta t \times (1 - T0))/T3$
    $\hat{B}[k+1] = -\Delta t \times T1/T3$
    $k = k + 2$
end
```

The $\hat{\mathbf{C}}$ matrix is obtained in a similar approach. The C vector has all components of the C matrix, row-oriented.

```
Result: $\hat{\mathbf{C}}$ Matrix
$k = 1$
while $i < p^2$ do
    for $j = 1;\ j <= n_r;$ do
        $\hat{C}[k] = C[k] + C[k] \times \hat{A}[k]$
        $k = k + 1, j = j + 1$
    end
    for $j = n_r + 1;\ j < n;$ do
        $\hat{C}[k] = C[k] + C[k] \times \hat{A}[k] - C[k+1] \times \hat{A}[k+1]$
        $\hat{C}[k+1] = C[k+1] + C[k] \times \hat{A}[k+1] + C[k+1] \times \hat{A}[k]$
        $k = k + 2, j = j + 2$
    end
    $i = i + 1$
end
```

Next, the $\hat{\mathbf{D}}$ is calculated. This matrix carries the conductance values to be inserted in the global conductance matrix of the system. If the fitting is made with impedance values, this matrix should be inverted to obtain the conductance values. Again, the D vector is the vector representation of the D matrix, row-oriented.

```
Result: $\hat{\mathbf{D}}$ Matrix
$j = 1$
$k = 1$
while $i < p^2$ do
    $\hat{D}[k] = D[k]$
    for $l = 1;\ l <= n_r;$ do
        $\hat{D}[k] = \hat{D}[k] + C[j] \times \hat{B}[l]$
        $j = j + 1, l = l + 1$
    end
    for $l = n_r + 1;\ l < n;$ do
        $\hat{D}[k] = \hat{D}[k] + C[j] \times \hat{B}[l]$
        $\hat{D}[k] = \hat{D}[k] + C[j + 1] \times \hat{B}[l + 1]$
        $j = j + 2, l = l + 2$
    end
    $i = i + 1, k = k + 1$
end
```

2.6.2 Time-domain algorithm implementation

At every new time-step, the new historic sources must be calculated. It is made in two steps. Firstly, the state vector **X** is updated as shown in the following pseudocode. Here, the vectors **U** and U_H are the state-space inputs at the actual and previous time-step, respectively and X_H is the state vector at the previous time-step.

```
Result: State Vector X
$k = 1$
while $i < p$ do
    $A1 = U[i] + U_H[i]$
    for $l = 1;\ l <= n_r;$ do
        $X[k] = \hat{A}[l] \times X[k] + \hat{B}[l] \times A1$
        $k = k + 1, l = l + 1$
    end
    for $l = n_r + 1;\ l < n;$ do
        $X[k] = \hat{A}[l] \times X_H[k] + \hat{A}[l + 1] \times X_H[k + 1] + \hat{B}[l] \times A1$
        $X[k + 1] = -\hat{A}[l] \times X_H[k] + \hat{A}[l] \times X_H[k + 1] + \hat{B}[l + 1] \times A1$
        $k = k + 2, l = l + 2$
    end
    $i = i + 1$
end
```

Finally, the historic sources are calculated. The vectors **Y** and Y_H denote the historic sources at the current and previous time-step, respectively.

```
Result: Historic Sources $\mathbf{Y}$
$j = 1$
$k = 1$
for $l = 1;\ l <= p;$ do
    $Y[l] = -Y_H[l]$
    $i = 1$
    for $m = 1;\ m <= p;$ do
        for $z = 1;\ z <= n;$ do
            $Y[l] = Y[l] + \hat{C}[j] \times X[i]$
            $i = i + 1, j = j + 1$
        end
        $m = m + 1$
    end
    $i = i + 1$
end
```

2.7 Passivity Enforcement Routine

The FDNE time-domain implementation based on Vector Fitting can provide an unstable response if the model is not guaranteed to be passive, i.e., if the passive part of the equivalent generates energy. Thus, a passivity-enforcement post-processing step is often needed. There are several algorithms developed for this purpose and it is still a field of research. In this work, the Matlab routine available at SINTEF's website [48, 49] based on Quadratic Programming optimization was employed to assure the FDNE stability.

The passivity diagnosis of the model is performed by calculating the eigenvalues of the fitted impedance (or admittance) matrix for all frequencies within a given range. If any negative eigenvalue is found, the model is not passive, and then the post-processing passivity enforcement routine is necessary. The user can also opt for changing the number of poles or the number of iterations in the vector fitting algorithm to achieve a passive implementation, in a heuristic manner.

Figure 2.11 presents a flowchart of the implemented passivity diagnostic algorithm embedded in the SINTEQV tool. For illustrative purposes, an admittance matrix of order 2 with 10 poles is shown.

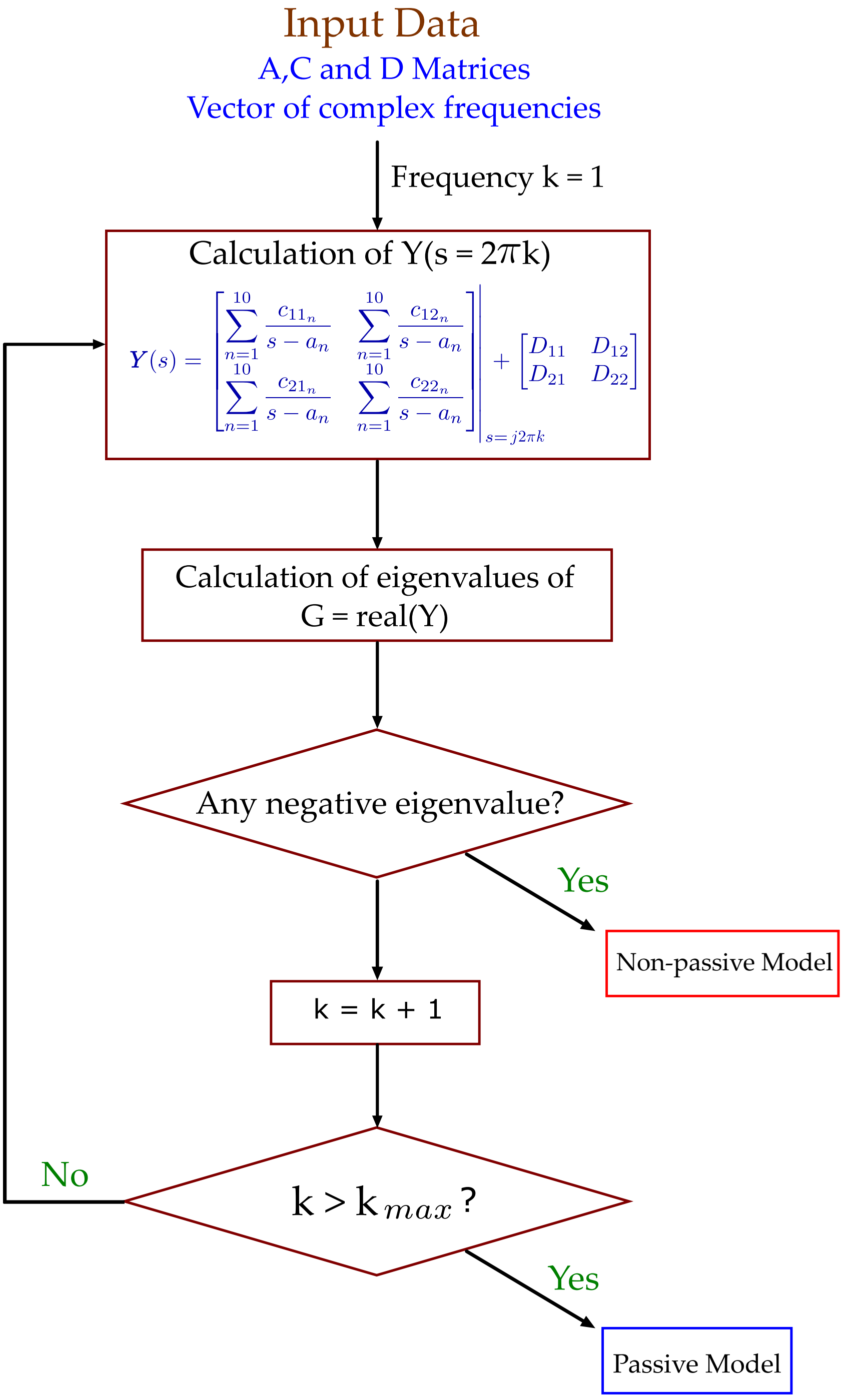
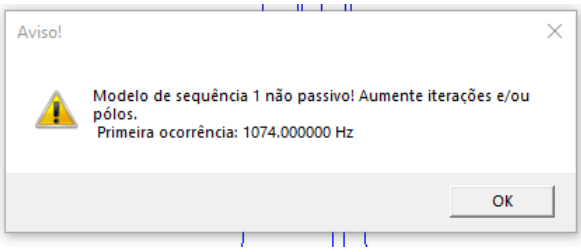

Input Data
A,C and D Matrices
Vector of complex frequencies
Frequency k = 1
Calculation of Y(s = 2πk)
$Y(s) = \left.\begin{bmatrix} \sum_{n=1}^{10} \frac{c11_n}{s-a_n} & \sum_{n=1}^{10} \frac{c12_n}{s-a_n} \\ \sum_{n=1}^{10} \frac{c21_n}{s-a_n} & \sum_{n=1}^{10} \frac{c22_n}{s-a_n} \end{bmatrix}\right|_{s=j2\pi k} + \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}$
Calculation of eigenvalues of G = real(Y)
Any negative eigenvalue?
Yes
Non-passive Model
k = k + 1
No
$k > k_{max}$?
Yes
Passive Model

Figure 2.11: Passivity diagnosis flowchart.

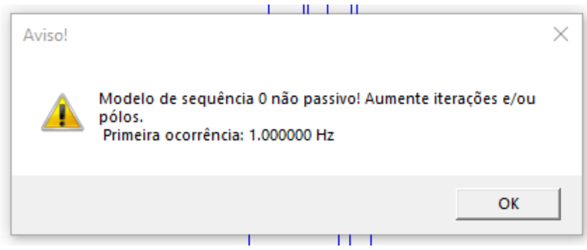
The diagnosis is executed after the calculation of the state-space matrices A, C and D, which are used to calculate the eigenvalues of the real part of the admittance matrix for each frequency within the fitted range. If no negative eigenvalues are found, the model is considered passive within the given range.

To demonstrate its operation, a non-passive case will be shown. It is a three-port equivalent fitted with 64 poles. The frequency scan was done in ATP, as explained in Appendix C, providing the VF routine the frequency responses associated with the three buses, for both sequence modes. After the fitting, the passivity check algorithm found passivity violations, as illustrated in Figures 2.12 and 2.13. The pop-up window displays the first frequency for which a negative eigenvalue was found.


Aviso!

Modelo de sequência 1 não passivo! Aumente iterações e/ou pólos.
Primeira ocorrência: 1074.000000 Hz

Figure 2.12: Message-box warning the first passivity violation found for the non-homopolar mode.


Aviso!

Modelo de sequência 0 não passivo! Aumente iterações e/ou pólos.
Primeira ocorrência: 1.000000 Hz

Figure 2.13: Message-box warning the first passivity violation found for the homopolar mode.

If the model cannot be made passive by changing the number of poles and/or iterations, it is exported normally, and then the post-processing routine must be used to impose passivity.

A 3-port FDNE in modal domain has six frequency responses inputs, three for each mode. Figure 2.14 presents the eigenvalues plots of the six fitted frequency responses, approximating around the zero in the y axis to highlight the passivity violations. As expected, the first negative eigenvalue occur in 1 Hz for the homopolar mode and in 1074 Hz for the non-homopolar mode.

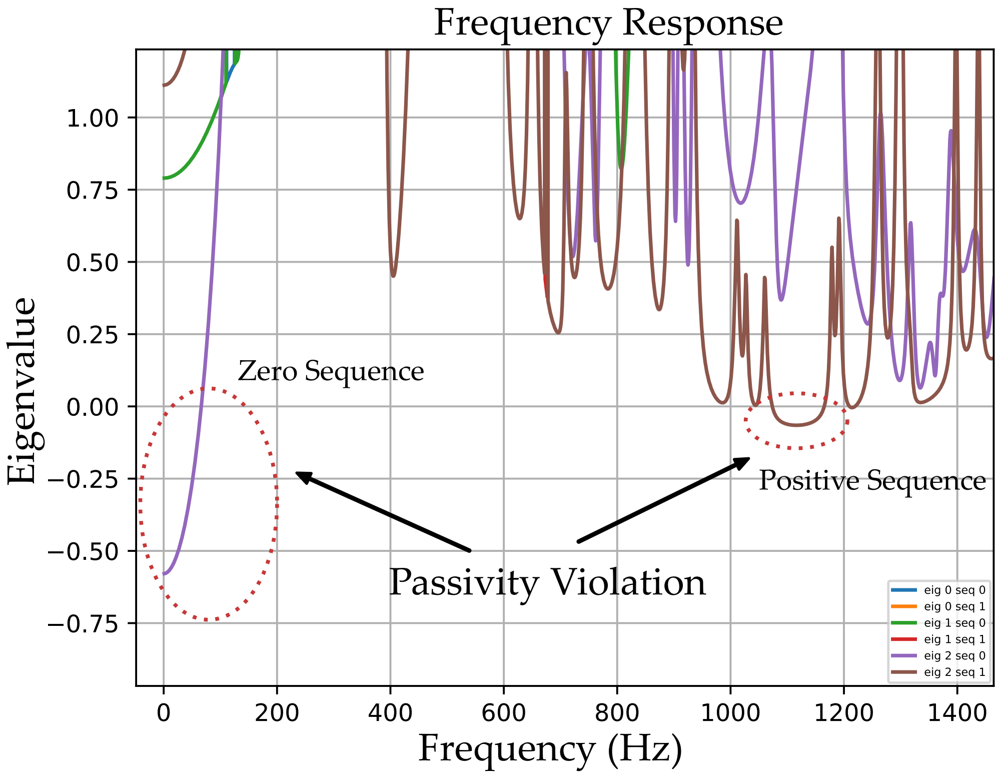


Figure 2.14: Ground and aerial modes eigenvalues, highlighting the passivity violation.

After the passivity enforcement routine, all eigenvalues are made positive, as depicted in Figure 2.15 which shows the homopolar mode eigenvalues before and after the routine. The comparison is made only for the frequency responses that violated passivity. The highlighted window in the figure approximates around the region of violation, showing that all eigenvalues after the imposition of passivity (in blue) become positive. Figure 2.16 presents the same comparison for the homopolar mode. As before, the negative eigenvalues become positive. The quadratic optimization used in the passivity enforcement routine imposes the all positive eigenvalues condition while minimizing the frequency response deviation. Figures 2.17 and 2.18 show the six nodal impedance frequency responses before and after the passivity enforcement. Good agreement is obtained. The errors are greater in regions where passivity was originally violated.

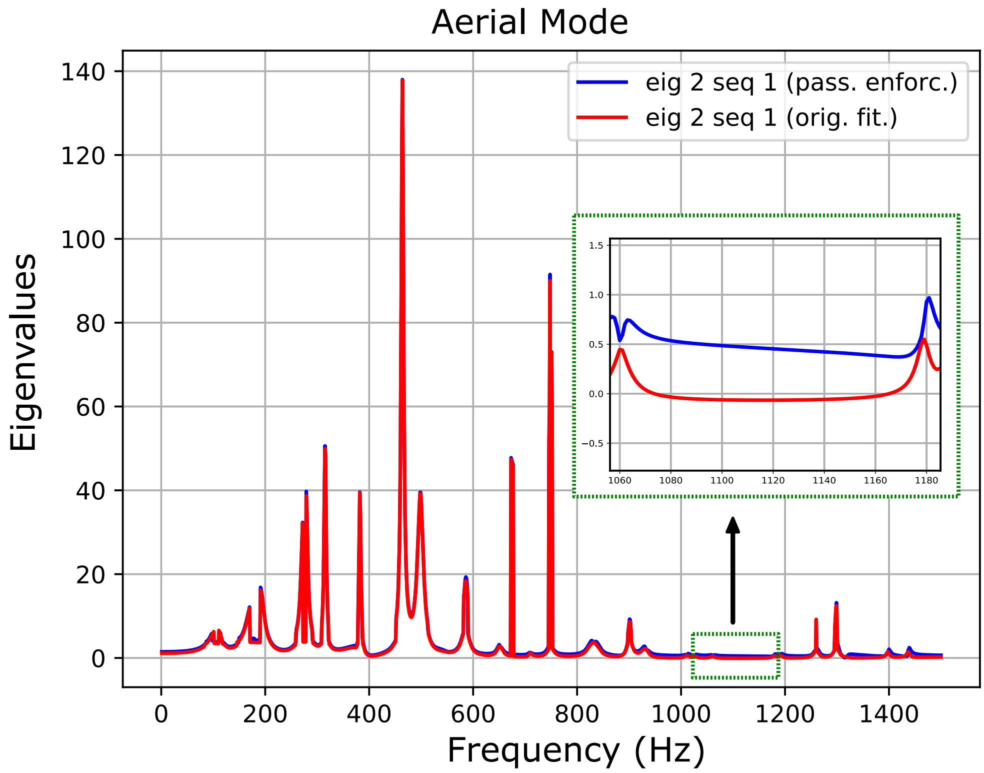


Figure 2.15: After and before passivity enforcement comparison of homopolar mode eigenvalues.

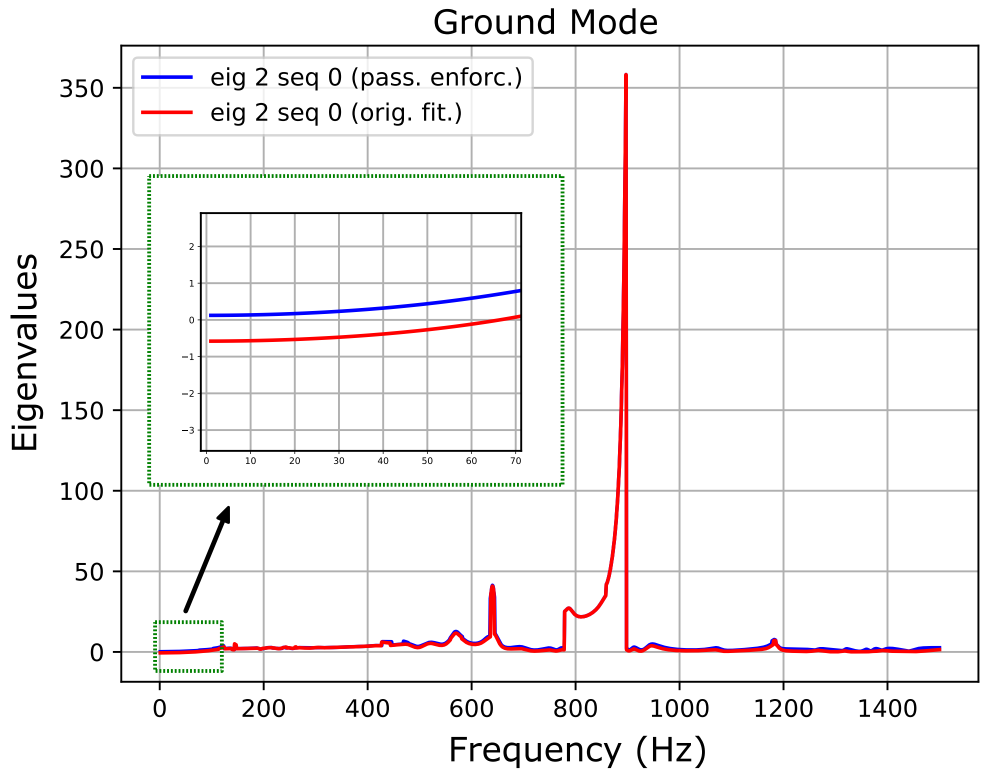


Figure 2.16: After and before passivity enforcement comparison of ground mode eigenvalues.

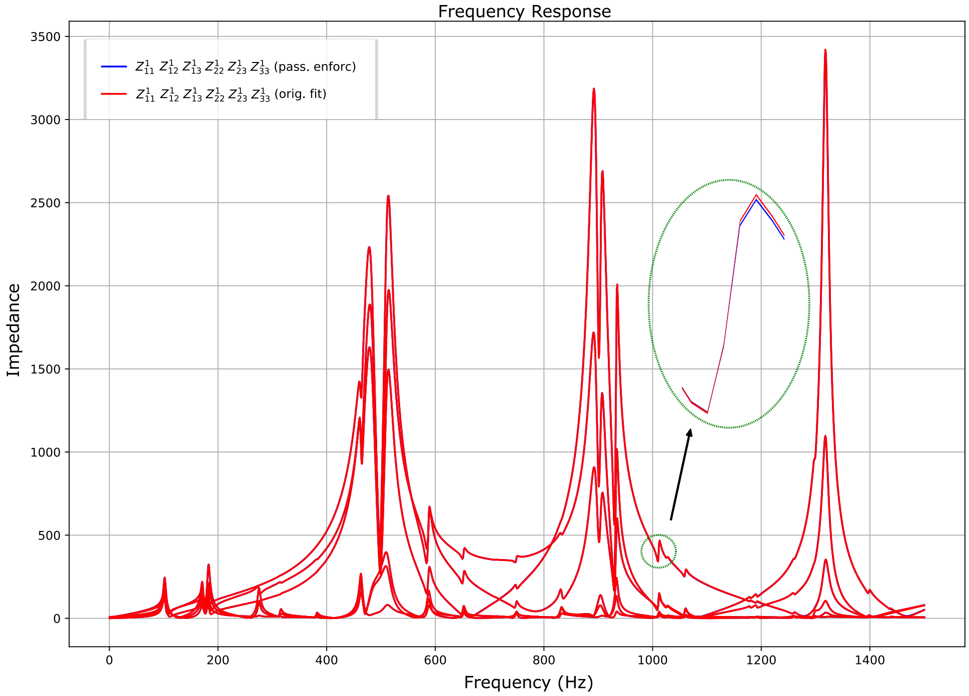


Figure 2.17: After and before passivity enforcement comparison of nodal impedance frequency responses for non-homopolar modes.

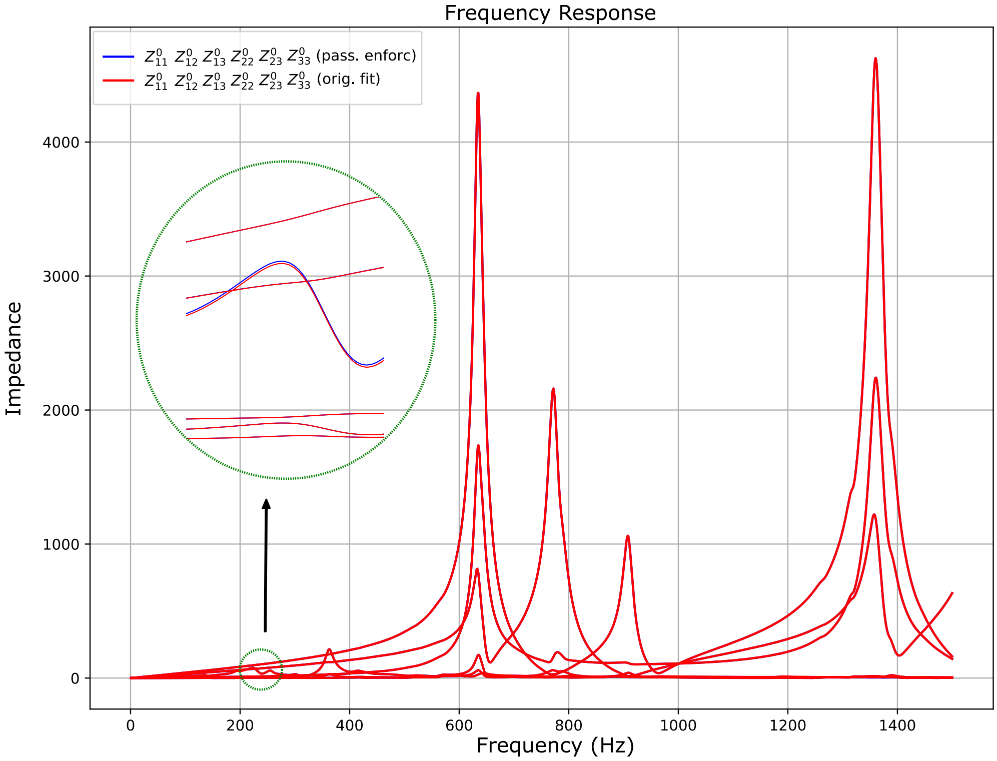


Figure 2.18: After and before passivity enforcement comparison of nodal impedance frequency responses for homopolar modes.

In this example case, the passivity enforcement is applied to the same frequency range that was used in the vector fitting (0 to 1500Hz). To ensure that the discretized Norton equivalent conductance matrix is positive-defined for the typical values of the integration time-step, it was observed that an increased passivity range was necessary. For a time-step of 50 μs, the passivity enforcement up to 5kHz was sufficient.

2.8 Chapter Outline

The vectorized FDNE algorithm presented in the subsection 2.6.2 is the core of implemented models along with this work. Its performance will be evaluated both for offline simulation in EMTP and for real-time in the RTDS, OPAL-RT and an FPGA board. The proposed method works through the vectorization of the state matrices in order to perform only arithmetic operations in the simulation code. This leads to a great performance increase and a substantial reduction in the size of the data files. Taking a large case as an example, consider an FDNE with 5 three-phase buses (15 boundary nodes), adjusted with 300 poles. As the A matrix order is pn, where p is the number of ports and n the number of poles, this leads to a 4500 order A matrix (in phase coordinates), which carries $4500^2 \approx 20 \times 10^6$ entries, of which only 300 are necessary. A sparsity handling approach would be sufficient for improving the time-domain loop, but since the entry data are generally the full matrices, it would require a long pre-processing time to read such big matrices and extract the needed information. The vectorized approach, on the other hand, can process the vectorized entry data very fast, as will be shown in the next chapter.

Chapter 3

EMTP® Implementation

This chapter presents the FDNE implementation in the EMTP® software. A large real network was modeled in order to assess the performance and accuracy of the developed component for a single port and a 10-port network reduction, in both phase and modal coordinates. Comparisons were made with the full network for validation purposes and with the conventional short circuit 60 Hz Thévenin equivalent to better point out the advantages of FDNEs.

3.1 Custom Component Structure in EMTP®

In the EMTP®, a generic user custom model is interfaced to the main solver by DLL files, through a method of request and participation, in which the DLL file must reply to request procedures callable from the EMTP Core. Some of these methods are mandatory, others depend on the type of simulation being executed by the core. For example, if initialization is requested in the GUI, the DLL file must contain the initialization procedure (a phasor solution at a given frequency) within a function that will be called only once at the start of the simulation. If that function does not exist in the DLL, an error will be raised and the simulation will be stopped.

The DLL can be programmed in any language, but since the EMTP core is written in Fortran-95, the same language was used for the DLL compiling. The compiler used was the Intel Visual Fortran Compiler with Microsoft Visual Studio, as recommended in the EMTP user manual. A basic example is provided in the default EMTP installation and was used as a starting template for the DLL generation. The EMTP Core provides both Trapezoidal and Euler numerical methods, as well as the Critical Dumping Adjustment (CDA) procedure that alternates between Trapezoidal and Euler methods to avoid switching oscillations. These features were all programmed in the FDNE DLL component to make it fully compatible with the EMTP tool.

3.2 Studied Network

As an example for assessing the FDNE accuracy and performance, a large network representing part of the southeast Brazilian transmission system has been modeled in the EMTP. This case was originally developed in ATP for studying the introduction of a 500 kV transmission line connecting Fernão Dias and Terminal Rio substations in the states of São Paulo and Rio de Janeiro, respectively. It was ported to EMTP preserving the original topology and device modeling. It is composed of 134 three-phase buses, 222 transformers (two and three-winding), 104 constant parameters (Bergeron) transmission lines in 500 kV, 440 kV and 345 kV, 189 PI circuits representing transfer impedance between buses and 67 sources representing generators and Thévenin equivalents for voltage levels below 345 kV. The database conversion was validated by comparing the steady-state solution and also the impedance frequency responses as seen from several buses. The full network is illustrated in Figure 3.7.

The time-domain validation of the equivalent was done comparing the transient response after the energization of the first 500/336KV – 1186 MVA converter transformer of the Terminal Rio substation at the worst inrush condition (the time of which was obtained through statistical simulation) and under the unavailability of the Cachoeira Paulista - Fernão Dias 500 kV transmission line. This unavailability was found to shift the impedance frequency response seen from the Terminal Rio substation causing a resonance near the second harmonic (120 Hz). Since this is the main component (can reach 63%) of typical inrush wave-forms, its representation is essential to well evaluate the impact of the inrush currents. The voltage before the breaker closing was adjusted to the maximum operative value of 1.1 p.u. for all simulations. The per-unit base of peak line to ground voltage and peak current were defined as

$$V_b = \frac{V_{ll}}{\sqrt{3}} \times \sqrt{2} = \frac{500}{\sqrt{3}} \times \sqrt{2} = 408.24\ kV$$

$$I_b = \frac{P_{3\phi}}{\sqrt{3} \times V_{ll}} \times \sqrt{2} = \frac{1186}{\sqrt{3} \times 500} \times \sqrt{2} \times 1000 = 1936.7A$$

where V_{ll} and $P_{3\phi}$ are the line to line RMS voltage rating of the high side of the transformer and its three-phase power rating, respectively.

3.3 Case 1: Single Port Equivalent

As a first evaluation of the FDNE, the entire network except the transformer will be reduced. The resulting 1-port equivalent is shown in Figure 3.1. The frequency response of the impedance seen at the transformer bus was adjusted from 1 Hz to 2000 Hz, with a step of 1 Hz, with 100 poles for both sequences. The responses are shown in Figure 3.2 for the full network, the FDNE and the 60 Hz equivalent. The error of the FDNE response is shown in gray and has its maximum value of about 30 Ω at the frequency of 665 Hz. The short circuit equivalent frequency response shown in green has a linear shape with fixed slope since it is the impedance of a RL circuit. The green line matches with the original and the FDNE responses at the power frequency of 60 Hz, as expected.

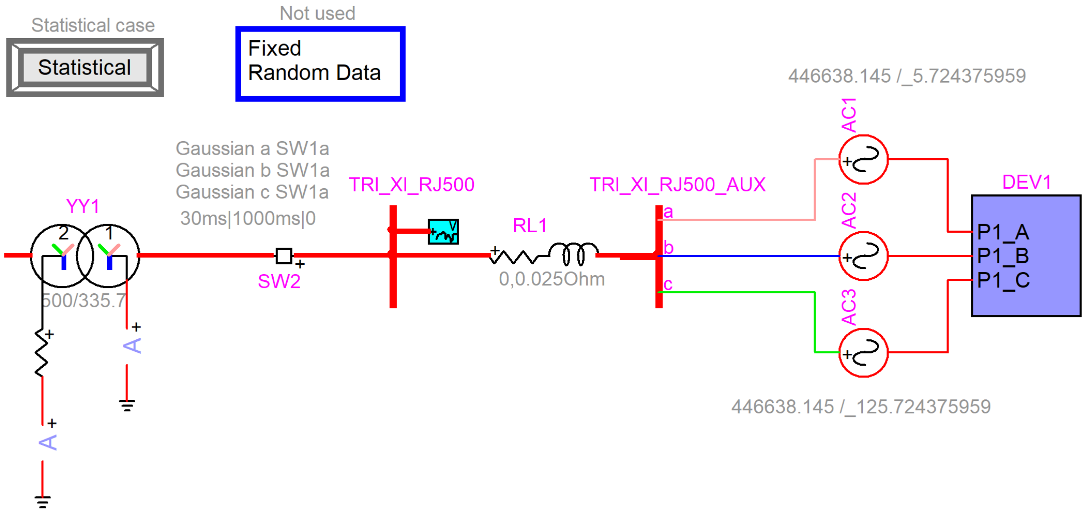


Figure 3.1: 1-port network reduction.

Statistical analysis has been performed for both complete and reduced systems, executing 200 simulations with a time-step of 20 μs and a total time of 300 ms. The closing time for each phase was obtained by a Gaussian distribution over a mean μ systematically distributed within one period, with a step of $\frac{1}{60\times200} = 83.33$ μs, ranging from 24 ms to 40 ms, generated by a fictitious switch. A standard deviation σ of 1.25 ms was used in the Gaussian distribution. Since EMTP by default employs a dispersion of $\sqrt{3}\sigma$ over the mean, a pole spread of $2\times\sqrt{3}\times1.25 \approx 4.3$ ms was obtained. The seed for the random number generator was predefined to generate the same set of switching times for both statistical simulations since our task is to assess the proximity of the FDNE and the full system responses. Figure 3.3 shows the switching times for each phase, as well as the systematic generated means within one period.

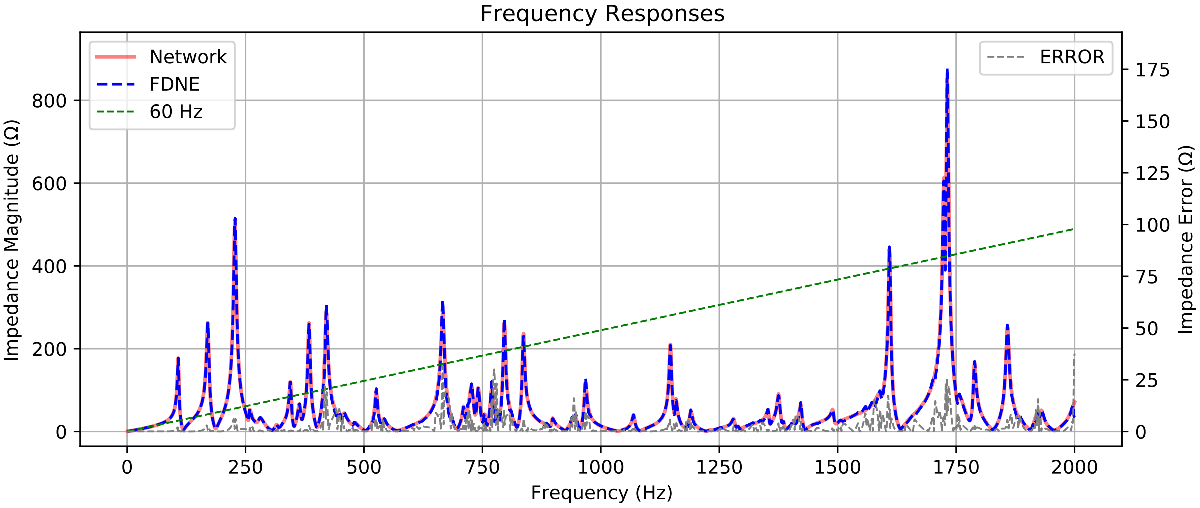


Figure 3.2: Frequency responses seen at the transformer bus.

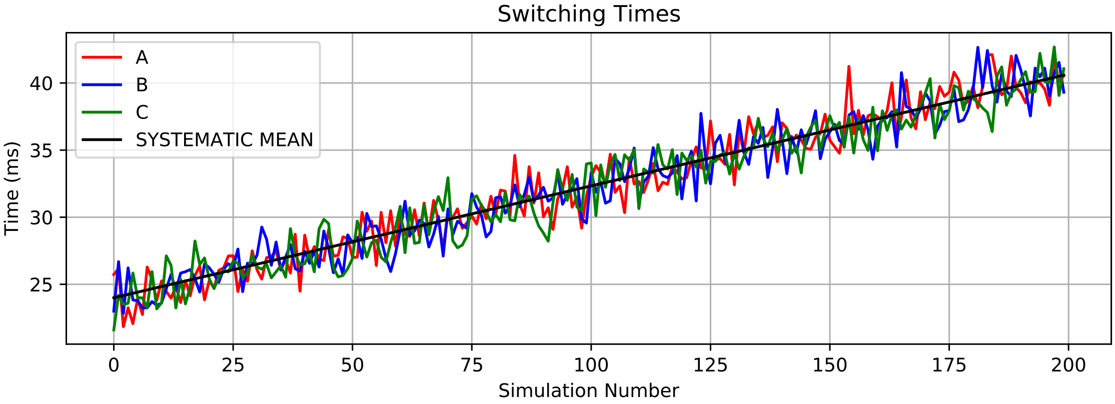


Figure 3.3: Switching times for phases A, B and C along the statistical analysis.

Table 3.1 presents the statistical analysis results executed with a pre-closing voltage of 1.1 pu. All metrics presented inaccuracies smaller than 10^{-2} per unit, indicating that the FDNE can replace the external network with virtually no loss of information. The full system took approximately 35 minutes to be simulated, while the reduced network ran almost 60 times faster, being executed in about 30 s. This speed increase is very useful, especially for switching studies with different scenarios, in which transmission line unavailability and topological changes have to be assessed.

Table 3.1: Statistical Results - 1 Port equivalent.

Case	Voltage (per unit)			Inrush Current (per unit)		
	Mean	SD σ	Maximum	Mean	SD σ	Maximum
Full Network	1.288	0.0495	1.429	3.231	0.47812	3.705
1 Port FDNE	1.287	0.0482	1.423	3.227	0.47806	3.703
Inaccuracy	0.00060	0.00133	0.005947	0.00315	0.00006	0.00263

The worst cases of overvoltage and inrush currents were simulated deterministically, corresponding to the simulations number 106 and 193, respectively. These cases are shown in Figures 3.4 and 3.5, comparing the waveforms of the complete and the reduced systems. The maximum deviations are 0.028 pu for voltage and 0.17 pu for current, which are negligible over the maximum values of 1.42 pu for voltage and 3.7 pu for current. Moreover, the maximum discrepancies tend to occur long after the first cycles and the worst inrush current is frequently observed in the first cycles. The difference in the first inrush peak was only 5 A, or approximately 0.0026 pu. The performance gains of the network reduction for several different numbers of ports and poles will be covered in section 3.5.

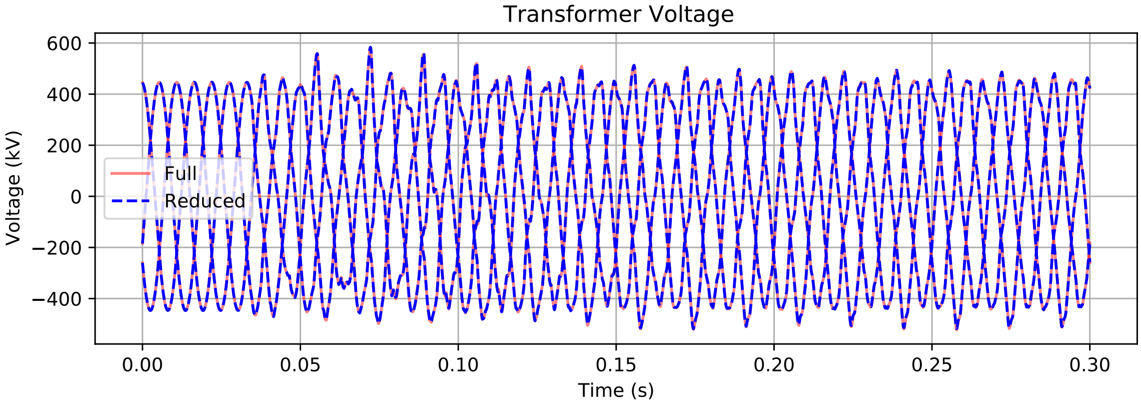


Figure 3.4: Worst overvoltage energization for both full and reduced systems.

3.4 Case 2: Multi-port Equivalent

Reducing the network to a single port system is straightforward, especially when modal coordinates are used, which leads to three state-space systems of order 1. Small systems like that generally do not have passivity issues, which are often observed for multi-port cases.

A very challenging multi-port case was made preserving mainly the substations in 500 kV and placing the others inside the equivalent. This resulted in 10

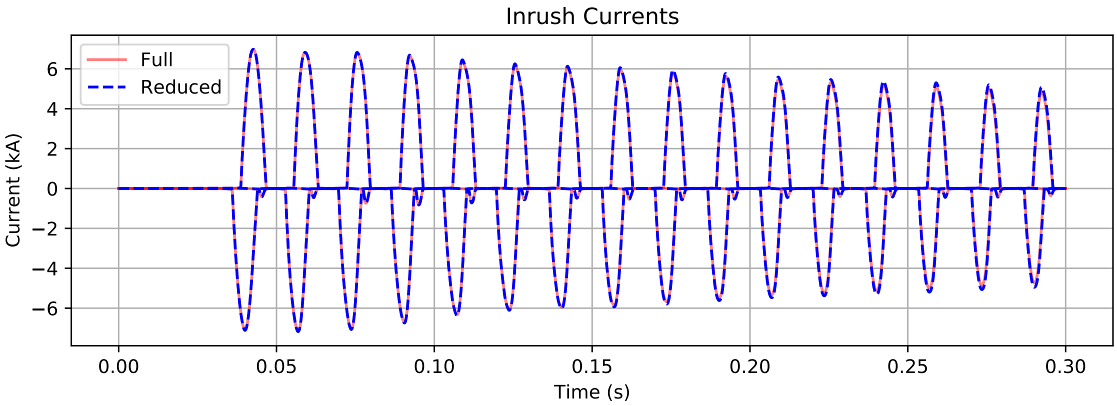


Figure 3.5: Worst inrush currents for both full and reduced systems.

boundary buses. The reduced system is illustrated in Figure 3.8, and the FDNE inside the DLL component is depicted in Figure 3.6. To fairly compare the short-circuit equivalent and the FDNE, a minimum electrical distance of two buses was maintained between the equivalent and the switching point. This rule of thumb is a mean of filtering the high frequencies of the switching waveform, what is necessary because the impedance of short-circuit equivalents is adjusted only for the power frequency, not for the higher ones. For the FDNE this constraint is not necessary, as it is expected to reproduce the all switching waveform frequency spectrum, as demonstrated through the complete reduction of the network in the previous section.

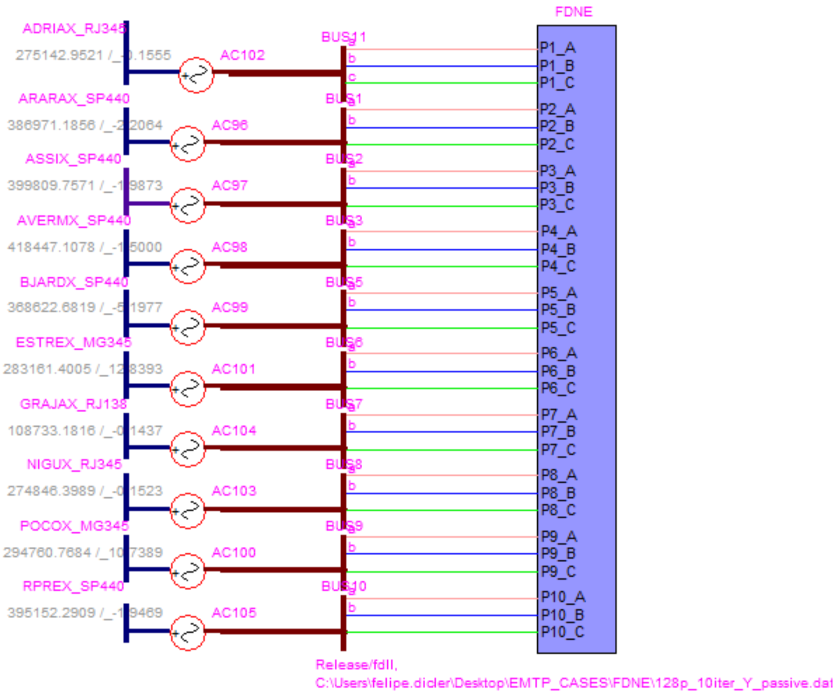


Figure 3.6: FDNE component.

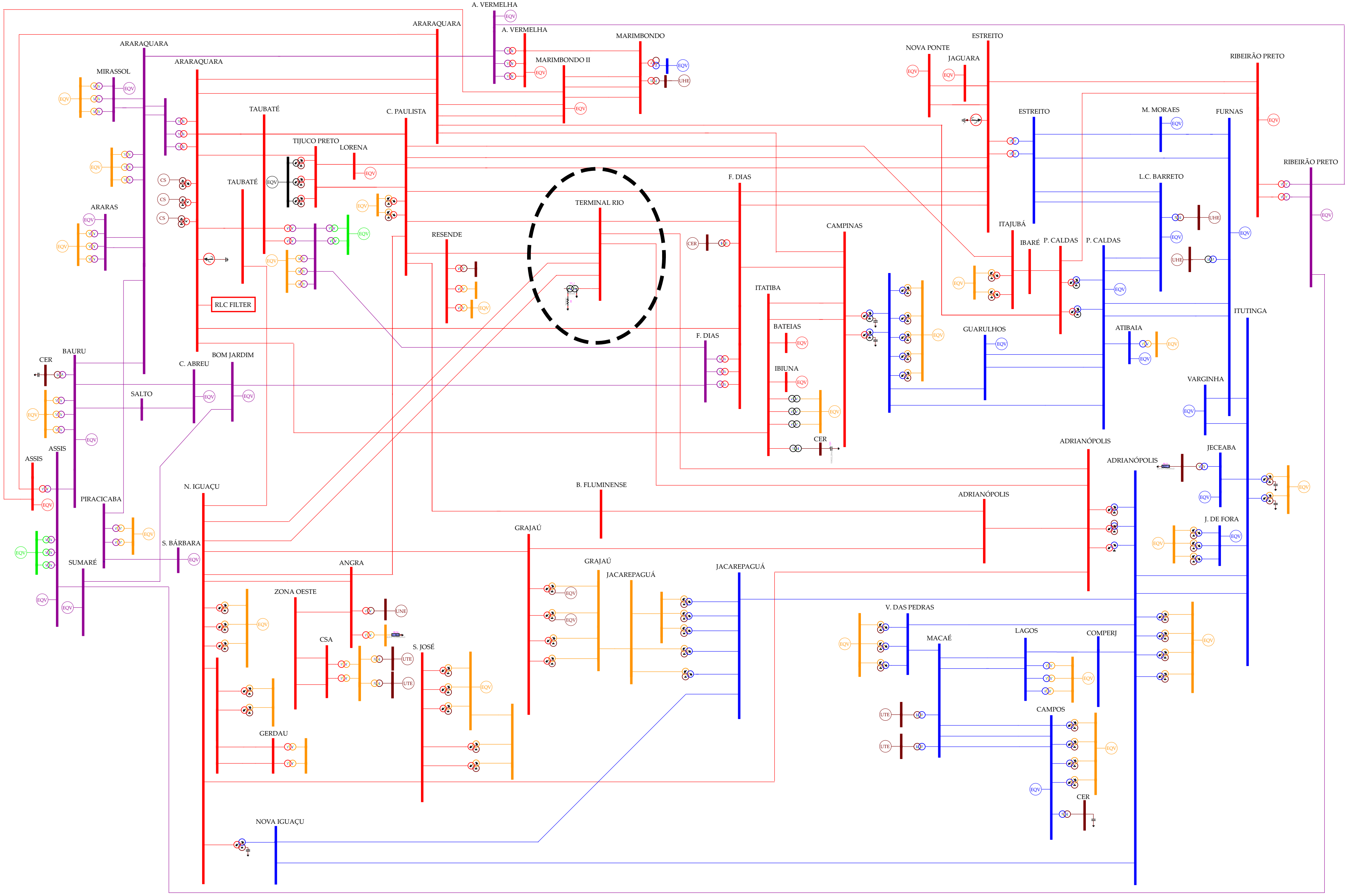


Figure 3.7: Single-line diagram of the power system modeled in EMTP.

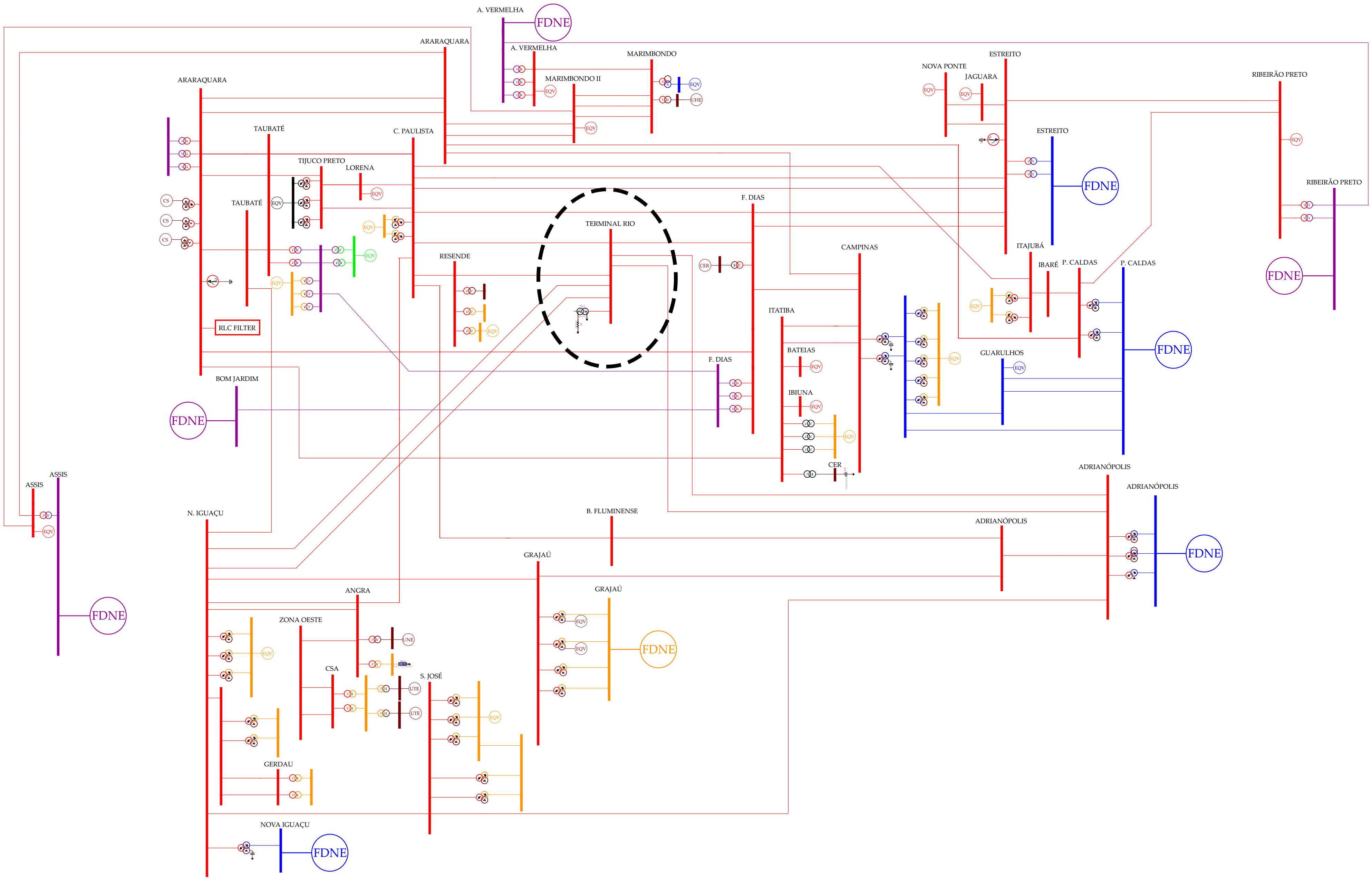


Figure 3.8: Single-line diagram of the reduced power system.

After obtaining the frequency responses for the boundary buses, the FDNE was generated through the Vector Fitting routine. As the network is completely balanced, the modal coordinates were used. The active part of the equivalent is obtained by removing the detailed network and reading the open circuit voltages at the boundary buses. These values give the Thevenin sources of both the FDNE and the short-circuit equivalents. It is also possible to convert them to Norton Sources, multiplying the voltage vector by the 60 Hz admittance matrix. This is necessary when the network solver does not handle ungrounded sources, as is the case of HYPERSIM, which is based on the traditional nodal approach instead of the Modified Nodal Analysis.

The frequency responses are obtained in modal coordinates, so the impedance matrices of the 10-port equivalent have order 10. A total of 90 frequency responses were fitted, with 55 for each sequence. This number is obtained as follows:

$$N_f = \frac{p \times (p+1)}{2} \times 2 = \frac{10 \times 11}{2} = 55 \tag{3.1}$$

where N_f is the number of frequency responses and p is the number of ports. This result is valid for modal impedance matrices, in which the number of non-repeating entries is the number of branches $p \times (p+1)$ and for balanced networks $(Z_1 = Z_2)$.

A different number of poles and frequency ranges were tested. All fittings for the 10-port case resulted in severe passivity violations, even for cases in which the fitting error was almost null. For orders higher than 300 poles, the passivity routine crashed due to memory limitation. Maximum errors of less than 1 Ω were observed using about 150 poles for the positive and 250 for the zero sequence. Employing 200 poles for both sequences in admittance units, fitting from 1 Hz to 2 kHz with a step of 2 Hz, with 5 iterations yielded a maximum error of 0.5 Ω in the positive and 262 Ω in the negative sequence, taking approximately 30 s to be executed. But these high-order cases often required a long time (1 to 3 hours) to reach passivity or even crashed the routine due to lack of RAM memory.

From the tests that have been performed a trade-off between fitting accuracy and frequency range was observed, i.e, increasing the frequency range imposes the employing of a higher-order approximation but makes passivity enforcement more difficult. On the other hand, a smaller frequency interval demands fewer poles and makes passivity enforcement easier, at the cost of decreasing the number of harmonics represented. Two cases were selected to compare both approaches.

3.4.1 128 poles fitting in positive and zero sequences

The first one, shown in Figure 3.9, was approximated from 1 to 2 kHz with 1 Hz of step with 128 poles for both sequences. For convenience, only the first response is shown, i.e, the self impedance seen from the Adrianópolis 345 kV substation. The other responses observed similar behavior with errors of the same order and several resonances. Large differences seen in the impedance peaks are introduced by the passivity enforcement routine. Although this lack of accuracy at the maxima, the series resonances are well fitted.

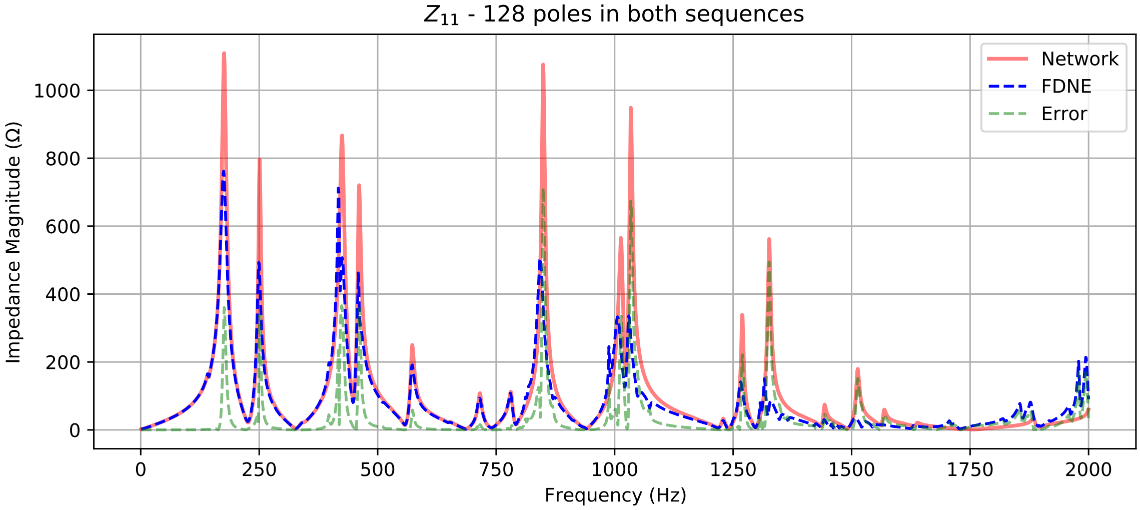


Figure 3.9: Frequency response of Z_{11} in external network and FDNE for the 1 - 2 kHz fit.

3.4.2 28 poles in zero and 40 poles in positive sequence

Another approach was done by decreasing the frequency range to 1 - 500 Hz and employing considerably fewer poles, leading to less severe passivity violations while still achieving sufficient accuracy in the fitting interval. Figure 3.10 shows the results of this second approximation, with 26 poles for positive and 40 poles for zero sequence.

3.4.3 Results

The same statistical methodology was applied for these cases, performing 200 simulations with a time-step of 20 μs and a total time of 300 ms, employing the same set of closing times for the circuit breaker as the 1-port example and evaluating the same transformer energization. Table 3.2 shows a comparison between the following statistical cases: i) the complete system; ii) the reduced system by Thevenin

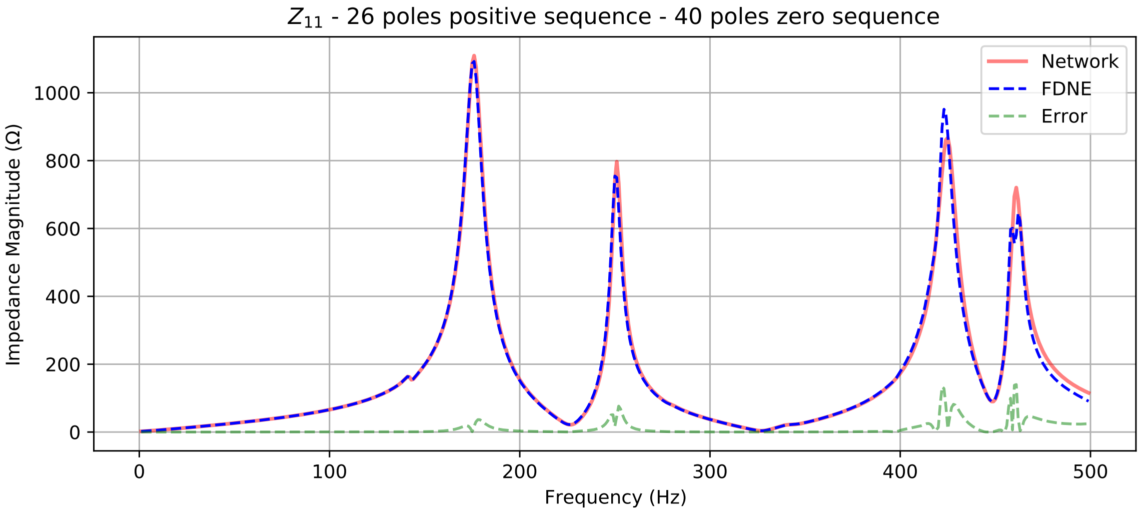


Figure 3.10: Frequency response of Z_{11} in external network and FDNE for the 1 - 500 Hz fit.

Equivalent; iii) the reduced system by a 10 port FDNE comprising 128 poles in both sequences and iv) the reduced system by a 10 port FDNE with 26 poles in positive and 40 poles in the zero-sequence.

Table 3.2: Statistical Results - 10 port equivalent.

Case	Voltage (per unit)			Inrush Current (per unit)		
	Mean	SD σ	Maximum	Mean	SD σ	Maximum
Full Network	1.288	0.0495	1.429	3.231	0.478	3.705
Thevenin	1.474	0.0678	1.644	3.304	0.507	3.869
FDNE 128 poles	1.256	0.0423	1.375	3.321	0.487	3.801
FDNE 26/40 poles	1.291	0.0504	1.436	3.250	0.481	3.725

The Thevenin Equivalent presented the highest deviation from the original system, while both FDNEs provided similar accuracy. The 26/40 poles FDNE achieved the best approximation as expected because its frequency response was the closest to the original response (in the first harmonics, up to 500 Hz).

The closing times leading to the worst inrush and overvoltages were the same in the FDNEs and the complete system, but for the Thevenin Equivalent, due to the lack of accuracy, the worst conditions occurred for other times. Deterministic simulations were executed for the switching times that led to the worst condition for each equivalent.

Figures 3.11 and 3.12 present the overvoltage and inrush currents, respectively, following the transformer energization for the complete system and the 28/40 poles FDNE reduced system. It is visible that the approximation up to 500 Hz is sufficient for reproducing the energization waveforms.

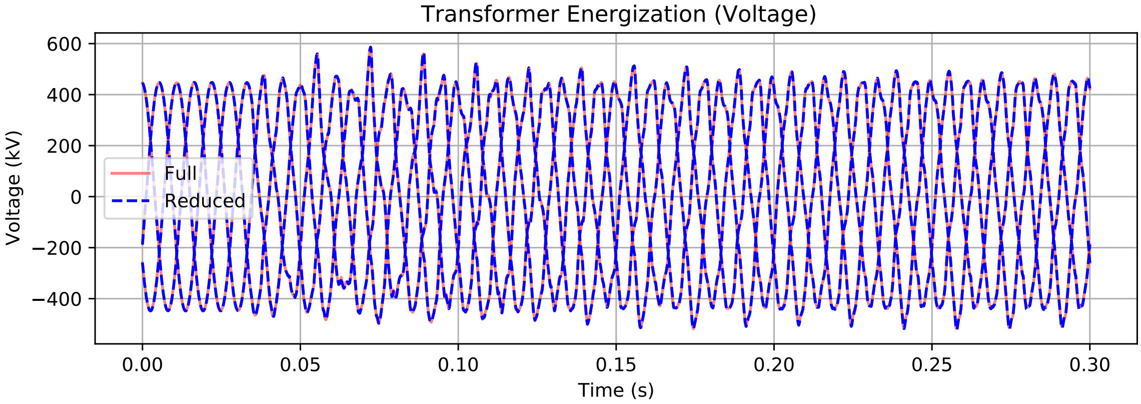


Figure 3.11: Overvoltages for the complete system and the 26/40 poles FDNE. (Simulation # 106).

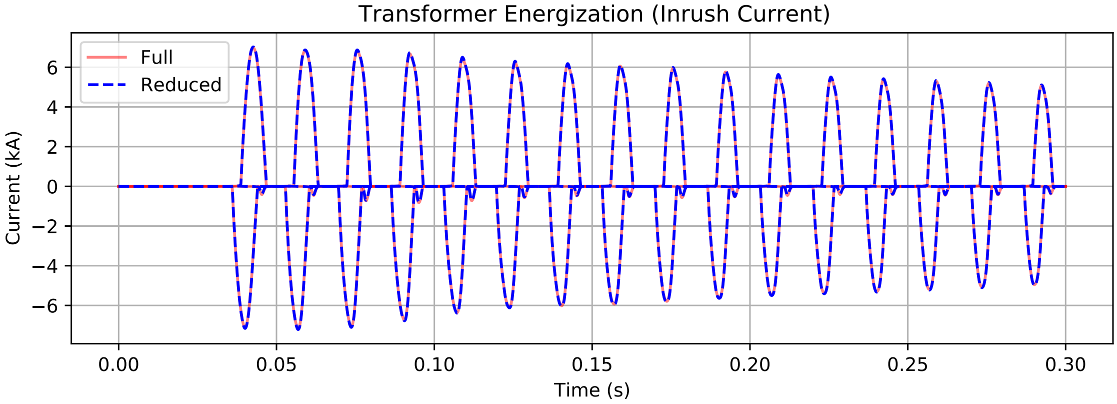


Figure 3.12: Inrush currents for the complete system and the 26/40 poles FDNE. (Simulation # 119).

For the sake of comparison, Figures 3.13 and 3.14 illustrate the results for the Thevenin Equivalent. It is clear that this case can lead to wrong results since a maximum error of about 301 kV was observed in the line to ground voltage, or approximately 0.73 pu, and a maximum error of 1.8 kA in the inrush current, corresponding to approximately 0.93 pu. The errors are highlighted in Figures 3.15a and 3.15b.

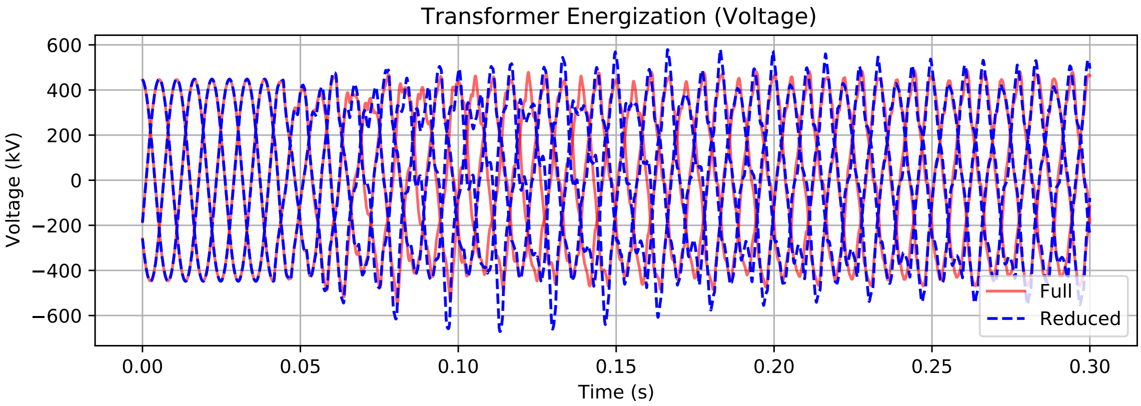


Figure 3.13: Overvoltages for the complete system and the Thevenin Equivalent. (Simulation # 182).

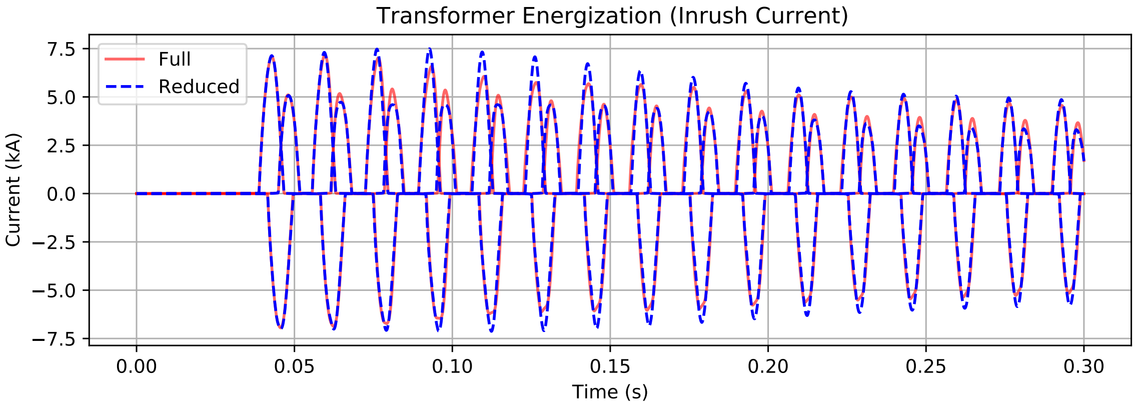
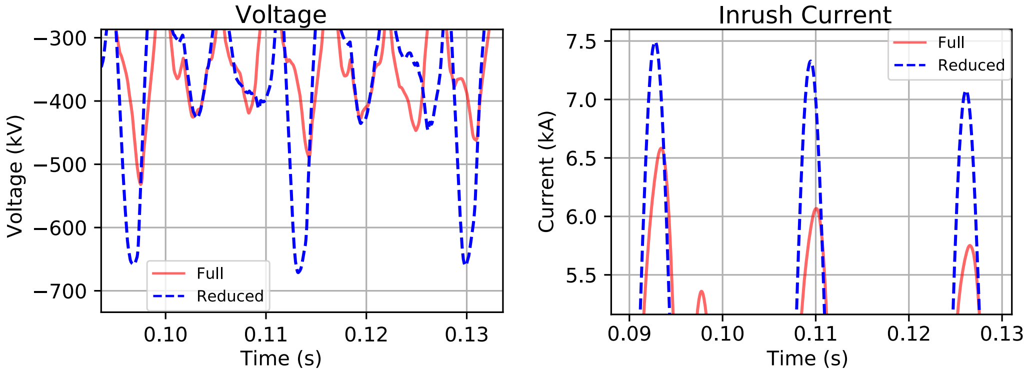


Figure 3.14: Inrush currents for the complete system and the Thevenin Equivalent. (Simulation # 160).

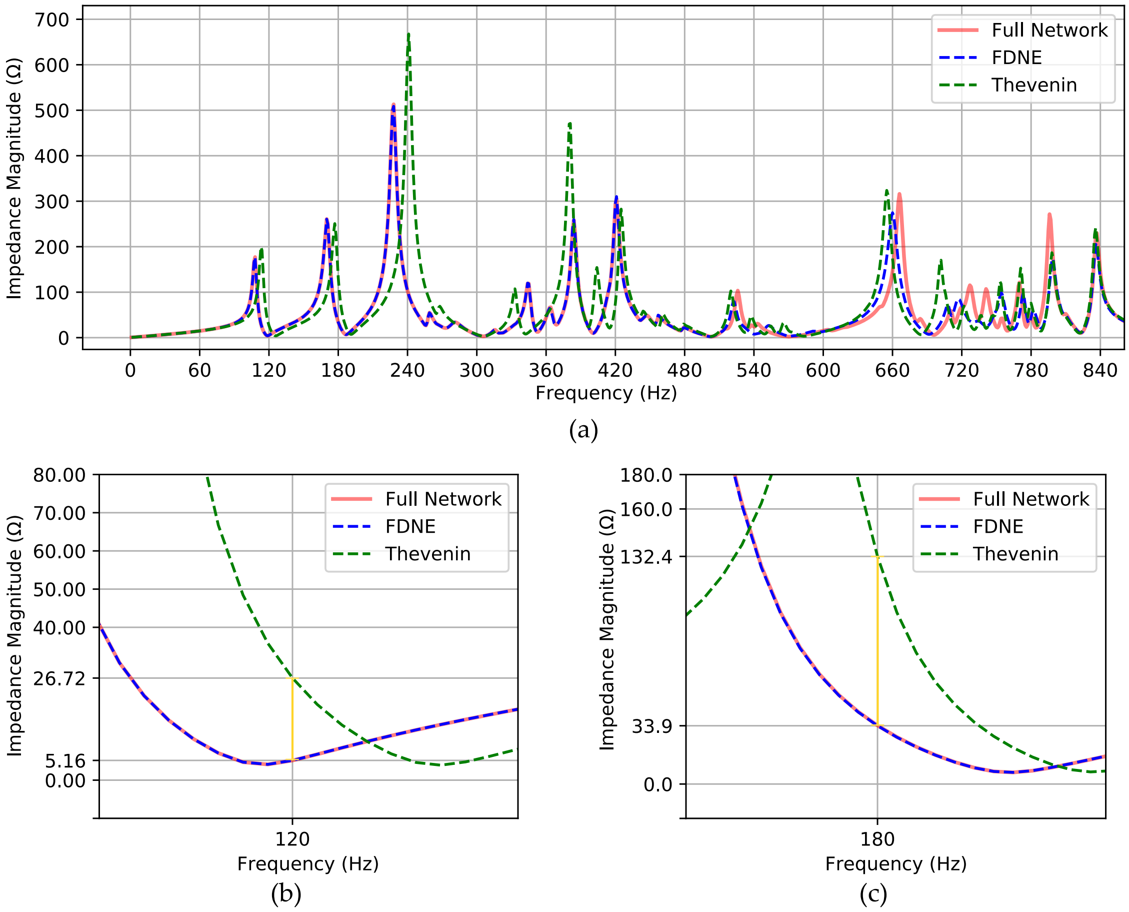


(a)

(b)

Figure 3.15: a) 0.73 pu voltage error and b)0.93 pu current error.

As the Thevenin Equivalent does not take the frequency response into account, such errors are always possible. One way to assess the suitability of the 60 Hz equivalent for a proposed study is to compare the frequency response of the complete and the reduced systems at the bus subjected to the switching event (not the equivalent terminals). Figure 3.16a presents the frequency response seen at the Terminal Rio substation for the i) the full network; ii) the reduced network by Thevenin and iii) the reduced network by the FDNE with 26/40 poles. Figures 3.16b and 3.16c show the approximation around the second and third harmonics, respectively, showing that an error of more than 5 times occurs at 120 Hz and almost 4 times at 180 Hz, what explains the large errors in the time-domain results with the Thevenin Equivalent, as these harmonics are present in the inrush currents. On the other hand, the frequency response of the FDNE system presented negligible discrepancy up to 500 Hz, as this was the upper limit of the approximation.



(a)

(b)

(c)

Figure 3.16: a) Frequency responses of the full network and the reduced systems by Thevenin and FDNE. b) Approximation around 120 Hz and c) Approximation around 180 Hz.

3.5 Performance Evaluation

For evaluating the performance of the implemented FDNE solver, challenging cases have been executed with different numbers of poles and ports. To restrict the assessment to the FDNE component, the simulated systems were composed only by the FDNE connected to a current source, as illustrated in Figure 3.17. The efficient implementation was evaluated in modal and phase coordinates and compared to the default EMTP solver, which is a regular state-space solver that receives the full state matrices. The size of data files was also compared. All cases were run with steady-state initialization, with a time-step of 10 μs and a total time of 1 s. As the purpose of this section is to evaluate performance, no further details regarding the time-domain responses are given. To carry out a performance assessment, it is enough that the case is stable in the time-domain.

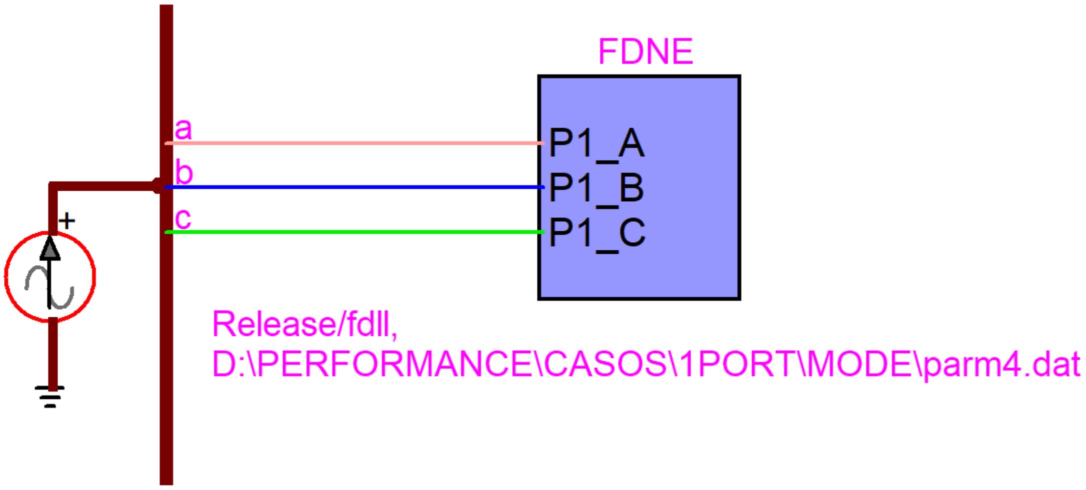


Figure 3.17: Circuit used for performance evaluation.

All cases presented in this section were fitted with admittance parameters. The simulations were executed in a notebook with the following configuration: Intel i7-10610U 1.8GHz - 2.3 GHz processor, with 16 GB of RAM and SSD Hard Disk, running the Windows 10 OS.

Table 3.3 shows the results for the following cases: (i) 1 port, 300 poles; (ii) 2 ports, 400 poles; (iii) 3 ports, 500 poles; (iv) 5 ports, 600 poles and (v) 10 ports, 128 poles. Those high number of ports and poles were purposely chosen to assess the maximum size upon which an FDNE would no longer be feasible from the computational point of view. The execution time shown is an average of over five simulations. The modal domain cases have the same number of poles for both sequences. The 10-port case was executed only in modal coordinates.

Table 3.3: Performance Results.

	Case	Total Time (s)	Data File Size
1 Port, 300 poles	Default	61	17 Mb
	Phase	1	61 kb
	Modal	0.9	28 kb
2 Port, 400 poles	Default	180	130 Mb
	Phase	1.6	298 kb
	Modal	1.2	80 kb
3 Port, 500 poles	Default	-	450 Mb
	Phase	2.2	800 kb
	Modal	1.6	164 kb
5 Port, 600 poles	Default	-	1.9 Gb
	Phase	5.3	2.7 Mb
	Modal	4.1	480 kb
10 Port, 128 poles	Default	-	-
	Phase	-	-
	Modal	4.7	421 kb

The **A** matrix order for case (i) is 900 and 300 for phase and modal domains, respectively. Since the modal state space has to be solved three times, one for each sequence, both domains had similar performances, with the modal domain slightly faster. This difference tends to increase with the size of the FDNE. The speed gain over the default component is about 60 times, approximately the same result obtained in the statistical analysis presented in the last section. The speed gain also tends to increase with the size of the FDNE, as shown in case ii, in which a gain of at least 90 times was observed. Also, the difference in data file sizes becomes relevant, due to the storage of the large and sparse A matrix. For the next cases, the default solver is no longer computationally feasible due to both data file size (up to Gigas) and execution time (hours).

The performance difference between the phase and modal implementations was not substantial in any performed simulation and all cases have been executed in acceptable times, even the 600 poles/5 ports case. It is then possible to conclude that offline time-domain computational burden is not an issue with the proposed FDNE implementation using the current commercially available personal computers.

Chapter 4

FDNE Implementation in RTDS and OPAL-RT simulators

This chapter introduces the developed FDNE models for the RTDS and OPAL-RT real-time simulators. Firstly, the model validation methodology is presented, which is based on the harmonic injection over the frequency range. Then, the FDNE implementation for each real-time hardware is presented. In order to make smaller time-steps possible, a strategy for splitting the FDNE into different components which can be executed in parallel is presented in the subsection Decoupled FDNE. Finally, the proposed Frequency Warping Correction algorithm is presented, which was developed in order to reduce the error associated with the discretization process.

4.1 Model Validation Methodology

Instead of evaluating the FDNE accuracy comparing switching transients waveforms against the complete network as done in the previous chapter, a more general approach will be described here, in which the frequency response is obtained from a time-domain simulation. The advantage is that the whole frequency spectrum can be precisely evaluated, instead of the specific harmonics that compose the switching waveform. The disadvantage is that several simulations must be performed, one for each port of the FDNE. This validation methodology is based on a custom component that performs a current injection composed of the summation of unitary sinusoidal waves over the chosen frequency range, as shown in Equation 4.1. The default range is set 1 to 2000Hz. The developed real-time FDNE models were evaluated with this methodology.

$$I(t) = \sum_{k=1}^{2000} \sin(2\pi k \times t) \tag{4.1}$$

This model is coupled to the FDNE in a time-domain simulation. As the current injections are unitary, the voltage values measured at each port represent its associated impedance. To obtain the frequency response, the voltage values are sent to a Fast Fourier Transform (FFT) routine programmed in Python.

Figure 4.1 shows a circuit containing a two-port equivalent and the harmonic injection component, built in the RSCAD environment used to create the circuits simulated in the RTDS simulator.

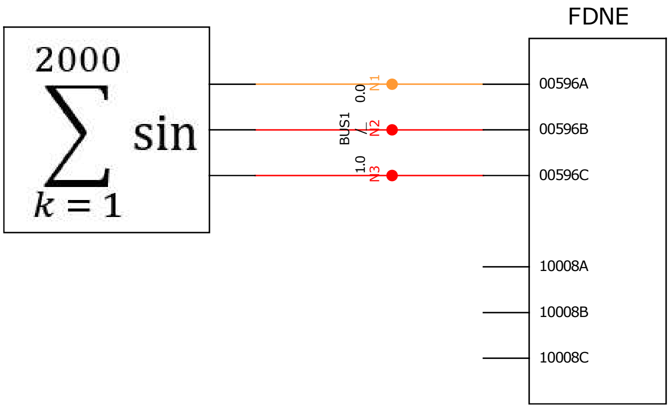


Figure 4.1: The harmonic injection (left) and FDNE component (right).

A typical resulting waveform of the harmonic injection component is shown in Figure 4.2. The period of this waveform is 1 s, as the lowest injected frequency is 1Hz. Calculating the FFT over this signal gives the frequency response associated with the respective bus where the voltage was measured.

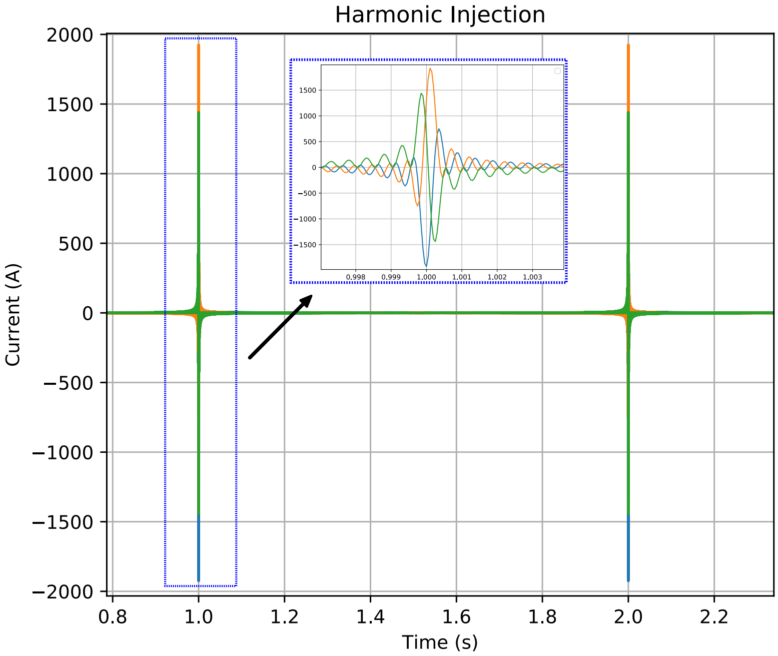


Figure 4.2: Resulting waveform of the harmonic injection component.

So instead of applying faults for generating harmonics and comparing the time-domain waveforms of the full and the equivalent networks, the harmonic injection methodology ensures an accurate comparison of the impedance associated with every frequency within the fitted range.

4.2 RTDS Simulator

The Novacor RTDS platform, shown in Figure 4.3, is composed of a proprietary motherboard with a POWER-8 RISC processor containing 10 cores running at 3.5 GHz.

Figure 4.3: Novacor from RTDS.

To demonstrate the FDNE real-time feasibility, a three-port equivalent will be fitted with 128 poles representing part of the Brazilian North System, whose simplified one-line diagram is presented in Figure 4.4. The original network was modeled for the pre-operational studies of the 450 MVA/500-230 kV three winding transformer in the Miranda II substation. The fundamental-frequency short-circuit equivalents were calculated based on the short circuit case of December 2010. The three ports of the FDNE are associated with the following buses: Miranda II (500kV), São Luis II (500 kV) and Peritoró (230 kV) represented in Figure 4.5. The external network is represented by red buses, while the internal network has black buses. The frequency scan is done with ATP, as explained in Appendix C, in order to provide the Vector Fitting routine the frequency responses associated with the three buses, for both sequence modes. The full ATP circuit has 491 nodes and 862 branches.

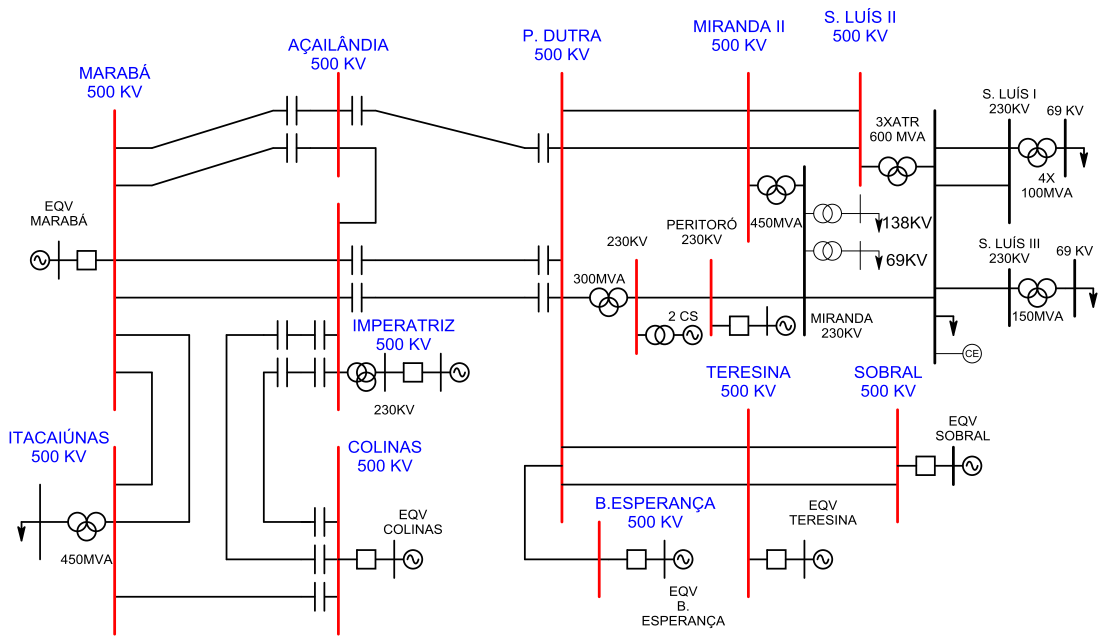


Figure 4.4: Simplified one-line diagram of part of the North Brazilian System.

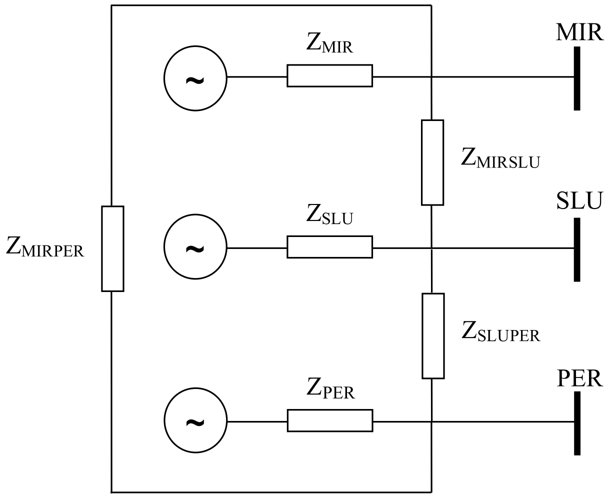


Figure 4.5: Three-port FDNE.

After the VF procedure and the passivity checking, the program automatically creates three files used to generate the user-customized model (extensions .def, .h and .c) in the RSCAD. These files have to be placed in the proper user components folder of RSCAD. This case had passivity violations, so the passivity enforcement post-processing routine was applied. Then, the CBuilder tool provided in the RSCAD suite is required to compile the model before running it in real-time. After the compilation, the component appears in the RSCAD library to be instantiated by the user. The FDNE will be validated following the steps presented in section 4.1. The simulated circuit is shown in Figure 4.6. The following results make use of the Frequency Warping Correction algorithm, which consists of a frequency shift in the frequency response given to the vector fitting algorithm, which will be presented in Section 4.5.

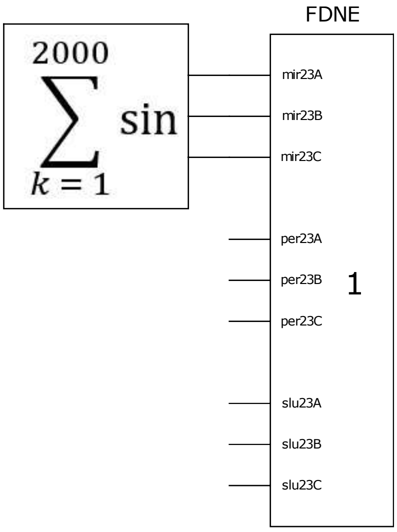


Figure 4.6: FDNE validation circuit.

Figures 4.7 and 4.8 show the module and phase comparison between the external network frequency and the FDNE frequency responses, for the positive sequence impedance associated with the Miranda bus. All results presented absolute and RMS error below 0.1 Ω, due to the high number of poles employed and the warping error correction algorithm. Without this correction algorithm, the discretization warping error would lead to an increasing deviation that would be noticeable after 1000 Hz when executing the simulation with a 50 μs time-step, as will be shown in Section4.5.

Table 4.1 presents the numerical values of the frequency responses of the FDNE, the frequency domain response (Vector Fitting) and the original external network. The FDNE is expected to match the Vector Fitting results since the warping correction is being used. As the 128 poles provided a good representation of the network, the FDNE frequency response matched with the network response as well. The plots related to the other results are shown in the Appendix E.

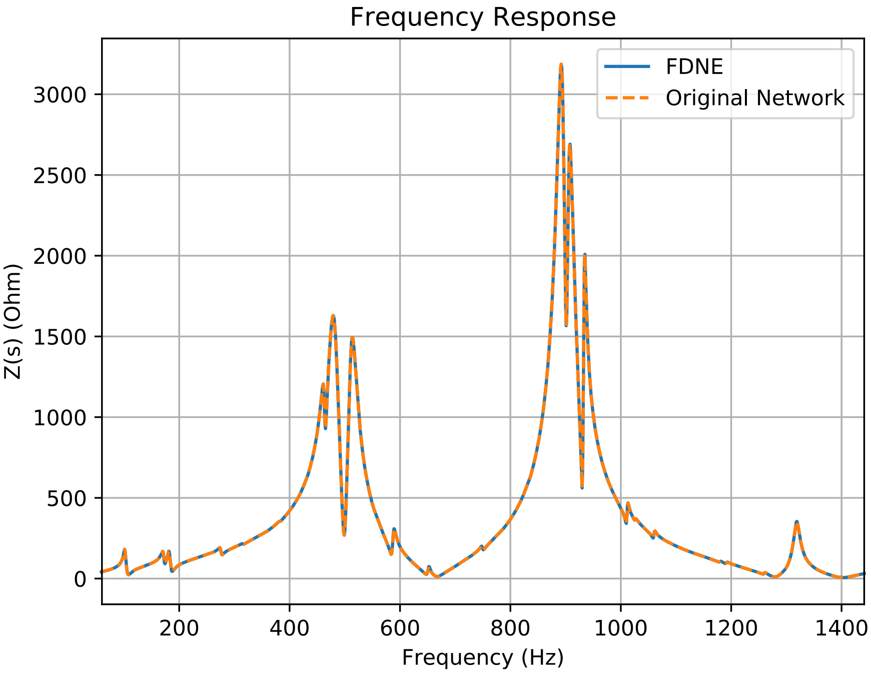


Figure 4.7: Z_{11} positive sequence amplitude frequency response (Original Network and FDNE).

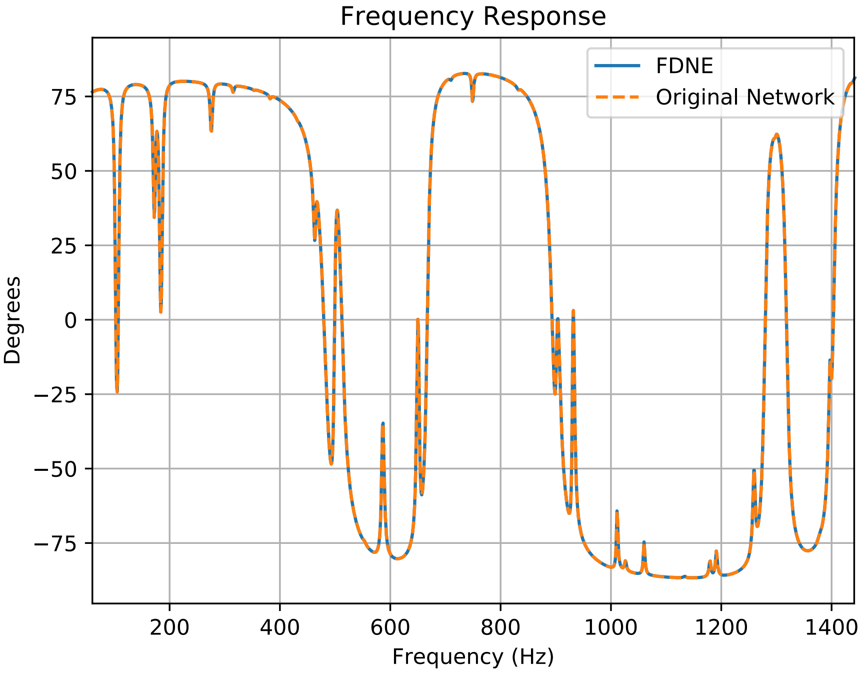


Figure 4.8: Z_{11} positive sequence phase frequency response (Original Network and FDNE).

Table 4.1: Z_{11} positive sequence frequency response values (Original Network, Vector Fitting and FDNE).

	Z_{11}					
Freq (Hz)	FDNE		Vector Fitting		Original Network	
	Mod (Ω)	Phase (°)	Mod (Ω)	Phase (°)	Mod (Ω)	Phase (°)
60	42.21	76.48	42.21	76.48	42.21	76.48
120	53.83	76.96	53.83	76.96	53.83	76.96
180	158.64	52.33	158.63	52.33	158.63	52.33
240	136.03	79.97	136.02	79.97	136.02	79.97
300	193.86	79.10	193.85	79.10	193.85	79.10
360	295.47	77.00	295.45	77.00	295.45	77.00
420	532.17	70.38	532.14	70.38	532.14	70.38
480	1621.51	-2.32	1621.49	-2.32	1621.49	-2.32
540	611.53	-69.03	611.50	-69.03	611.50	-69.03
600	197.20	-78.67	197.19	-78.67	197.19	-78.67
660	26.20	-53.74	26.20	-53.74	26.20	-53.74
720	119.06	82.23	119.06	82.23	119.06	82.23
780	280.83	82.32	280.82	82.32	280.82	82.32
840	678.99	76.71	678.96	76.71	678.96	76.71
900	1704.11	-21.84	1704.08	-21.84	1704.08	-21.84
960	806.93	-77.83	806.89	-77.83	806.89	-77.83
1020	392.63	-83.23	392.61	-83.23	392.61	-83.23
1080	236.95	-86.03	236.94	-86.03	236.94	-86.03
1140	148.34	-86.61	148.33	-86.61	148.33	-86.61
1200	91.74	-85.63	91.73	-85.63	91.73	-85.63
1260	34.35	-50.57	34.35	-50.57	34.35	-50.57

The above results are obtained with a 50 μs time-step. The minimum achievable time-step for the 3-port, 128 poles FDNE was 13 μs. Other results for some different numbers of ports and poles are presented in Table 4.2. These values were obtained in a trial-and-error approach, decreasing the time-step until the error of "time-step overflow" occurs.

Table 4.2: Minimum time-step for different number of ports and poles (in μs).

		Poles		
		32	64	128
Ports	1	4	5	5
	2	5	6	8
	3	7	9	13

It is worth noting that these values do not exactly represent the computation time of the FDNE component. Instead, they represent the FDNE computation time plus the network solution time. Because in these examples the network is small, it can be assumed that the FDNE computation time is close to these values.

With the purpose of estimating how many FDNEs can be allocated to a single processor, a set of simulations with multiple instantiations of the same component was performed with a typical time-step of 50 μs, ensuring that only one core of the POWER8 processor is used. This estimation is presented in Table 4.3.

Table 4.3: Maximum number of FDNEs for a 50 μs time-step simulation.

Case	Number of components
2 port, 64 poles	18
2 port, 128 poles	10
3 port, 64 poles	9
3 port, 128 poles	5

4.3 OPAL-RT Simulator

The OP5707 OPAL-RT platform, shown in Figure 4.9, is composed of up to two Intel Xeon E5 3GHz processors with 16 cores and a VC707 FPGA board. Unlike RTDS which produces its own boards, the OPAL-RT is based on commercial off-the-shelf hardware.

The OP5707 is connected via an ethernet interface to a host computer, in which the EMT simulations are programmed in the RT-LAB or the HYPERSIM tool, the latter being the one used in this work to program the FDNE component.

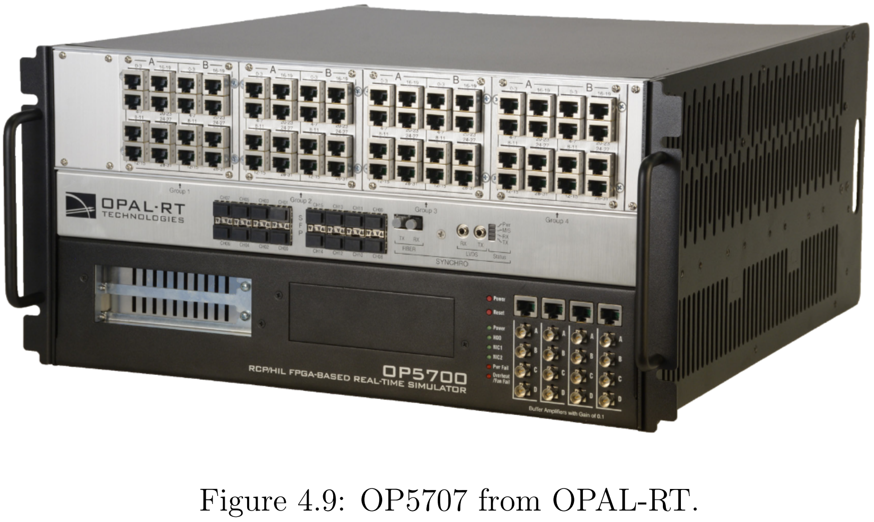


Figure 4.9: OP5707 from OPAL-RT.

The HYPERSIM's network solution is similar to the RTDS solution, regarding the Nodal Analysis Method and the representation of electrical components as Norton Equivalents. Consequently, except for some template differences, the code used to program the customized model is the same. The automatic code generation for the FDNE component was also implemented for HYPERSIM.

To compare both platforms, the same network was reduced and validated with the same harmonic injection methodology.

Figure 4.10 illustrates the circuit simulated in HYPERSIM, containing the harmonic injection component, the 3-port FDNE component and a Point-on-Wave component required for acquisition synchronization in the ScopeView tool. This component is mandatory in every model with network components.

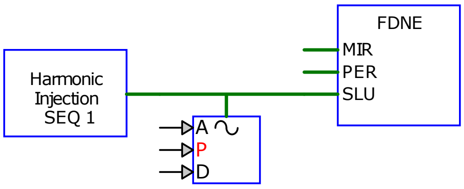
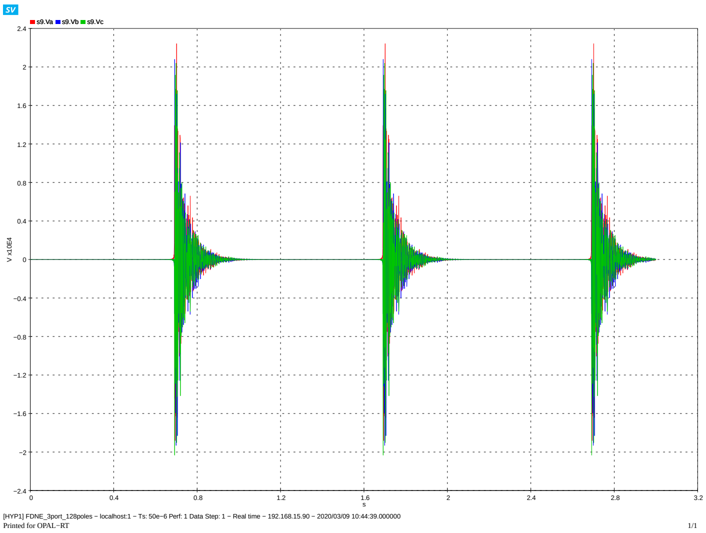


Figure 4.10: FDNE circuit test in HYPERSIM.

The voltage values measured at the bus Miranda when the harmonic injection is applied to the São Luis bus are shown in Figure 4.11. The impedance frequency response shown in Figure 4.12 is obtained after applying the FFT to the voltage signal. The FDNE modeled in the OPAL-RT presented the same accuracy obtained in the RTDS, with absolute and RMS errors below 0.1 Ω for all frequency responses.



[HYP1] FDNE_3port_128poles − localhost:1 − Ts: 50e−6 Perf: 1 Data Step: 1 − Real time − 192.168.15.90 − 2020/03/09 10:44:39.000000
Printed for OPAL−RT
1/1

Figure 4.11: Voltage waveform measured at the FDNE bus subjected to harmonic injection.

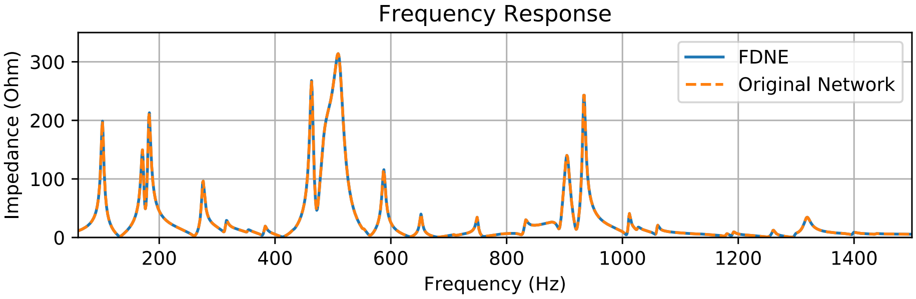


Figure 4.12: Impedance frequency response.

The HyperView tool available in HYPERSIM is used to monitor and assess the computational performance of the running simulation. It can be used to obtain the time spent by each core to perform all the operations within a time-step. Tables 4.4, 4.5 and 4.6 presents the HyperView diagnosis for simulations with 1, 2 and 3 FDNE components, respectively. The 'exec' column indicates the execution time of the present time-step. It changes continuously during the simulation. 'Exec max' indicates the maximum execution time observed since the beginning of the simulation. 'Remain' and 'Remain min' are the remaining times in comparison with the time-step duration. 'Comm' and 'Comm Max' are the communication overheads between cores, which are very small for this case as only one core is being used. 'Sim' and 'Sim Max' refers to the actual time the time-step took. Ideally, this value is always the same as the time-step. 'Acq/Par' indicates the time spent for sampling the results to be plotted. This value increases with the number of variables being monitored. The HYPERSIM suite works with the stretched time-step concept, which means that if it cannot achieve real-time execution, the simulation keeps on with the best achievable performance, unlike the RTDS that raises the "time-step overflow" error when the simulator cannot achieve real-time execution. The last column 'Stretched Step' indicates how many time-steps were not executed in real-time.

Table 4.4: HyperView results for the case with one FDNE.

Exec	Exec Max	Remain	Remain Min	Comm	Comm Max	Sim	Sim Max	Acq Par	Streched Step
22.5	23.34	27.45	26.56	0.02	50.04	50.04	50.11	0.26	0

Table 4.5: HyperView results for the case with two FDNEs.

Exec	Exec Max	Remain	Remain Min	Comm	Comm Max	Sim	Sim Max	Acq Par	Streched Step
44.96	45.9	4.96	3.99	0.01	0.05	50.01	50.07	0.23	0

Table 4.6: HyperView results for the case with three FDNEs.

Exec	Exec Max	Remain	Remain Min	Comm	Comm Max	Sim	Sim Max	Acq Par	Streched Step
67.56	69.45	0.18	0.17	0.02	0.05	67.77	69.77	0.2	431115

Since in this case there are only 9 nodes, the time spent with the network solution is negligible in comparison with the FDNE computing, what makes the minimum time-step increase almost linearly with the number of instantiated FDNE components.

Table 4.7 presents the execution time performance for the implemented FDNE.

Table 4.7: Time-step performance for 3-port, 128 poles FDNE.

Number of FDNEs	Execution (μs)
1	22.5
2	44.96
3	67.56

When instantiating three FDNEs components, the execution time exceeds the 50 μs time-step. Table 4.6 shows that the case is running with a high number of stretched time-steps, thus it is not running in real-time.

The minimum achievable time-step (for which no stretched time-step occurred) for the 3-port, 128 poles FDNE was 25 μs, almost double than RTDS. However, for small FDNEs, the OPAL-RT presented better performance. Table 4.8 presents the execution time and the minimum time-step for FDNEs with different numbers of ports and poles. The Execution Time column refers to the time measured in HyperView, while the Minimum Δt column represents the minimum time-step for execution without any occurrence of stretched step.

Table 4.8: Execution time and Minimum time-steps for FDNEs with different number of poles and ports.

Ports	Poles	Execution Time (μs)	Minimum Δt (μs)
1	32	1.72	3
	64	3.14	4
	128	4.98	6
2	32	3.58	5
	64	6.5	8
	128	12.28	14
3	32	6.45	8
	64	12.5	13
	128	23.67	25

For the sake of comparison, 4.9 presents the same results that before, but including a column for the built-in State-Space solver, which solves the system using the full state matrices, and the FPGA implementation that will be shown in the next chapter. Places with an asterisk indicate that the case could not be executed due to memory limitation.

Table 4.9: Execution time and Minimum time-steps for FDNEs with different number of poles and ports.

Ports	Poles	Built-in SS Component Δt (μs)	Custom Component Δt (μs)	FPGA Δt (μs)
1	32	3	3	3
	64	5	4	4
	128	9	6	4
2	32	6	5	4
	64	13	8	4
	128	*	14	4
3	32	13	8	4
	64	*	13	6
	128	*	25	7

Table 4.3 presents the maximum number of FDNEs components which can be instantiated in a simulation with 50 μs time-step. For all cases, only one core of the Intel Xeon E5 processor is used.

Table 4.10: Maximum number of FDNEs for a 50 μs time-step simulation.

Case	Number of components
2 port, 64 poles	7
2 port, 128 poles	4
3 port, 64 poles	4
3 port, 128 poles	2

The real-time results presented so far indicates that a parallelization scheme would benefit the FDNE simulation in real-time, as in both tested platforms the maximum FDNE sizes for typical time-steps are in the range of 3 to 4 ports, with hundreds of poles.

4.4 Decoupled FDNE

The FDNE implementation presented above exploits sparsity, but not parallelism, which is crucial to increase runtime performance and, consequently, reduce the integration time-step. This is possible because the rational model of the FDNE is composed of fractional functions, and thus can be easily split. This option can be especially interesting for cases in which, due to the high number of ports and/or poles, the typical integration step of 50 μs becomes unfeasible. FDNE parallelism is based on the partitioning of the state-space solver component, so that each part is associated with a different processor, thus allowing real-time simulation with smaller time-steps. The implemented algorithm is based on the reference [58]. In the following exposition, the modal sequence was not specified, as the expressions are identical for positive, negative and zero sequences. For a system with p ports, the equation 4.2 presents the nodal impedance transfer function associated with each fitted impedance branch i, j.

$$Z_{ij}(s) = d_{ij} + \sum_{k=1}^{n} \frac{c_{ijk}}{s - a_k} \quad i, j = 1, 2, \ldots, p \tag{4.2}$$

where, as previously, c_{ijk} is the kth residue associated with port i, j, and p is the number of ports. The a_k coefficients are poles (real or complex). If the synthesized transfer function has impedance dimension, $Z(s) = \frac{V(s)}{I(s)}$, so the state-space system that is employed for its time-domain solution has current inputs and voltage outputs, corresponding to a multi-port Thevenin Equivalent. However, network solution algorithms usually expect to receive a current injection from custom models and to stamp conductances in its global matrix. Therefore, converting from Thevenin to Norton equivalent is required.

The equation 4.2, being the sum of fractional functions, can be split into several components. The equation 4.3 presents a two-component partitioning, each of them allocating half of the poles.

$$Z_{ij}(s) = d_{ij} + \sum_{k=1}^{n/2} \frac{c_{ijk}}{s - a_k} + \sum_{k=n/2+1}^{n} \frac{c_{ijk}}{s - a_k} \quad i, j = 1, 2, \ldots, p \tag{4.3}$$

The term d_{ij} can be allocated to any of the components. The equations 4.4 and 4.5 splits the portions called 'a' and 'b' of the original nodal impedance.

$$Z^a_{ij}(s) = d_{ij} + \sum_{k=1}^{n/2} \frac{c_{ijk}}{s - a_k} \quad i, j = 1, 2, \ldots, p \tag{4.4}$$

$$Z^b_{ij}(s) = \sum_{k=n/2+1}^{n} \frac{c_{ijk}}{s - a_k} \quad i, j = 1, 2, \ldots, p \tag{4.5}$$

Figures 4.13 and 4.14 show a two-component partitioning related to a two-port Thevenin and Norton Equivalents, respectively.

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} + \begin{bmatrix} v_{th_1} \\ v_{th_2} \end{bmatrix} \xrightarrow{\qquad\qquad} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} + \begin{bmatrix} v^a_{th_1} + v^b_{th_1} \\ v^a_{th_2} + v^b_{th_2} \end{bmatrix}$$

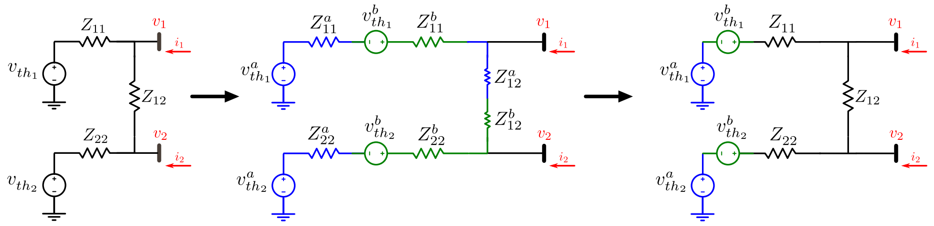


Figure 4.13: Multi-port Thevenin Equivalent Partitioning.

$$\begin{bmatrix} i_1 + i_{N_1} \\ i_2 + i_{N_2} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \xrightarrow{\qquad\qquad} \begin{bmatrix} i_1 + i^a_{N_1} + i^b_{N_1} \\ i_2 + i^a_{N_2} + i^b_{N_2} \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

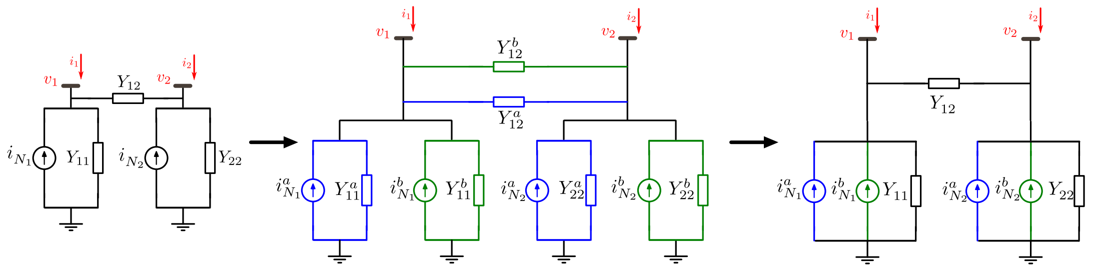


Figure 4.14: Multi-port Norton Equivalent Partitioning.

In both figures, blue components refer to the 'a' portion, while green components refer to the other. This illustrative case describes a sectioning in only two components. However, the partitioning can be extended up to the number of poles, a limit situation in which each component would be associated with one single pole. When obtaining the state space system, the modal canonical form was used [72], in which the diagonal state matrix **A** is composed of blocks of unit size corresponding to the real poles and 2x2 blocks corresponding to each pair of complex conjugate poles. This formulation is suggested by the author of the method so that matrix **A** has only real elements.

Next, the state equations for an example case of a two-port equivalent fitted with six poles (two real and four complexes) and sectioned into three parallel components will be illustrated, to demonstrate how the state space system resulting from Vector Fitting can be partitioned into components whose solutions are independent of each other, and therefore parallel. Real poles 1 and 2 will be allocated to the first core, the first conjugate pair will be in the second core and the second conjugate pair in the third core.

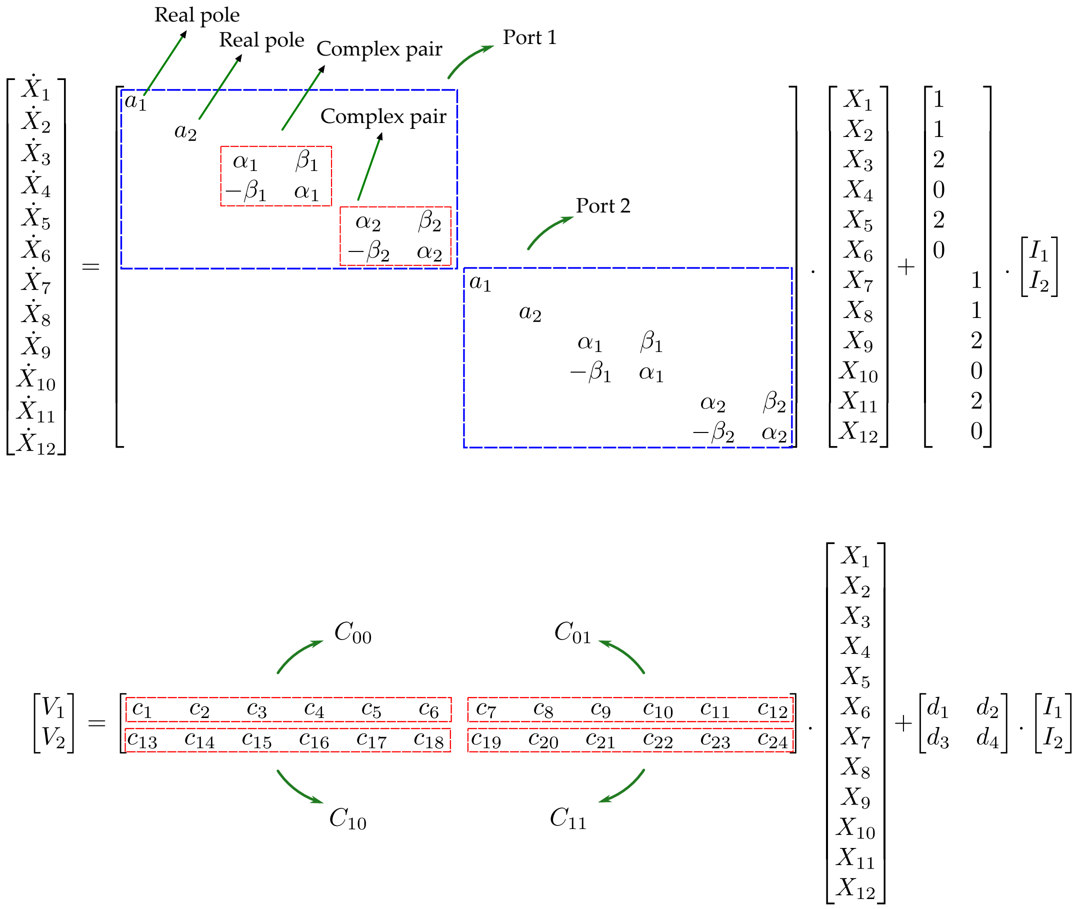
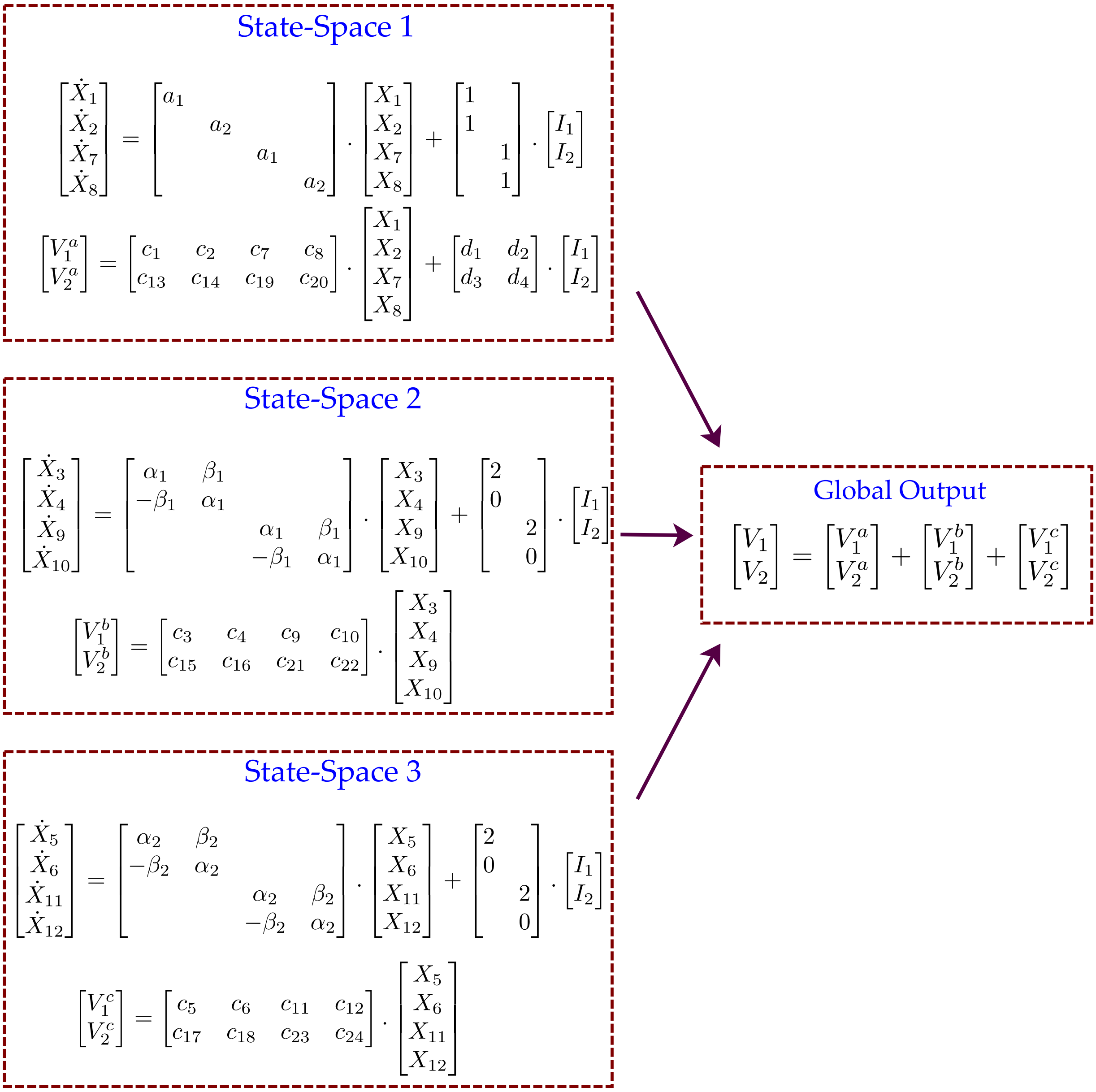


Figure 4.15: Full system state-space representation.

The number of states is $(p \text{ x } n)$, where p is the number of ports and n is the number of poles. Each blue block of the state matrix **A** represents the total set of poles in the system. There is one block for each port. The blocks are identical because this implementation of vector fitting employs the same set of poles for all elements of the nodal impedance matrix. Matrix **B** is normalized so that the components associated with real poles are 1, the components associated with the first elements of complex poles are 2, and the second elements are 0.

The **C** matrix has $(p^2$ x $n)$ components, as it is composed of n matrices $(p$ x $p)$. The blocks associated with each component of the 2x2 impedance matrix are highlighted in the image. For balanced systems, $C_{01} = C_{10}$. The components C_1 and C_{13} are associated with pole 1, C_2 and C_{14} to pole 2, and so respectively. When splitting the system, the state space matrices must be rearranged in order to decouple the poles related to each decoupled portion.

Figure 4.16 illustrates the sectioning of the previous state space into three independent state spaces. The first component is associated with poles 1 and 2, the second with poles 3 and 4, and the third with poles 5 and 6. A careful examination must be done in order to group the correct state variables to their respective poles. In this case, real poles 1 and 2 are associated with states X_1 and X_2 in the first port and X_7 and X_8 in the second port. The same procedure is applied to the **B** and **C** matrix, ensuring that they are re-organized in the same order of the states, per example, 1,2,7,8, for the first component.



State-Space 1

$$\begin{bmatrix}\dot{X}_1\\ \dot{X}_2\\ \dot{X}_7\\ \dot{X}_8\end{bmatrix} = \begin{bmatrix}a_1 & & & \\ & a_2 & & \\ & & a_1 & \\ & & & a_2\end{bmatrix} \cdot \begin{bmatrix}X_1\\ X_2\\ X_7\\ X_8\end{bmatrix} + \begin{bmatrix}1 & \\ 1 & \\ & 1\\ & 1\end{bmatrix} \cdot \begin{bmatrix}I_1\\ I_2\end{bmatrix}$$

$$\begin{bmatrix}V_1^a\\ V_2^a\end{bmatrix} = \begin{bmatrix}c_1 & c_2 & c_7 & c_8\\ c_{13} & c_{14} & c_{19} & c_{20}\end{bmatrix} \cdot \begin{bmatrix}X_1\\ X_2\\ X_7\\ X_8\end{bmatrix} + \begin{bmatrix}d_1 & d_2\\ d_3 & d_4\end{bmatrix} \cdot \begin{bmatrix}I_1\\ I_2\end{bmatrix}$$

State-Space 2

$$\begin{bmatrix}\dot{X}_3\\ \dot{X}_4\\ \dot{X}_9\\ \dot{X}_{10}\end{bmatrix} = \begin{bmatrix}\alpha_1 & \beta_1 & & \\ -\beta_1 & \alpha_1 & & \\ & & \alpha_1 & \beta_1\\ & & -\beta_1 & \alpha_1\end{bmatrix} \cdot \begin{bmatrix}X_3\\ X_4\\ X_9\\ X_{10}\end{bmatrix} + \begin{bmatrix}2 & \\ 0 & \\ & 2\\ & 0\end{bmatrix} \cdot \begin{bmatrix}I_1\\ I_2\end{bmatrix}$$

$$\begin{bmatrix}V_1^b\\ V_2^b\end{bmatrix} = \begin{bmatrix}c_3 & c_4 & c_9 & c_{10}\\ c_{15} & c_{16} & c_{21} & c_{22}\end{bmatrix} \cdot \begin{bmatrix}X_3\\ X_4\\ X_9\\ X_{10}\end{bmatrix}$$

State-Space 3

$$\begin{bmatrix}\dot{X}_5\\ \dot{X}_6\\ \dot{X}_{11}\\ \dot{X}_{12}\end{bmatrix} = \begin{bmatrix}\alpha_2 & \beta_2 & & \\ -\beta_2 & \alpha_2 & & \\ & & \alpha_2 & \beta_2\\ & & -\beta_2 & \alpha_2\end{bmatrix} \cdot \begin{bmatrix}X_5\\ X_6\\ X_{11}\\ X_{12}\end{bmatrix} + \begin{bmatrix}2 & \\ 0 & \\ & 2\\ & 0\end{bmatrix} \cdot \begin{bmatrix}I_1\\ I_2\end{bmatrix}$$

$$\begin{bmatrix}V_1^c\\ V_2^c\end{bmatrix} = \begin{bmatrix}c_5 & c_6 & c_{11} & c_{12}\\ c_{17} & c_{18} & c_{23} & c_{24}\end{bmatrix} \cdot \begin{bmatrix}X_5\\ X_6\\ X_{11}\\ X_{12}\end{bmatrix}$$

Global Output

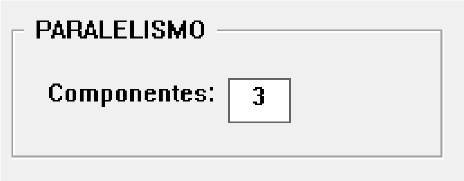
$$\begin{bmatrix}V_1\\ V_2\end{bmatrix} = \begin{bmatrix}V_1^a\\ V_2^a\end{bmatrix} + \begin{bmatrix}V_1^b\\ V_2^b\end{bmatrix} + \begin{bmatrix}V_1^c\\ V_2^c\end{bmatrix}$$

Figure 4.16: State-space sectioning into three components.

The full state-space solution V can be obtained by summing the individual contributions V^a, V^b and V^c, as described in the last equation of Figure 4.16. In practice, this summation is not required because each individual component transforms its solution to a Norton Equivalent, and injects it directly into the global admittance matrix. In the case of admittance fitting, no transformation is necessary.

The matrix $\mathbf{D}$ can be allocated to any component and does not need to be sectioned. In Thevenin Equivalents (when fitting impedance), each component transforms its respective voltage output into current injections, satisfying the Norton equivalent circuit illustrated in Figure 4.13. It should be noted that not every pole sectioning is possible. It is not possible, per example, to partition a pair of complex conjugate poles, since the 2x2 block associated with complex poles is dependent on the adjacent states. The proper error handling was done, in order to prevent the user to choose invalid sectioning.

A routine for automatically generating the parallel FDNE component was developed for the RTDS simulator. To select the number of components, the user must fill the text box in the graphical interface, illustrated in Figure 4.17. If parallelism is not required, the user must keep the default number 1.



PARALELISMO

Componentes: 3

Figure 4.17: Text box for selecting the number of components.

Each component is handled as a different user-customized model. In RTDS cBuilder environment, three files are generated for each customized model with extensions, .h, .c and .def. Each of them must be compiled separately in cbuilder and imported in RSCAD. Figure 4.18 shows the validation circuit for the decoupled model, with the same network reduction shown in the previous sections, but with 400 poles in order to increase the computational burden, sectioned in four components. The blue box contains the four components, as shown in Figure 4.19 with their respective buses joined together.

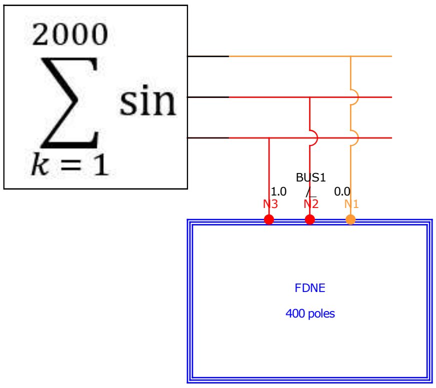


Figure 4.18: Decoupled model validation circuit.

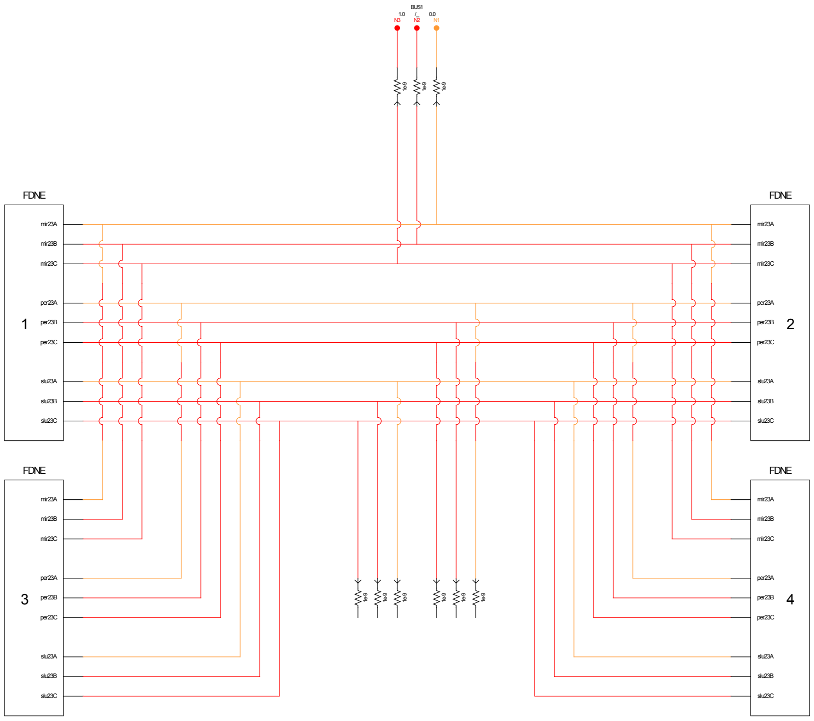


Figure 4.19: Four decoupled FDNE components, each one tasked to one different core.

The minimum time-step obtained for this high-pole FDNE without processor parallelism was 35 μs. Allocating 4 cores, one for each decoupled component, decreased the minimum time-step to 9 μs, as the FDNE component is the main computational load of this case. The numerical results are exactly the same as the full state-space model since the sectioning does not resulted in any mathematical difference in the model.

4.5 Frequency Warping Correction

This section explains the frequency warping correction algorithm used in the previous sections, which ensures that the frequency response obtained from time-domain simulation matches the frequency response resulted from the vector fitting, calculated in the frequency domain.

The trapezoidal numerical method used for the discrete-time model synthesis maps the plane of continuous frequencies into the discrete plane $z = je^{-j\omega_d T}$, where ω is the continuous angular frequency, ω_d the discrete angular frequency and T the integration time-step. For frequencies $\omega_d < 0.25/T$, the discrete frequency ω_d is approximately equal to ω. However, for higher frequencies, the error becomes considerable. The following expression [73, 74] relates both frequencies:

$$\omega_d = \frac{2}{T}\,\mathrm{arctg}\left(\frac{\omega T}{2}\right) \tag{4.6}$$

For the typical integration time-step used in EMT simulation of T = 50 μs, the frequency $f = 1000Hz$ is mapped to $f_d = 991.89Hz$, with an error of approximately 1%. For f = 2000Hz, f_d = 1937.84, with an error of approximately 3%.

Figure 4.20 shows a comparison between a frequency response calculated with the state matrices in the frequency domain $(Z(s) = C(sI - A)^{-1}B + D)$, and the response obtained through the application of the Fast Fourier Transform to the voltage signal measured on the FDNE port, in an EMT simulation performed on the RTDS with integration step of 50 μs, with the harmonic injection methodology explained in Section 4.1.

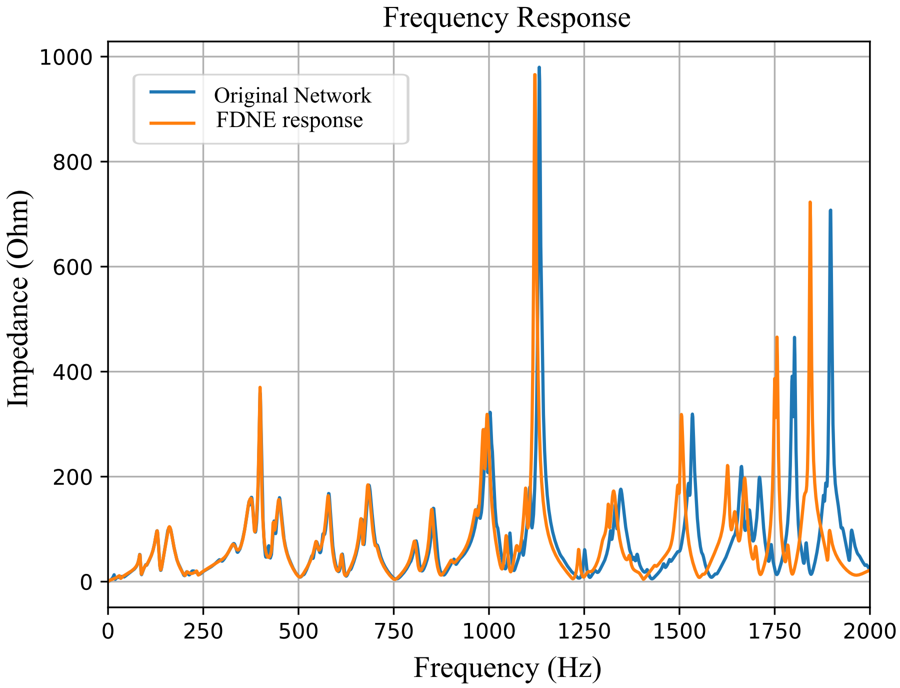


Figure 4.20: Frequency-domain frequency response (blue), and frequency response obtained in time-domain simulation with T=50 μs (orange).

As expected, there is a growing discrepancy after 1000Hz. One can also select one of the resonance peaks to check if the observed discrepancy corresponds to that predicted by equation 4.6. Choosing the existing peak at f = 1802Hz, according to Equation 4.6, it is expected the same peak to be located at $f_d = \frac{1}{\pi T}\,\mathrm{arctg}\,(\pi \times 1802 \times T) = 1756\mathrm{Hz}$ in the frequency response obtained after discretization. The Figure 4.21 shows an approximation around this resonance, confirming the value predicted by the equation 4.6.

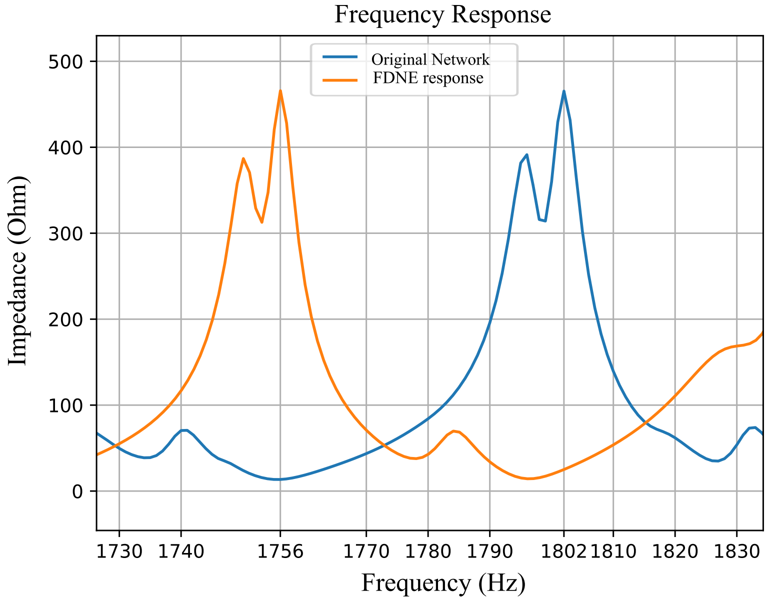


Figure 4.21: Frequency warping error at f=1802Hz for T=50 μs.

A more adequate performance is obtained for this frequency range decreasing the integration time-step to T = 10 μs, as illustrated in Figure 4.22. Looking to the same point f = 1802Hz, the resonance in the response obtained after the discretization is expected to be located in $f_d = \frac{1}{\pi T} \operatorname{arctg}(\pi \times 1802 \times T) = 1800 Hz$. The Figure 4.22 shows an approximation around this resonance, confirming again the value predicted by the equation 4.6.

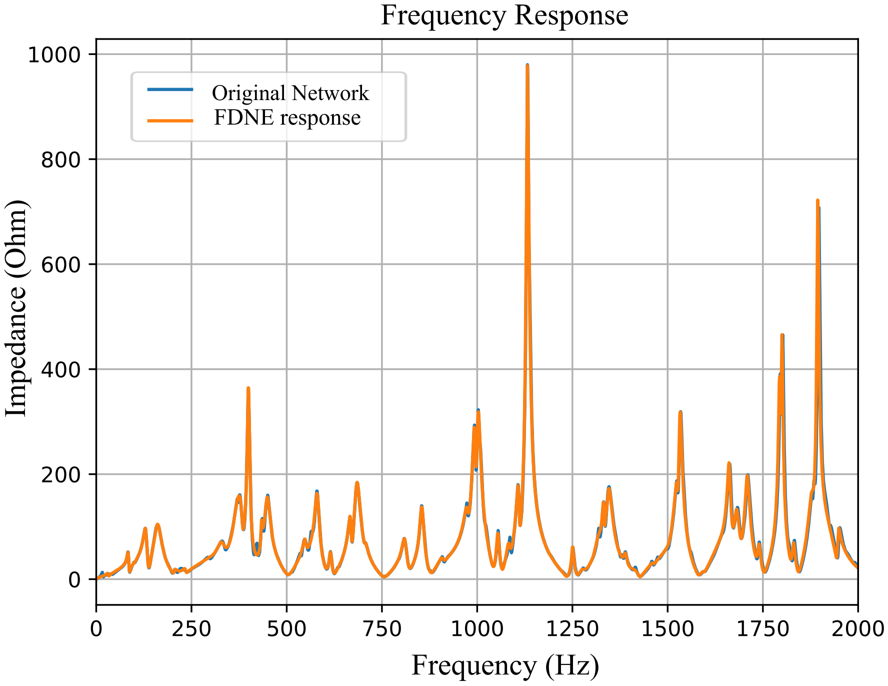


Figure 4.22: Frequency-domain frequency response (blue), and frequency response obtained in time-domain simulation with T=10 μs (orange).

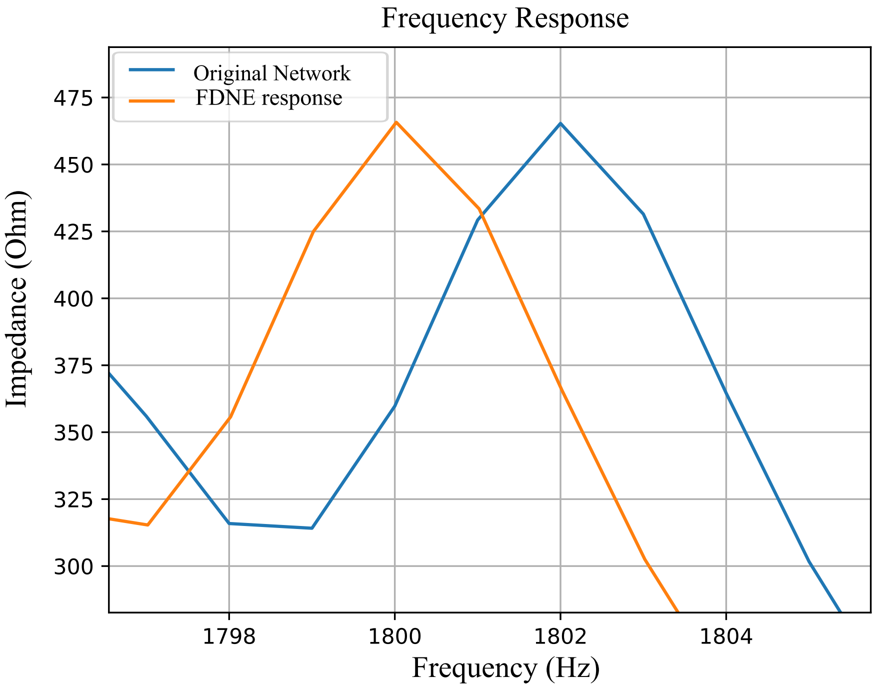


Figure 4.23: Frequency warping error at f=1802Hz for T=10 μs.

It is observed that frequency warping error can be mitigated by decreasing the time-step. However, as such decrease in time-step will not always be possible, an algorithm was developed, which promotes a shift in the frequency axis delivered to the vector fitting routine. This displacement can completely remove the warping error and is only possible because there is an analytical expression for the frequency warping, which depends on the integration time-step.

The idea of the frequency warping correction algorithm is to shift the frequency axis of the input frequency response by a quantity equal to the error associated with each frequency, in the opposite direction, in such a way that after the discretization, the frequency response in time-domain matches the original one. This is done by isolating the frequency ω in 4.6, obtaining

$$\omega = \frac{2}{T}\tan\left(\frac{\omega_d T}{2}\right) \tag{4.7}$$

Considering $\omega' = \omega'_0, \omega'_1, ..., \omega'_{max}$ as the original frequency axis, the warp correction algorithm makes $\omega' = \omega_d$ for each frequency and calculates ω with Eq. 4.7, as shown in the following pseudo-code

```
Result: Shifted frequency axis.
for $k = 1;\ k <= k_{max};$ do
    $\omega_k = \frac{2}{T}\tan(\frac{T}{2} \times \omega'_k)$
    $k = k + 1$
end
```

Figure 4.24 shows a superposition of the original external network frequency response (obtained with the frequency scan approach, which is a phasor solution - with no discretization error), the original FDNE response, i.e, the one obtained in time-domain simulation (harmonic injection plus FFT) and the shifted input frequency response calculated in the frequency domain ($Z(s) = C(sI - A)^{-1}B + D$). The shifted curve (in green color), after discretization, will deliver a time-domain response that matches perfectly with the blue curve. In other words, by shifting the input frequency response with the error associated with each frequency, it is possible to remove the warping error introduced by the numerical method.

As it is necessary for the developed algorithm to know a priori the time-step to be used in the EMT simulation, an adjustment menu was placed at the graphical user interface of the program, shown in Figure 4.25.

It should be noted that the model synthesized using the pre-warping adjustment must always be discretized with the same time-step used in the adjustment. If the model is intended to be simulated with varying time-steps and new model generation is not expected, it is recommended not to use the adjustment, making sure that the time-step is small enough so that the model's response is accurate in the frequency range of interest. The error associated with each frequency, depending on the time-step, can be known by means of the equation 4.6.

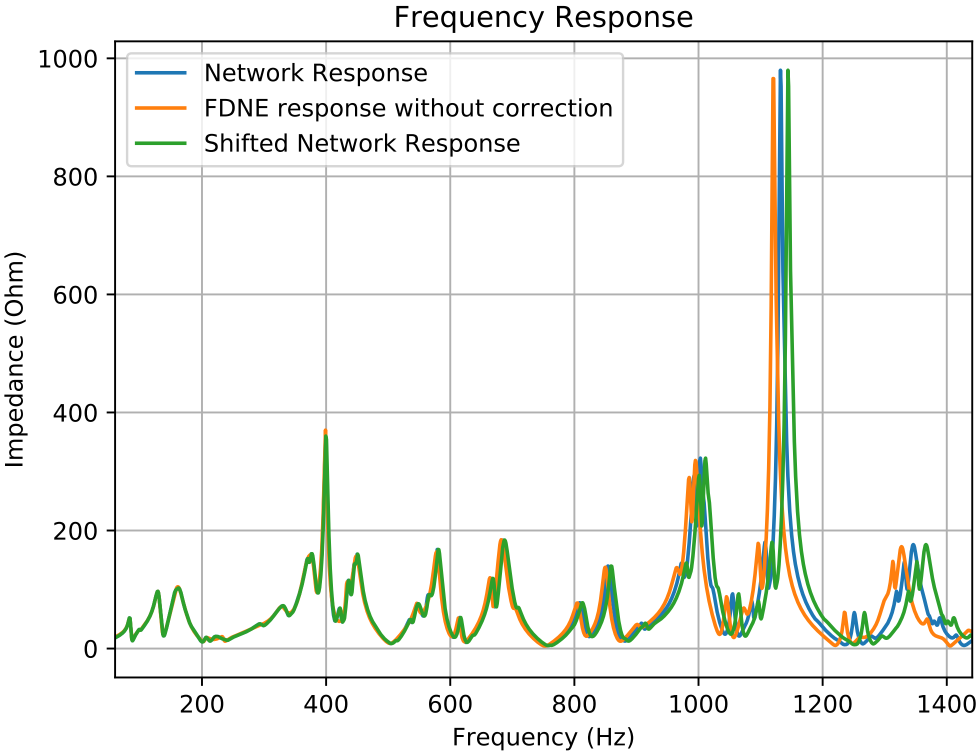
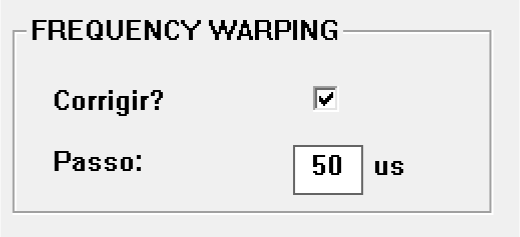


Figure 4.24: Superposition of the original network response, the original FDNE output and the shifted input frequency response.



FREQUENCY WARPING

Corrigir? ☑

Passo: 50 us

Figure 4.25: Frequency warping correction menu.

Chapter 5

FPGA Design Architecture of the FDNE Model

In this chapter the fundamental concepts about FPGAs are presented, detailing their architecture, usage and programming methods. Afterward, the hardware architecture of the developed FDNE will be presented, highlighting the FPGA-RTDS interface and the results obtained.

5.1 FPGA Basics

At the beginning of its introduction in the 1980s, field-programmable gate arrays were initially conceived as a prototyping tool for debugging digital designs of electronic boards before high-scale production. At that time, they offered limited speed and logic density. Further advances in architecture and process have resulted in major improvements in performance, logic density, cost and usability. As the clock speed growth of processors decreased during the last decades, the parallel approach became an ubiquitous solution, what increased the competitiveness of FPGAs in relation to CPUs for many high-performance applications. Today, there is widespread usage of FPGA technology for diverse applications such as: Aerospace, Defense, Image and Audio Processing, Data Centers, Scientific Instruments, Distributed Monetary Systems, etc.

The basic building block of the FPGA is the Look-Up Table (LUT) logic function generator, whose number of inputs ranges from 3 to 8 for different vendors' families. For example, the Xilinx 7 series[75], whose Virtex7 family is used in this work, has 6-input look-up tables. They are located inside structures called Configurable Logic Blocks (CLBs), which together with Digital Signal Processing (DSP) blocks and Block RAMs (BRAMs) comprises the main FPGA resources, whose utilization and performance levels are used to estimate the quality of algorithm implementation. The VC7VX485T hardware resources are shown in Table 5.1.

Table 5.1: VC7VX485T Hardware Resources.

Slices	DSP Slices	BRAM 36 Kb
75900	2800	1030

5.1.1 FPGA Design Tools and Work Flow

FPGAs are usually programmed by any of the following three methods: Hardware Description Languages (HDL), high-level graphical interfaces and the High-Level Synthesis (HLS) approach, based on the C language.

The most used HDLs are Verilog and VHDL, which guarantee high performance and precision when properly designed, but demand a long development time. Graphical interfaces such as the LabVIEW FPGA from National Instruments enable the user to program FPGAs without the knowledge of HDLs. This approach is frequently used for quickly validating control and algorithm designs, but it is not applied for intensive applications, as it does not allow for much design flexibility and performance.

The HLS method translates a C function into an encapsulated HDL core, which is called an Intellectual Property (IP) Core. In this approach, the hardware performance constraints, such as latency, initiation interval and area usage, are specified through code directives. This enables a great reduction in development time while achieving good performance. It represents a trade-off between the performance of the pure HDL code and the development time of the graphical interface method. However, the HLS method requires HDL knowledge, as the IP COREs have to be instantiated into the main HDL design. The HLS method is provided by Xilinx through the Vivado High Level Synthesis tool. This is the method chosen for this work.

Except for the constraint directives, interface ports and variable storage, the FDNE IP CORE was programmed in HLS with almost the same C code used in the RTDS and OPAL-RT environments, with some minor modifications to embed the code inside a C function, as expected by the HLS tool.

5.2 Architecture of the Developed System

5.2.1 FPGA Evaluation Board

Figure 5.1 shows the VC707 Xilinx FPGA board, which hosts a Virtex7 XC7VX485T chip. This platform provides a wide range of components and connectivity options of which only the Small Form-factor Pluggable (SFP) interface is used in this work.

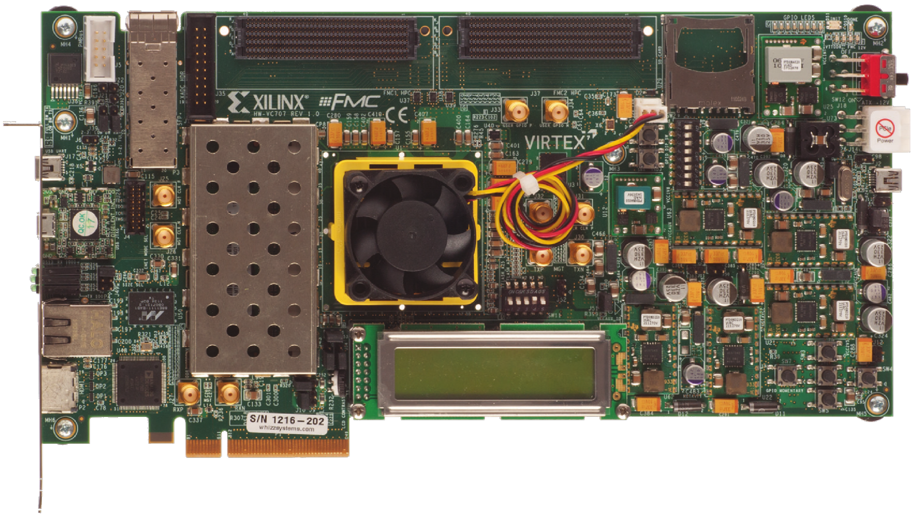


Figure 5.1: Xilinx VC707 board.

5.2.2 SFP Fiber Module

The SFP module is an industry-standard for network interface modules. It is a popular format jointly developed and supported by many network component vendors. Its form factor and electrical interface are specified by a multi-source agreement. SFP modules are commonly used in modern routers, interface cards and switches. The interface between processing and extension RTDS cards is done via SFP modules, using a link speed of 2.5Gbit s^{-1}, via Lucent Connector standard fiber connections.

5.2.3 RTDS-FPGA Interface

The RTDS-FPGA interface is performed through an IP CORE available in the RSCAD tool, called `RTDS_InterfaceModule`, composed of an encrypted VHDL code that can be instantiated as a black box inside the main VHDL design programmed in Vivado. On the EMT simulation side, in the RSCAD, a component called GTFPGA must be instantiated within the simulated circuit. The number and type of input and output variables must be properly passed to the component. The available types are 32-bit floating point and 32-bit unsigned integer. The IP CORE is encrypted so that the proprietary Gigabit Transceiver (GT) protocol can be preserved. This ensures that only the VC707 board (for which the encrypted IP CORE was compiled) can be interfaced into the RTDS. The inputs and outputs of the `RTDS_InterfaceModule` are shown in Figure 5.2.

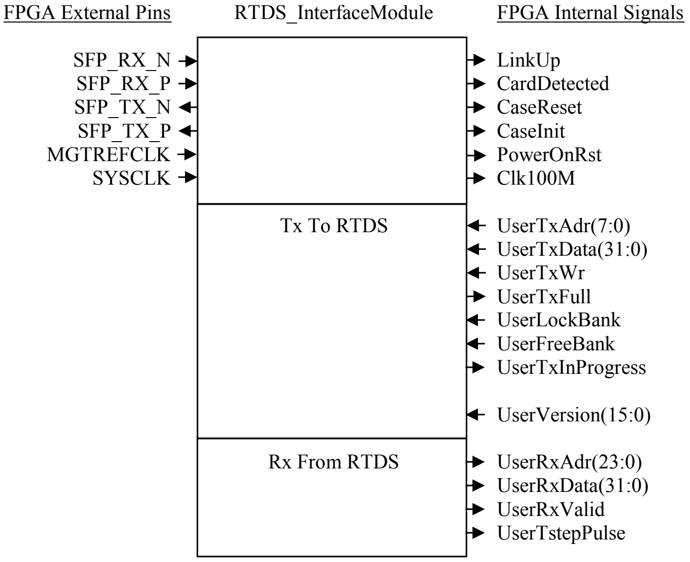


Figure 5.2: Interface IO.

The module is sourced by a 200 MHz system clock (which is the main source clock of the board) and a specific low-jitter 125MHz clock for the Gigabit Transceiver structure. The other external ports are the send and receive pairs of the SFP module. A 100 MHz clock is created inside the IP, which is used to source the user logic external to the IP.

All hardware structures related to the GT operation are inside the IP. The link operates at 2.5Gbit s^{-1}, which a 8b/10b encoding for error correction, resulting in a bandwidth of 250MB s^{-1}.

The remaining IP signals will be explained in the subsection 5.2.5.

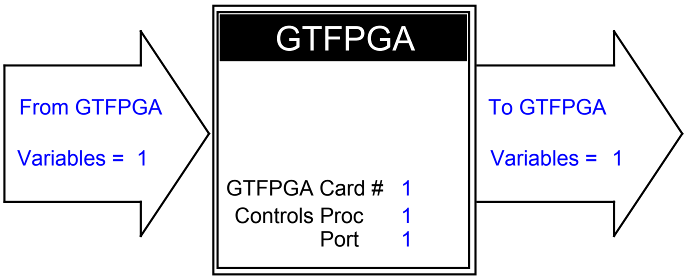
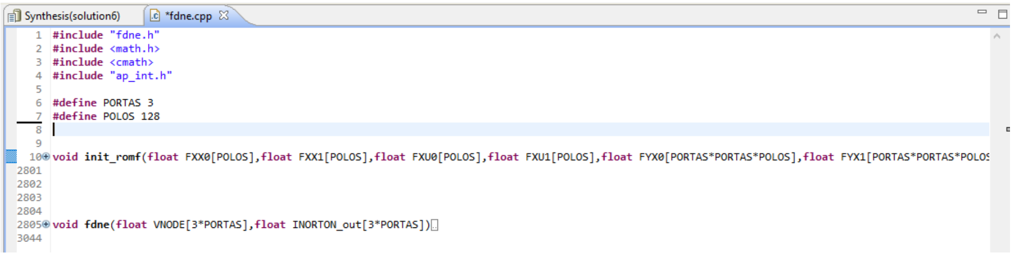


Figure 5.3: RSCAD component for FPGA interface.

5.2.4 HLS-based FDNE component

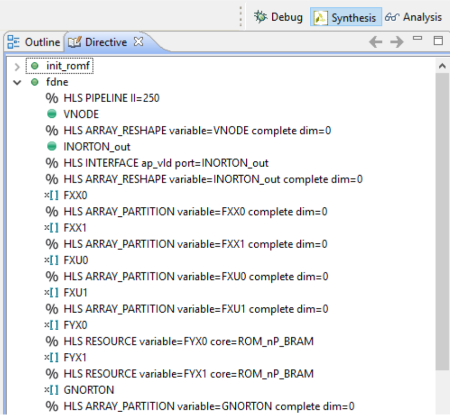
As exposed in Section 5.1.1, the Vivado High Level Synthesis tool maps a C function into an HDL IP CORE, using user-defined directives to specify the post-translation hardware performance. The main HLS FDNE code is composed of two functions. The first one is used to store and initialize all the necessary parameters in ROM memories. The second one has the FDNE algorithm itself, with voltage inputs and current outputs, as illustrated in Figure 5.4, where their instantiations are shown.



```
#include "fdne.h"
#include <math.h>
#include <cmath>
#include "ap_int.h"

#define PORTAS 3
#define POLOS 128



void init_romf(float FXX0[POLOS],float FXX1[POLOS],float FXU0[POLOS],float FXU1[POLOS],float FYX0[PORTAS*PORTAS*POLOS],float FYX1[PORTAS*PORTAS*POLOS




void fdne(float VNODE[3*PORTAS],float INORTON_out[3*PORTAS])
```

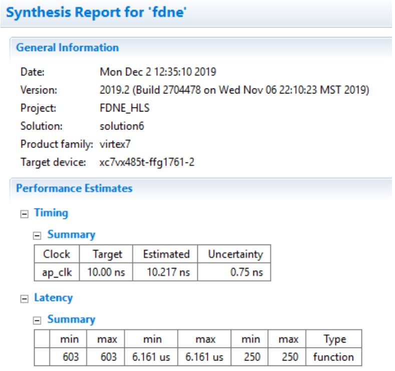
Figure 5.4: HLS main code for FDNE component (change to 1 port code).

Figure 5.5 shows some directives used for the IP synthesis. After synthesizing the design, the HLS tool presents a report for the estimated clock performance and area usage, which is shown if Figure 5.6. The directives associated with a performance report can be saved in structures called solutions, which are independent of the C code. It is a common practice to vary the directives in a heuristic approach, monitoring the reports and saving the results into solutions to be compared.



```
init_romf
fdne
    % HLS PIPELINE II=250
    VNODE
    % HLS ARRAY_RESHAPE variable=VNODE complete dim=0
    INORTON_out
    % HLS INTERFACE ap_vld port=INORTON_out
    % HLS ARRAY_RESHAPE variable=INORTON_out complete dim=0
    [] FXX0
    % HLS ARRAY_PARTITION variable=FXX0 complete dim=0
    [] FXX1
    % HLS ARRAY_PARTITION variable=FXX1 complete dim=0
    [] FXU0
    % HLS ARRAY_PARTITION variable=FXU0 complete dim=0
    [] FXU1
    % HLS ARRAY_PARTITION variable=FXU1 complete dim=0
    [] FYX0
    % HLS RESOURCE variable=FYX0 core=ROM_nP_BRAM
    [] FYX1
    % HLS RESOURCE variable=FYX1 core=ROM_nP_BRAM
    [] GNORTON
    % HLS ARRAY_PARTITION variable=GNORTON complete dim=0
```

Figure 5.5: HLS directives.



Synthesis Report for 'fdne'

General Information

Date:	Mon Dec 2 12:35:10 2019
Version:	2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project:	FDNE_HLS
Solution:	solution6
Product family:	virtex7
Target device:	xc7vx485t-ffg1761-2

Performance Estimates

⊟ Timing

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	10.217 ns	0.75 ns

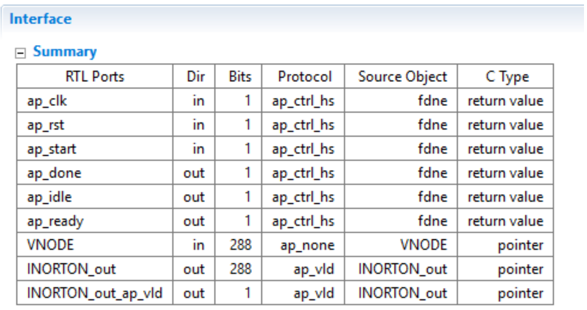
⊟ Latency

⊟ Summary

	min	max	min	max	min	max	Type
	603	603	6.161 us	6.161 us	250	250	function

Figure 5.6: HLS report.

The IP CORE input/output interface port map is also generated by the report, shown in Figure 5.7 for a 3-port FDNE. The tool automatically adds to the interface a clock and reset inputs and a control protocol composed of the following ports: ap_start, ap_done, ap_idle and ap_ready. A complete description of the control protocol can be found in the UG871 HLS Tutorial [76]. In short, the ap_start signal triggers the block operation, the ap_ready signal indicates when the design is ready for new inputs, the ap_done indicates when the design has completed all operations in the current transaction and the ap_idle indicates if the design is operating or idle. The ap_vld attached to the INORTON output vector indicates a valid output.



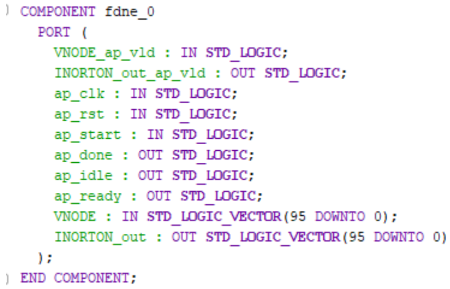
Interface

⊟ Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	fdne	return value
ap_rst	in	1	ap_ctrl_hs	fdne	return value
ap_start	in	1	ap_ctrl_hs	fdne	return value
ap_done	out	1	ap_ctrl_hs	fdne	return value
ap_idle	out	1	ap_ctrl_hs	fdne	return value
ap_ready	out	1	ap_ctrl_hs	fdne	return value
VNODE	in	288	ap_none	VNODE	pointer
INORTON_out	out	288	ap_vld	INORTON_out	pointer
INORTON_out_ap_vld	out	1	ap_vld	INORTON_out	pointer

Figure 5.7: HLS interface for a three-port FDNE.

After a trial and error approach changing directives so that the designed IP met the expected performance and area usage, it can be encapsulated into an IP CORE to be instantiated in the Vivado VHDL main design. Figures 5.8 and 5.9 show the template instantiation code and the module graphical representation in register transfer level schematic, respectively. This case represents a single three-phase port equivalent, as both the nodal voltage vector (VNODE) and the current injections vector (INORTON_out) have 96 bits (three 32-bit variables).



```
COMPONENT fdne_0
  PORT (
    VNODE_ap_vld : IN STD_LOGIC;
    INORTON_out_ap_vld : OUT STD_LOGIC;
    ap_clk : IN STD_LOGIC;
    ap_rst : IN STD_LOGIC;
    ap_start : IN STD_LOGIC;
    ap_done : OUT STD_LOGIC;
    ap_idle : OUT STD_LOGIC;
    ap_ready : OUT STD_LOGIC;
    VNODE : IN STD_LOGIC_VECTOR(95 DOWNTO 0);
    INORTON_out : OUT STD_LOGIC_VECTOR(95 DOWNTO 0)
  );
END COMPONENT;
```

Figure 5.8: The FDNE IP CORE VHDL instantiation template.

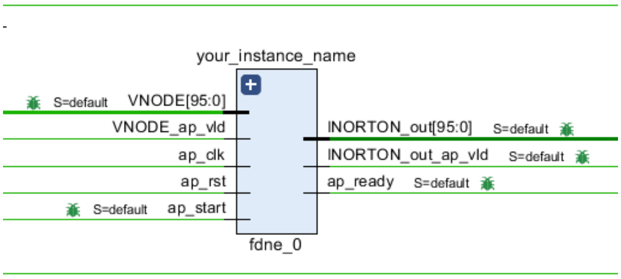


Figure 5.9: FDNE module in RTL schematic.

In Figure 5.9, some signals have a green bug attached to them. This indicates that it is routed to a debug core that enables the signal to be monitored in real-time. This is done by the Integrated Logic Analyzer Core (ILA CORE), whose captured waveforms will be shown in the next subsection. This IP CORE is used only in the development stage, as it can increase area usage and can decrease performance significantly.

Table 5.2 shows estimations of the better-achieved area usage and clock performance for a 1-port 64 poles and a 3-port 128 poles FDNEs. These values are estimations and must be confirmed after the implementation step.

Table 5.2: Estimation of area usage and clock performance (100 MHz) for two different FDNEs.

	1 port 64 poles	3 port 128 poles
Δt (μs)	2.85	6.1
BRAM (%)	0	1
DSP (%)	1	4
FF (%)	1	32
LUT (%)	3	38

The area usage of the 1 port 64 poles FDNE is very small, so it would be possible to increase its timing performance, using more logic blocks, if needed. But as the 2.85 μs time-step obtained is sufficiently small for most EMT applications, the timing performance was not stressed. The 3 port 128 poles FDNE required significantly more logic blocks to be implemented, due to the increased size of the state matrices. Its usage of FF is high due to the "Array Partition = complete" directive, which imposes the arrays to be stored in registers, instead of BRAM. The advantage of using registers is that the complete array is read with a single clock pulse, unlike the BRAMs, which deliver only one or two variables at a time, depending on the type of memory. Thus, it is a timing performance versus area usage compromise.

5.2.5 Main VHDL Design in Vivado

The overall system design is composed of the `RTDS_InterfaceModule` and the `FDNE_Module` IP COREs interconnected by two state machines implemented in VHDL for receiving and sending data to the RTDS, which are depicted in Figures 5.10 and 5.11, respectively.

The `UsertTstepPulse` signal, which is an output of `RTDS_InterfaceModule`, indicates the beginning of a time-step in the EMT simulation at the RTDS side. A delay of around 400 ns due to the transceiver's operation and the optical link is estimated in the user manual. After about 100 clock cycles the `UserRxValid` signal indicates that a word is available to be read at the `UserRxAdr` address, represented by the index k in the state machine. After receiving the expected number of words, the signal `ap_start` is asserted, triggering and starting the FDNE IP CORE.

The signal `INORTON_out_ap_vld` is asserted when the FDNE IP CORE outputs are available and ready to be read. This signal triggers the "sending to RTDS" process, changing from the state-machine from the IDLE state to the LOCK state, where the signals `UserLockBank` which locks the memory bank and `UserTxWr`, the transmission write enable, are asserted. In the next state SEND the address `UserTxAdr` is provided and the data to be transmitted is assigned to the `UserTxData` signal. After all expected words are transmitted, the `UserFreeBank` is asserted in the FREE state, after which the state backs to IDLE. This ensures that all transmitted values will be updated in the same time step at the RTDS side. The de-assertion of the `UserTxInProgress` indicates that the transmission is over and the RTDS received all expected words.

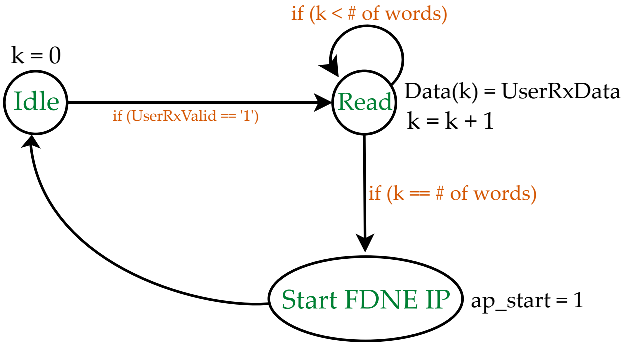


Figure 5.10: Reading data state machine.

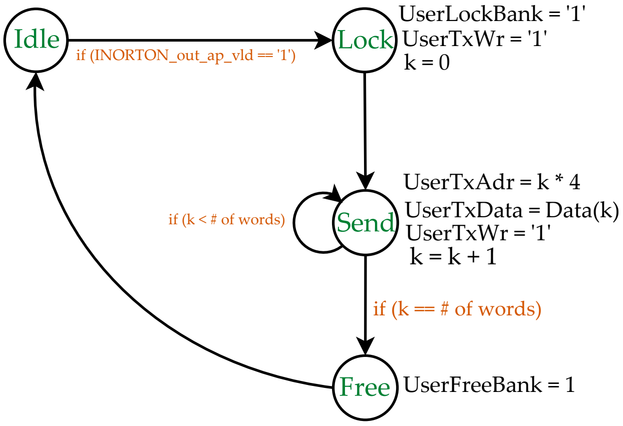


Figure 5.11: Sending data state machine.

The ILA captured waveforms are shown in the Figures 5.12 and 5.13.

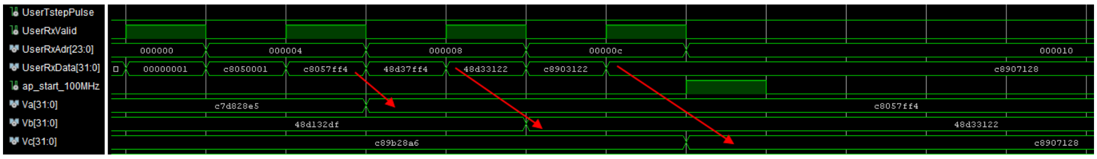


Figure 5.12: Receiving data from RTDS waveforms.

Referring to the Figure 5.12, the assertion of the signal `UserRxValid` indicates that a valid word has been placed at the `UserRxData` signal. The first received word is a 32-bit unsigned variable valued 0x00000001. The following values are the three terminal voltages represented by 32-bit floating point numbers, stored in Va, Vb and Vc signals, respectively. When the last word is received, the signal `ap_start_100MHz` is asserted, triggering and starting the FDNE component, as all expected inputs are readily available.

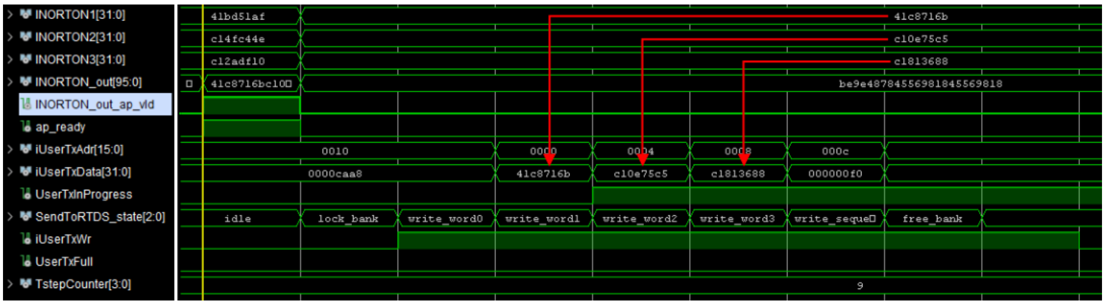


Figure 5.13: Sending data to RTDS waveforms.

Referring to the Figure 5.13, the `INORTON_out_ap_vld` assertion indicates that in the next clock rising, the `INORTON_out` signal has valid output data from the FDNE component and triggers the sending state machine. Then, after the *lock* *bank*, the write enable `iUserRxWr` is asserted and the `INORTON` signals are progressively sent to the `iUserTxData` signal. The `UserTxInProgress` indicates that data is being transmitted to RTDS. After all data is sent, the sending state machine returns to the idle state.

The Figures 5.14, 5.15 and 5.16 presents the Vivado's report of hardware components usage, the timing summary and the area usage after implementation. The hardware utilization refers to the whole design, including the `RTDS_-InterfaceModule` and the `FDNE_Module` IPs and the logic employed to construct the sending and receiving state machines.

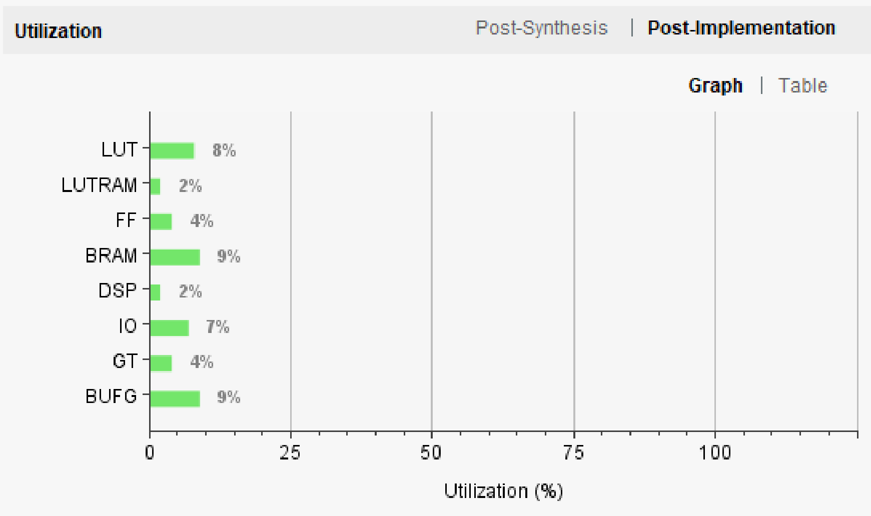
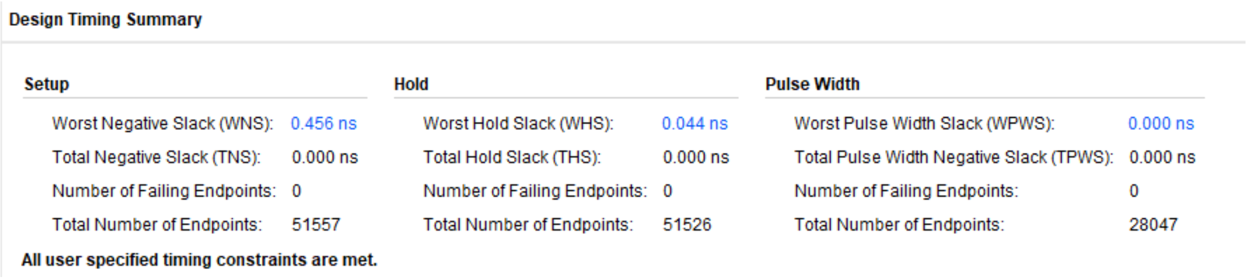


Figure 5.14: Post-implementation hardware utilization.

The timing summary confirms that all expected timing constraints are met.



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.456 ns	Worst Hold Slack (WHS):	0.044 ns	Worst Pulse Width Slack (WPWS):	0.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	51557	Total Number of Endpoints:	51526	Total Number of Endpoints:	28047

All user specified timing constraints are met.

Figure 5.15: Post-implementation timing summary.

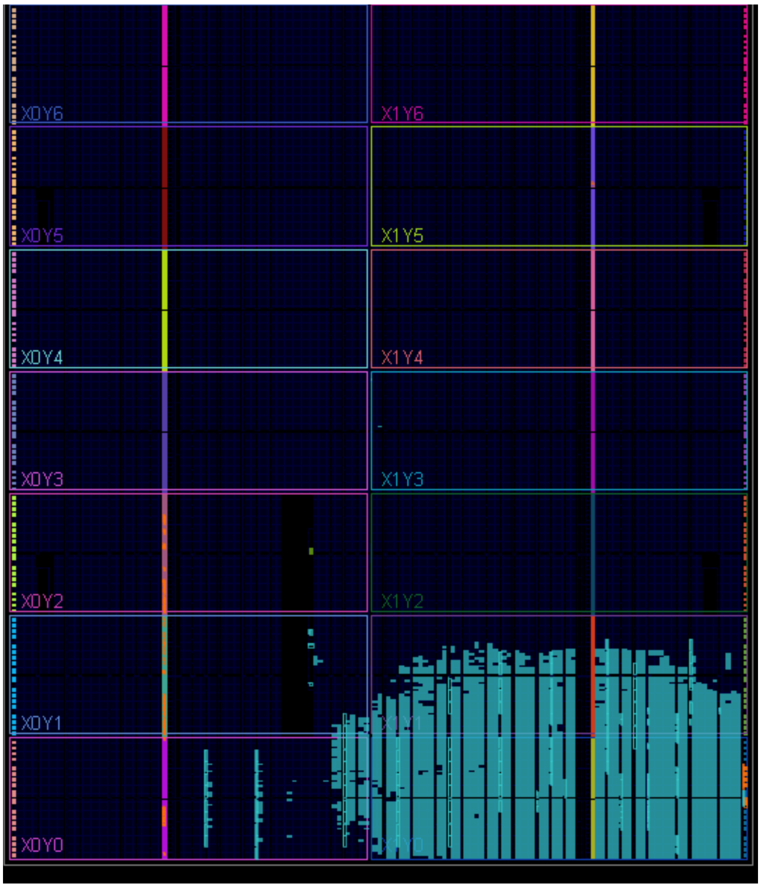


Figure 5.16: Post-implementation area usage.

5.2.6 RSCAD Design

The circuit simulated in the RSCAD tool is illustrated in Figure 5.17. The GTFPGA component, as described in subsection 5.17, defines the number and type (32 bit floating point or integer) of the words to be transmitted and received from the FPGA board. The current injection component is the same used in the validation examples from the previous chapter. In addition to the GTFPGA, another component, name FDNE-FPGA, is necessary to instantiate the nodal terminals and to stamp the FDNE conductance into the global conductance matrix of the simulated network. This component also converts the variables received from the FPGA component into nodal injections to populate the current vector in the nodal analysis global solution of the network. The GTFPGA component cannot send its variables to the current vector itself, as it is understood by the solver as a control component, and as such, its values are control variables.

Unfortunately, the process of converting a control variable into a power variable (i.e., sending the variable from the GTFPGA to the FDNE-FPGA component) introduces a delay of one time-step. This delay can be handled with the employment of a stub-line (a transmission line whose propagation delay matches or surpasses the integration time-step) that decouples the system but modifies the frequency response. RTDS has recently presented the Aurora Link component, which performs the Aurora 8b10b protocol developed by Xilinx for high-speed communication. According to the user guide, this component could stall the network solution until the variables come back from the FPGA. By doing so, the currents vector could theoretically be injected in the same time-step corresponding to the voltage readings. Unfortunately, at the time of this work, the Aurora Link component was not readily available.

To circumvent this limitation, a scheme using the RTDS *Substep* feature has been adopted. This tool is only available for the Novacor hardware. The *Substep* environment implements a multi-rate simulation, allowing the user to model part of the network with a smaller time-step than the mainstep. *Substep* simulation executes the contents inside of a hierarchy box multiple times in every mainstep, being interfaced to the main time-step with a transmission line. Its results are transferred to the mainstep network once every mainstep.

The substep environment is employed specifically for the exchanging of variables between the GTFPGA and the FDNE-FPGA components. This is done by placing both components inside the substep box, but keeping the FDNE discretization for the main time-step. Besides that, the FDNE must be enforced to be executed only in the first substep. For example, with a main time-step is 50μ s and a substep multiplier of 5, a substep of 10μ s will be inferred. Then, in the first substep, the FDNE component will calculate current injections for a time-step of 50μ s and will be in the idle state for the next four substeps. At the RTDS side, this nodal contribution will populate the current vector at the end of the second substep. In this way, the FDNE is solved in the first substep and the conversion is done in the second, reducing the delay to one substep, instead of one main step.

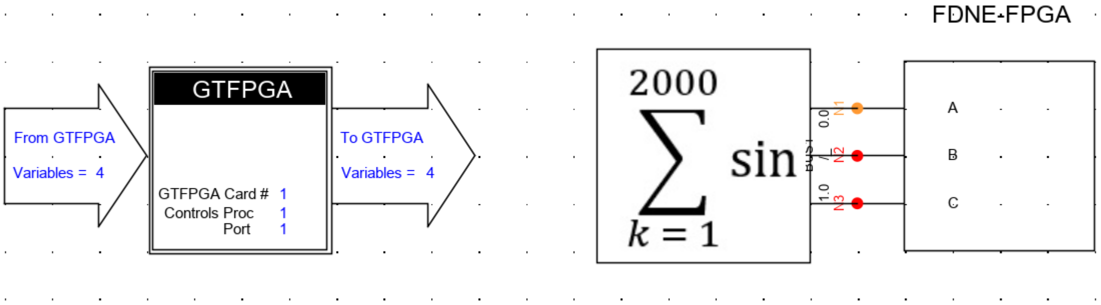


Figure 5.17: Simulation with substep component.

Due to the used *Substep* scheme, the state-machine of Figure 5.10 was modified to only trigger the FDNE component at the first of the five substeps. This was achieved by sending to the FPGA a substep identifier ranging from 1 to 5, named as `flag` signal in the state-machine of Figure 5.18.

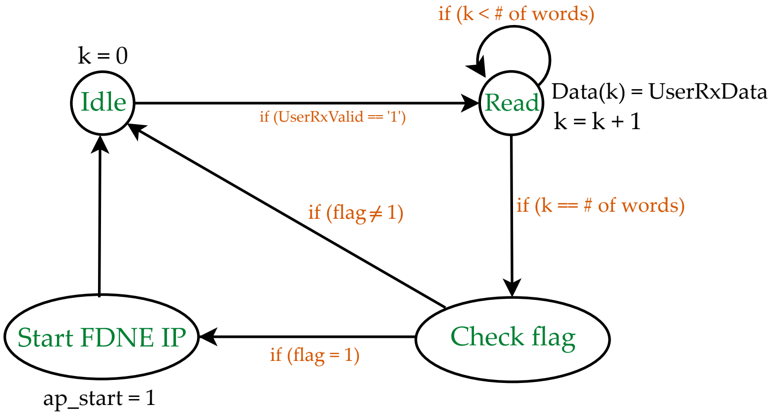


Figure 5.18: Reading data state machine with flag signal, in order to only trigger FDNE at the first substep.

In the waveform of Figure 5.19, the `UserTstepPulse` signal is asserted at the beginning of a time-step. The `UserRxValid` signal is asserted at every substep, as the GTFPGA component is inside the *Substep* box and receives new voltage values every substep. As expected, the `ap_start` signal that triggers the FDNE IP CORE is only asserted at the first substep.

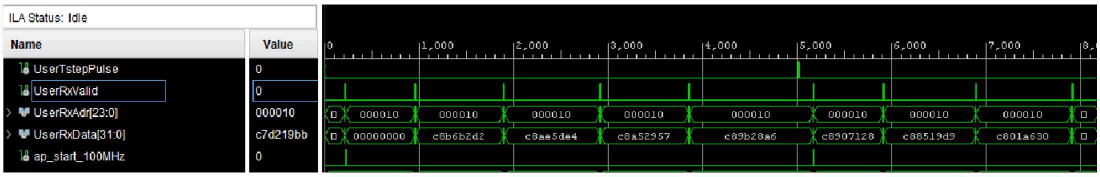


Figure 5.19: Waveform showing the *Substep* scheme.

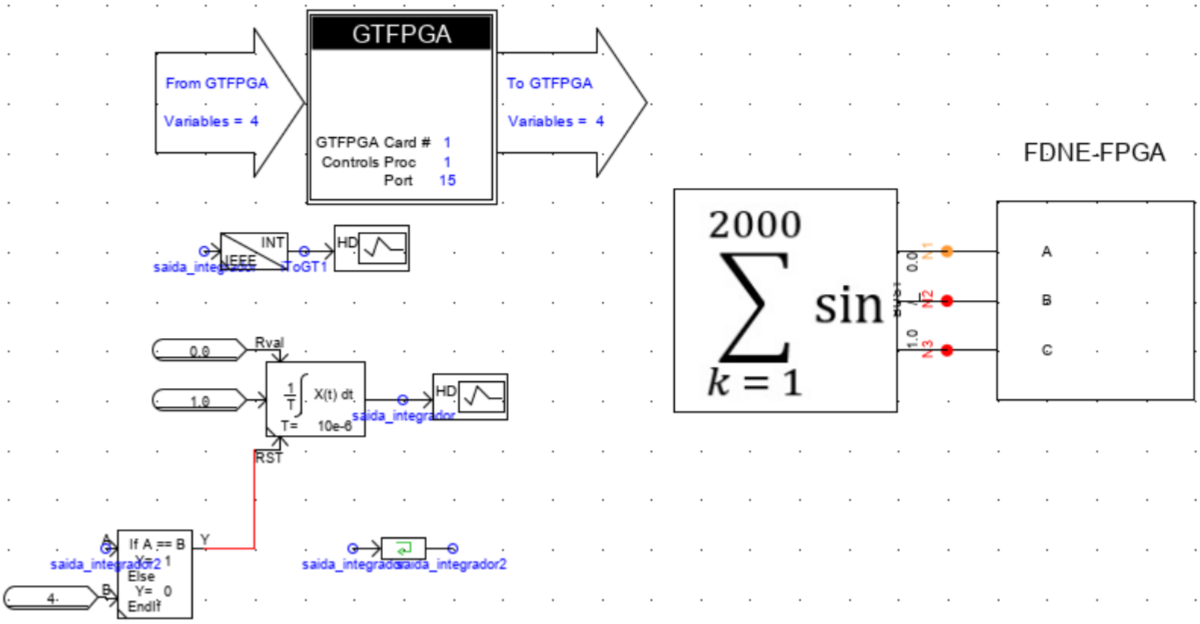


Figure 5.20: Simulated circuit for FPGA-based FDNE validation.

5.3 Validation

5.3.1 Harmonic Injection

The same external network and validation methodology of the previous chapter was applied in the FPGA. Figure 5.21 shows the harmonic voltage measured at the Miranda 500 kV FDNE terminal, for the three-port case adjusted with 128 poles. After applying the FFT to it, the impedance frequency response of Figures 5.22 and 5.23 was obtained, validating the FDNE component.

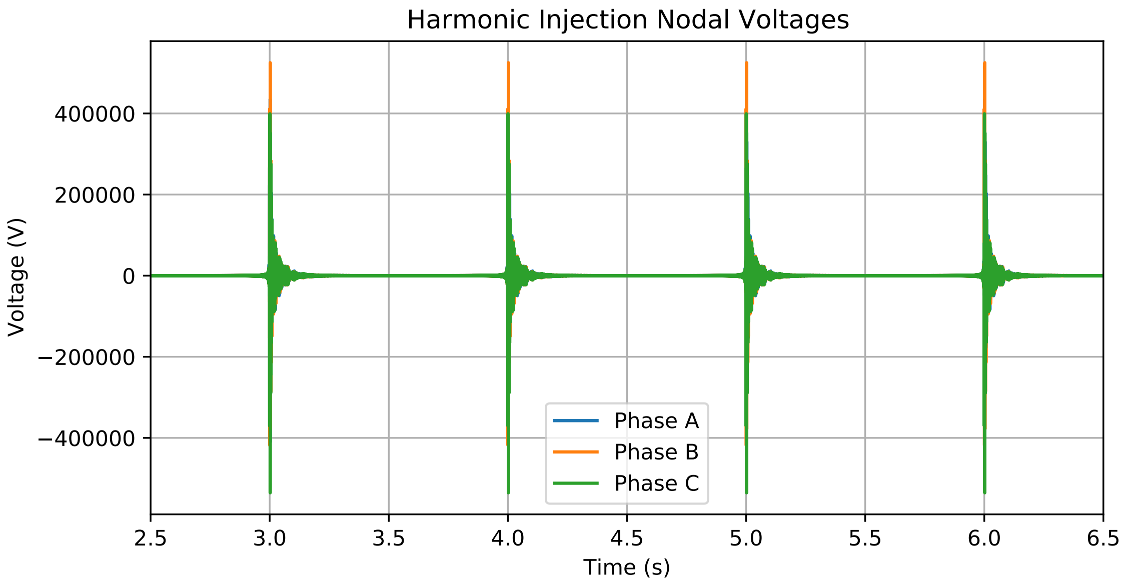


Figure 5.21: Voltage waveform measured at the FDNE bus subjected to harmonic injection.

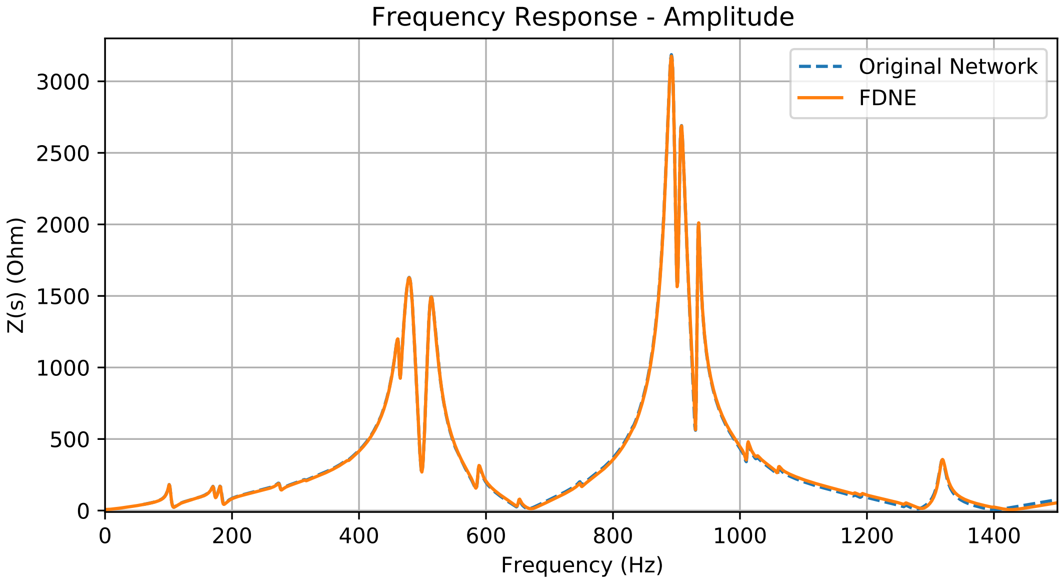


Figure 5.22: Impedance Module frequency response.

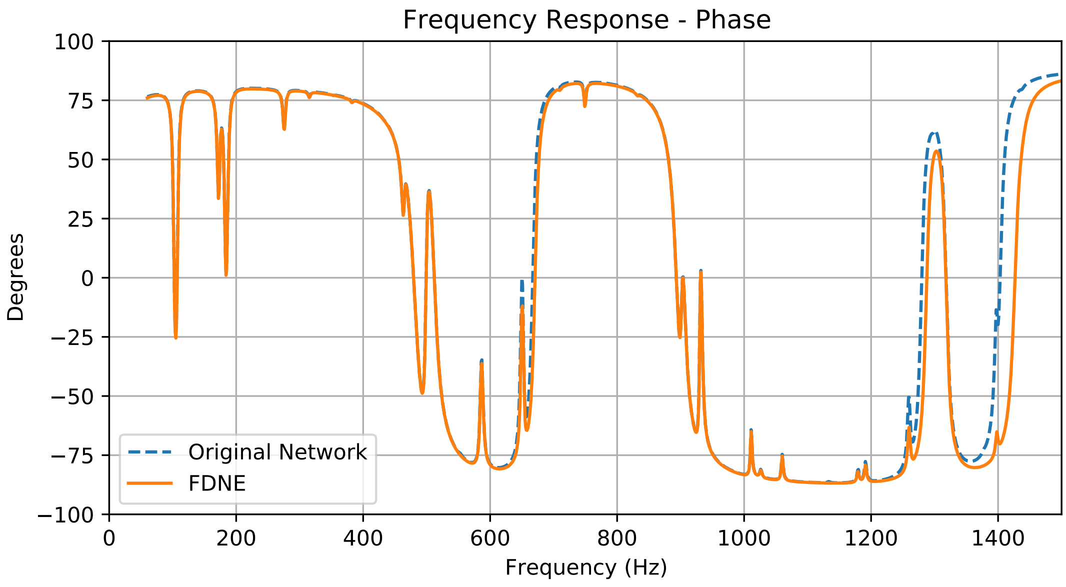


Figure 5.23: Impedance Phase frequency response.

5.3.2 Transformer Energization

A challenging case was performed exploring the co-simulation setup with the RTDS and OPAL-RT platforms that was developed by ONS in 2018 and since then has been employed to better use the resources of both simulators. The co-simulation is carried out by splitting the network with transmission lines whose propagation delay is equal to or greater than the time-step, what makes the conductance matrix block diagonal, allowing each block to be solved by a different core or platform. The simulators are interfaced through the VC707 FPGA board located in the OP-5700 (OPAL-RT) hardware, which communicates with the RTDS through the same IP CORE used in the FDNE interface, shown in Figure 5.2 in the previous subsection. The bitstream programmed in the FPGA board for the co-simulation was the same used on a previous work, developed in Aachen University, to connect the same two simulators [34]. More details about the co-simulation implementation can be found in the reference [77].

For this case, the previous network was reduced to a single port, 64 poles equivalent for assessing the energization of the 450 MVA 500 - 230 kV three winding transformer located at the Miranda II substation. In order to allow the co-simulation, a stub line was placed between the transformer and the FDNE. The simulated circuit is shown in Figure 5.24. The present simulation was presented in the RT20 conference from OPAL-RT, which can be accessed in [78].

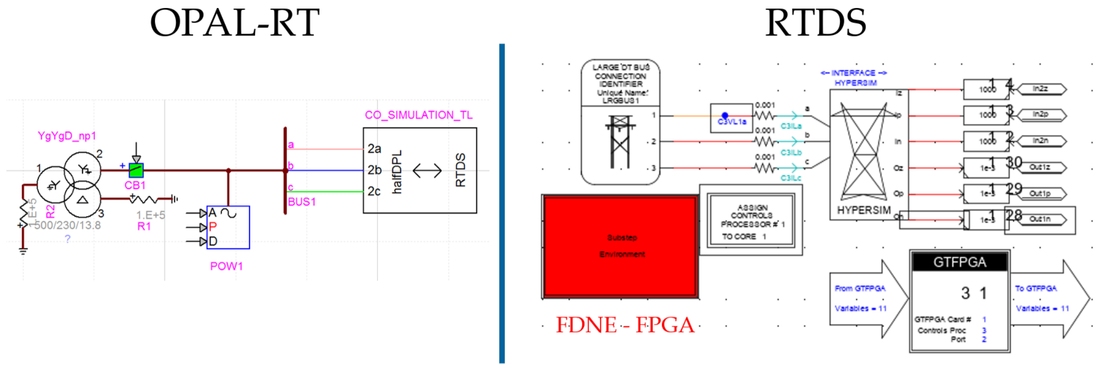


Figure 5.24: Co-simulation infrastructure.

On the left side of the figure is the circuit simulated in Hypersim, composed of the three-winding transformer and half side of the transmission line employed in the co-simulation, which is the interface to the RTDS simulator. On the right side is the simulated circuit in RTDS, composed of the transmission line terminal interfaced with the Hypersim, the GTFPGA component that implements this interface and the substep component, inside which the FDNE and its respective GTFPGA component are located.

The full network was simulated in the Hypersim. Figures 5.25 and 5.26 show the comparison of the full and the reduced networks for the voltage and inrush currents waveforms, respectively. These results presented similar accuracy as the offline model in EMTP, validating the FPGA-based FDNE model and the substep scheme for reducing the time-step delay.

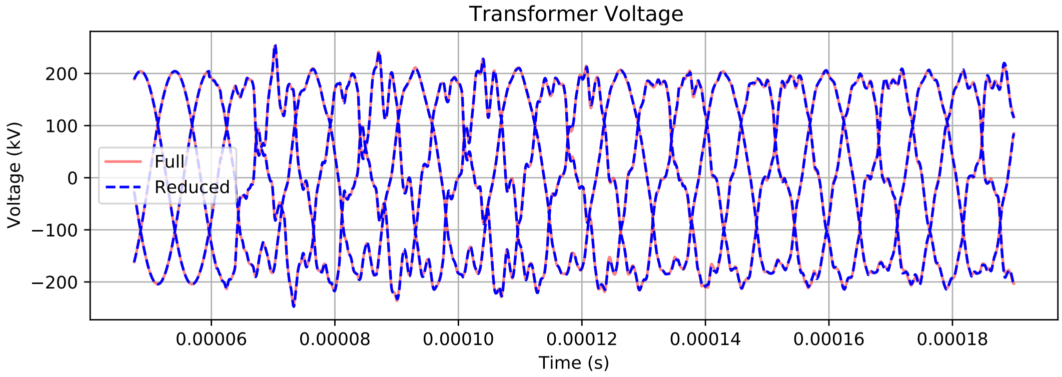


Figure 5.25: Voltage at the transformer bus during energization.

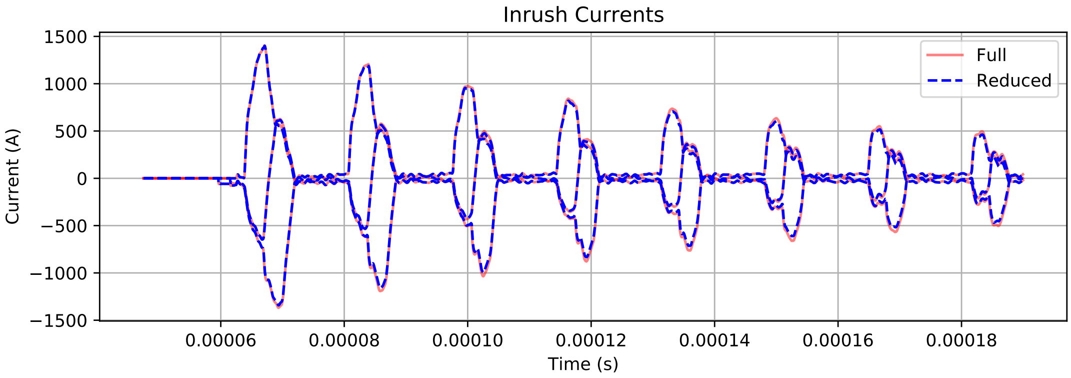


Figure 5.26: Transformer inrush current during energization.

5.3.3 Performance

The model worked as intended and allowed for a representation of FDNEs with a high number of poles and ports. Compiling HLS models and generating bitstreams is a very time-consuming task, what limited the number of different FDNEs modeled in the FPGA during the time of this work. Table 5.3 presents a comparison of the minimum time-step obtained with the three FDNE implementations for real-time, namely, the RTDS, the OPAL-RT and the FPGA integrated with the RTDS. One should note that these results do not refer to the execution time of the FDNE component, instead, these are the minimum time-step of the simulation composed only by the FDNE connected to a current source.

For the smallest cases the minimum time-steps are similar, probably because the FDNE execution is so fast that the network solving time takes precedence. Besides that, there is not much difference in the FPGA cases, as the communication overhead spent the most of the time. This difference becomes observable for the bigger cases, where the FDNE execution times overwhelm the network solver and the communication overhead. For all cases, the FDNE embedded in FPGA has overcome the IBM POWER8 and Intel Xeon implementations.

As done for the other real-time implementations, the time-step was increased to 50 μs in order to assess the maximum number of FDNE components in a typical EMT study. This comparison was made only for the three-port, 128 poles FDNE. A directive was placed in the HLS code, allowing the FDNE component to spend 4000 cycles to be solved, what gives 40 μs with the employed 100MHz clock. This number was chosen to allow 10 μs to be spent on communication overhead and on the network solver. A maximum of 9 components could be allocated in the FPGA, against 5 and 2 for one core of the RTDS and OPAL-RT, respectively. This was not exhaustively optimized, therefore further enhancements are still possible.

Table 5.3: Execution time and Minimum time-steps for FDNEs with different number of poles and ports.

Ports	Poles	RTDS Δt (μs)	Custom Component Δt (μs)	FPGA Δt (μs)
1	32	4	3	3
	64	5	4	4
	128	5	6	4
2	32	5	5	4
	64	6	8	4
	128	8	14	4
3	32	7	8	4
	64	9	13	6
	128	13	25	7

5.4 Chapter Outline

This chapter presented a description of the FDNE implementation in the FPGA platform, illustrating the basic concepts of this type of hardware, its programming and development methods. Several details about the implementation in HLS were illustrated. The interface with the RTDS platform was presented, showing the data communication waveforms.

Two EMT results were presented, the first illustrating the frequency response of the FDNE in the frequency range of the fitting, and the second showing a three-winding transformer energization, in a co-simulation scheme between the OPAL-RT and RTDS platforms, where the FDNE was implemented in the FPGA interfaced with the RTDS side.

The performance results showed that the FPGA implementation presented the smallest time-steps and the largest amount of FDNEs per core, compared to the RTDS and OPAL-RT implementations. The only disadvantage observed was related to the delay of one time-step between receiving the FPGA data and its respective insertion in the network solution, which brought the need for a scheme using the substep environment, which requires the use of a transmission line between the FDNE and the network. Further investigations are needed to assess whether the same delay will be observed when interfacing an FPGA component to the OPAL-RT simulator.

The use of customized components in FPGA allows better employment of real-time simulation resources since they offer high performance at a much lower cost than real-time simulation hardware. This practice is essential for large-scale simulation of power systems, allowing large networks to be represented for different types of studies, without the need to expand the simulator to levels that would be prohibitive from a financial point of view.

The overall workflow for executing a simulation with the FPGA board integrated into the RTDS as a custom model is:

1. Generate the HLS code in the SINTEQV tool.
2. Compile the FDNE IP CORE in Vivado HLS.
3. Instantiate the IP at the main Vivado design.
4. Generate the bitstream and program it in the board.
5. Configure the GTFPGA component with the proper number of receiving and transmitting words.
6. Run RSCAD simulation.

Chapter 6

Conclusions and Future Work

6.1 Summary

The main objective of this work was to develop a customized State-Space solver to increase the time-domain performance of the Vector-Fitting based FDNE. It relies on the representation of the state matrices as vectors, and comprises only simple arithmetic operations at the discretization and time-domain solution of the model, without resorting to the use of any linear algebra library. The developed solver can perform exactly the same mathematical operations as a default state-space solver, with an increase in performance that can reach several orders of magnitude, depending on the number of terminals and poles of the equivalent.

The main fundamentals regarding the FDNE interfacing to traditional EMT solvers were carefully explained, showing the implementation differences that exist when employing impedance or admittance fittings, as well as phase or modal coordinates. An initialization scheme for simulators that runs steady-state solution was presented and implemented in the EMTP software.

Statistical study-cases of transformer energization were shown to fully present the advantages of the FDNE, as well as the limitations in the use of traditional short circuit equivalents when the frequency response of the reduced network is not properly examined.

Regarding the real-time implementation, an assessment of feasible FDNE sizes for achieving the mark of 50 μs in the RTDS and OPAL-RT simulators was made. The FPGA implementation was presented in detail, examining the High-Level-Synthesis approach and also the interface to the RTDS simulator. A scheme making use of the *Substep* has been developed in order to circumvent the one time-step delay imposed by the control to power variable conversion, as the GTFPGA component is instantiated as a control component in RSCAD. Besides that, an algorithm has been developed for the automatic synthesis of sectioned FDNEs targeting parallel processing for the RTDS platform. The fundamental strategy of FDNE sectioning was fully described. Also, an algorithm that shifts the frequency axis of the input frequency responses was proposed to correct the warping error introduced by the trapezoidal integration method, allowing for the perfect match between the original network frequency response (the one obtained in the frequency-domain, performing several steady-state solutions, one for each desired frequency) and the response obtained in time-domain, obtained through the injection of a summation of unitary sine waves and applying the FFT to the resulting voltage waveform.

6.2 Conclusions

This dissertation shows that the traditional FDNE based on Vector Fitting is suitable for real-time simulation, either programmed in FPGA or as a user-customized model in commercial simulators. For large systems with a high number of ports and poles, the FPGA may be the only viable option. For smaller systems, the user customized model embedded on the existing simulators avoids the need to interface with external hardware.

For large-scale real-time simulations, the amount of hardware needed to obtain time-steps compatible with the electrical phenomena to be studied (on the order of microseconds) is very high, often being financially prohibitive, given the high cost of this type of hardware. The use of customized models in FPGA is a promising solution to better represent network components with high computational efficiency, at low cost, when compared to real-time simulators.

6.3 Future Work

The following directions are suggested for future developments:

- The improvement of the FPGA-based FDNE model workflow by allowing the FDNE parameters to be changed without the need to regenerate the bitstream. This is accomplished with the *updatemem* Xilinx tool and would allow the user to generate the FPGA model directly from SINTEQV GUI.

- The FPGA interface to the OPAL-RT platform. In this work, only the RTDS interface has been developed.

- The development of an FPGA-based wide-band equivalent for representing also the low-frequency phenomena related to the rotating machine electromechanical interactions with the grid.

- Perform a more detailed study on the accuracy differences regarding the adoption of admittance or impedance quantities for the fitting.

- Employing this solver to another Vector Fitting applications.

- Assess the performance of the developed FDNE model in the new generations of FPGA boards.

- Investigate other possibilities for removing the one time-step delay in the external FPGA models in RTDS.

References

[1] ▲ MAYO, L. A. *Simulation without Replication: How Some Digital Computer Simulations Serve as Scientific Experiments*. Ph.D. dissertation, University of Notre Dame, Notre Dame, Indiana, 2011.

[2] ▲ SANTO, S. E., FRANÇA, V., ALMEIDA, H. "Testing a Protection system using the RTDS Batch Mode Facility", *Proc. of the IPST 2001*, v. 2, pp. 447–452, 2001.

[3] ▲ SANTO, S., FRANCA, V. "FURNAS experience on real-time computer simulations of power systems", pp. 1248–1253, 2004. doi: 10.1109/PSCE.2004.1397642. Disponível em: <http://ieeexplore.ieee.org/document/1397642/>.

[4] ▲ DE BARROS, H. M., DE CASTRO, A., TAKAHATA, A. Y., et al. "O Simulador de Sistemas de Corrente Contínua do ONS e os Desafios para a Interligação do Primeiro Bipolo de Belo Monte ao Sistema Interligado Nacional", *XXIV SNPTEE*.

[5] ▲ ANEEL. "Edital de Leilão N°007/2008. Anexo 6F-CC- Lote LF-CC". 2008.

[6] ▲ ARRILLAGA, J., WATSON, N. "Computer Modelling of Electric Power Systems (Including Facts)". 2001.

[7] ▲ WATSON, N., ARRILLAGA, J. *Power systems electromagnetic transients simulation*, v. 39. IET, 2003.

[8] ▲ KUNDUR, P., BALU, N. J., LAUBY, M. G. *Power system stability and control*, v. 7. McGraw-hill New York, 1994.

[9] ▲ DOMMEL, H. W. "Techniques for analyzing electromagnetic transients", *IEEE Computer Applications in Power*, v. 10, n. 3, pp. 18–21, 1997.

[10] ▲ HO, C.-W., RUEHLI, A., BRENNAN, P. "The modified nodal approach to network analysis", *IEEE Transactions on circuits and systems*, v. 22, n. 6, pp. 504–509, 1975.

[11] ▲ MAHSEREDJIAN, J., ALVARADO, F. “Creating an electromagnetic transients program in MATLAB: MatEMTP”, *IEEE Transactions on Power Delivery*, v. 12, n. 1, pp. 380–388, 1997.

[12] ▲ MARTINEZ-VELASCO, J. A. *Transient analysis of power systems: solution techniques, tools and applications*. John Wiley & Sons, 2014.

[13] ▲ FARUQUE, M. O., STRASSER, T., LAUSS, G., et al. “Real-time simulation technologies for power systems design, testing, and analysis”, *IEEE Power and Energy Technology Systems Journal*, v. 2, n. 2, pp. 63–73, 2015.

[14] ▲ DUNN, W. H., ELDERT, C., LEVONIAN, P. V. “A Digital Computer for Use in an Operational Flight Trainer”, *IRE Transactions on Electronic Computers*, v. EC-4, n. 2, pp. 55–63, jun. 1955. ISSN: 0367-9950. doi: 10.1109/IRETELC.1955.5407892. Disponível em: <http://ieeexplore.ieee.org/document/5407892/>.

[15] ▲ BAUER, W. F. “Aspects of Real-Time Simulation”, *IRE Transactions on Electronic Computers*, v. EC-7, n. 2, pp. 134–136, jun. 1958. ISSN: 0367-9950. doi: 10.1109/TEC.1958.5222524. Disponível em: <http://ieeexplore.ieee.org/document/5222524/>.

[16] ▲ CZERNIEJEWSKI, F. R. “Real Time Digital Solution of Power Reactor Kinetics Using Z-Transforms”, *IEEE Transactions on Nuclear Science*, v. 16, n. 1, pp. 218–221, 1969. ISSN: 0018-9499. doi: 10.1109/TNS.1969.4325110. Disponível em: <http://ieeexplore.ieee.org/document/4325110/>.

[17] ▲ ROY, L., TECH, M., RAO, N. D. “Real-time monitoring of power systems using fast second-order method”, *MA Y*, v. 130, n. 3, pp. 8, 1983.

[18] ▲ ELDER, L., METCALFE, M. “An Efficient Method for Real-Time Simulation of Large Power System Distribances”, *IEEE Transactions on Power Apparatus and Systems*, v. PAS-101, n. 2, pp. 334–339, fev. 1982. ISSN: 0018-9510. doi: 10.1109/TPAS.1982.317111. Disponível em: <http://ieeexplore.ieee.org/document/4111329/>.

[19] ▲ WALKER, L. N., OTT, G. E., DAY, A. L., et al. “Modification and Performance Evaluation of the Faster-Than-Real-Time Hybrid Simulator”, *IEEE Transactions on Power Apparatus and Systems*, v. PAS-97, n. 5, pp. 1795–1804, set. 1978. ISSN: 0018-9510. doi: 10.1109/TPAS.1978.354673. Disponível em: <http://ieeexplore.ieee.org/document/4181621/>.

[20] ▲ WANG, X., MATHUR, R. "Real-time digital simulator of the electromagnetic transients of transmission lines with frequency dependence", *IEEE Transactions on Power Delivery*, v. 4, n. 4, pp. 2249–2255, out. 1989. ISSN: 08858977. doi: 10.1109/61.35654. Disponível em: <http://ieeexplore.ieee.org/document/35654/>.

[21] ▲ MATHUR, R., WANG, X. "Real-time digital simulator of the electromagnetic transients of power transmission lines", *IEEE Transactions on Power Delivery*, v. 4, n. 2, pp. 1275–1280, abr. 1989. ISSN: 08858977. doi: 10.1109/61.25614. Disponível em: <http://ieeexplore.ieee.org/document/25614/>.

[22] ▲ KEZUNOVIC, M., AGANAGIC, M., SKENDZIC, V., et al. "Transients computation for relay testing in real-time", *IEEE Transactions on Power Delivery*, v. 9, n. 3, pp. 1298–1307, 1994.

[23] ▲ MCLAREN, P., KUFFEL, R., WIERCKX, R., et al. "A real time digital simulator for testing relays", *IEEE Transactions on Power Delivery*, v. 7, n. 1, pp. 207–213, 1992.

[24] ▲ FUJIMOTO, Y. "Real-time power system simulator on a PC cluster", *Proc. of the IPST'99*, 1999.

[25] ▲ PAK, L.-F., FARUQUE, M. O., NIE, X., et al. "A versatile cluster-based real-time digital simulator for power engineering research", *IEEE Transactions on Power Systems*, v. 21, n. 2, pp. 455–465, 2006.

[26] ▲ HOLLMAN, J. A., MARTÍ, J. R. "Real time network simulation with PC-cluster", *IEEE transactions on power systems*, v. 18, n. 2, pp. 563–569, 2003.

[27] ▲ DINAVAHI, V. R., IRAVANI, M. R., BONERT, R. "Real-time digital simulation of power electronic apparatus interfaced with digital controllers", *IEEE Transactions on Power Delivery*, v. 16, n. 4, pp. 775–781, 2001.

[28] ▲ PARMA, G. G., DINAVAHI, V. "Real-time digital hardware simulation of power electronics and drives", *IEEE Transactions on Power Delivery*, v. 22, n. 2, pp. 1235–1246, 2007.

[29] ▲ MATAR, M., IRAVANI, R. *An FPGA-based real-time simulator for the analysis of electromagnetic transients in electrical power systems*. Tese de Doutorado, Ph. D. dissertation, University of Toronto, 2009.

[30] ▲ MATAR, M., IRAVANI, R. "FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients", *IEEE Transactions on Power Delivery*, v. 25, n. 2, pp. 852–860, 2009.

[31] ▲ CHEN, Y., DINAVAHI, V. "FPGA-based real-time EMTP", *IEEE Transactions on Power Delivery*, v. 24, n. 2, pp. 892–902, 2008.

[32] ▲ CHEN, Y. "Large-scale real-time electromagnetic transient simulation of power systems using hardware emulation on FPGAs", 2012.

[33] ▲ STEVIC, M., ESTEBSARI, A., VOGEL, S., et al. "Multi-site European framework for real-time co-simulation of power systems", *IET Generation, Transmission & Distribution*, v. 11, n. 17, pp. 4126–4135, 2017.

[34] ▲ VOGEL, S. *Development of a modular and fully-digital PCIe-based interface to Real-Time Digital Simulator*. Tese de Doutorado, Master Thesis, RWTH Aachen University., 2016.

[35] ▲ MIRZ, M., VOGEL, S., REINKE, G., et al. "DPsim—A dynamic phasor real-time simulator for power systems", *SoftwareX*, v. 10, pp. 100253, 2019.

[36] ▲ KUFFEL, R., GIESBRECHT, J., MAGUIRE, T., et al. "RTDS-a fully digital power system simulator operating in real time". In: *Proceedings 1995 International Conference on Energy Management and Power Delivery EMPD'95*, v. 2, pp. 498–503. IEEE, 1995.

[37] ▲ VAN QUE, D., SOUMANGE, J., SYBILLE, G., et al. "Hypersim–an integrated real-time simulator for power networks and control systems". In: *Proc. Int. Conf. Digital Power System Simulators, Vasteras, Sweden*, 1999.

[38] ▲ PARÉ, D., TURMEL, G., SOUMAGNE, J., et al. "Validation tests of the hypersim digital real time simulator with a large AC-DC network". In: *Proc. Int. Conf. Power System Transients*, pp. 577–582, 2003.

[39] ▲ VEKIĆ, M. S., GRABIĆ, S. U., MAJSTOROVIĆ, D. P., et al. "Ultralow latency HIL platform for rapid development of complex power electronics systems", *IEEE Transactions on Power Electronics*, v. 27, n. 11, pp. 4436–4444, 2012.

[40] ▲ WG13, C. "The calculation of switching surges-II. Network representation for energisation and re-energisation studies on lines fed by an inductive source", *Electra*, v. 32, 1974.

[41] ▲ WG13, C. "The calculation of switching surges III. Transmission line representation for energisation and re-energisation studies with complex feeding networks", *Electra*, v. 62, pp. 45–78, 1979.

[42] ▲ HINGORANI, N. G., BURBERY, M. F. "Simulation of AC system impedance in HVDC system studies", *IEEE Transactions on power Apparatus and Systems*, , n. 5, pp. 820–828, 1970.

[43] ▲ MORCHED, A., BRANDWAJN, V. "Transmission network equivalents for electromagnetic transients studies", *IEEE transactions on power apparatus and systems*, , n. 9, pp. 2984–2994, 1983.

[44] ▲ GUSTAVSEN, B., SEMLYEN, A. "Rational approximation of frequency domain responses by vector fitting", *IEEE Transactions on power delivery*, v. 14, n. 3, pp. 1052–1061, 1999.

[45] ▲ GUSTAVSEN, B. "Improving the pole relocating properties of vector fitting", *IEEE Transactions on Power Delivery*, v. 21, n. 3, pp. 1587–1592, 2006.

[46] ▲ DESCHRIJVER, D., MROZOWSKI, M., DHAENE, T., et al. "Macromodeling of multiport systems using a fast implementation of the vector fitting method", *IEEE Microwave and wireless components letters*, v. 18, n. 6, pp. 383–385, 2008.

[47] ▲ BENNER, P. E., (ED.), W. S., (ED.), S. G.-T., et al. "Vector Fitting". In: *Model Order Reduction Volume 1: System- and Data-Driven Methods and Algorithms*, 1 ed., Berlin, Boston: De Gruyter, 2020.

[48] ▲ GUSTAVSEN, B., SEMLYEN, A. "Enforcing passivity for admittance matrices approximated by rational functions", *IEEE Transactions on Power Systems*, v. 16, n. 1, pp. 97–104, 2001.

[49] ▲ "Vector Fitting's Web Site". https://www.sintef.no/projectweb/vectfit/. Accessed: 2020-03-10.

[50] ▲ NODA, T. "Identification of a multiphase network equivalent for electromagnetic transient calculations using partitioned frequency response", *IEEE Transactions on Power Delivery*, v. 20, n. 2, pp. 1134–1142, 2005.

[51] ▲ CAMPELLO, T. M., VARRICCHIO, S. L., TARANTO, G. N., et al. "Enhancements in vector fitting implementation by using stopping criterion, frequency partitioning and model order reduction", *International Journal of Electrical Power & Energy Systems*, v. 120, pp. 105905, 2020.

[52] ▲ MORALES RODRIGUEZ, J. *Computation of Frequency Dependent Network Equivalents Using Vector Fitting, Matrix Pencil Method and Loewner Matrix*. Tese de Doutorado, Polytechnique Montréal, 2019.

[53] ▲ MORALES, J., MAHSEREDJIAN, J., RAMIREZ, A., et al. "A Loewner/MPM—VF combined rational fitting approach", *IEEE Transactions on Power Delivery*, v. 35, n. 2, pp. 802–808, 2019.

[54] ▲ MARTINS, N., PORTELA, C., GOMES, S. "Sequential computation of transfer function dominant poles of s-domain system models", *IEEE Transactions on Power Systems*, v. 24, n. 2, pp. 776–784, 2009.

[55] ▲ VARRICCHIO, S. L., FREITAS, F. D., MARTINS, N., et al. "Computation of dominant poles and residue matrices for multivariable transfer functions of infinite power system models", *IEEE Transactions on Power Systems*, v. 30, n. 3, pp. 1131–1142, 2014.

[56] ▲ SALVADOR, J. P., NETO, F. C., LIMA, A. C., et al. "Accuracy and Realization Issues in Frequency Dependent Sequence Networks", .

[57] ▲ SALVADOR, J. P. *Estudo de Equivalentes de Redes em Ampla Faixa de Frequência*. Tese de Doutorado, Dissertaçao de M. Sc., COPPE, 2014.

[58] ▲ HU, Y., WU, W., ZHANG, B. "Compacting and partitioning-based simulation solution for frequency-dependent network equivalents in real-time digital simulator", *IET Generation, Transmission & Distribution*, v. 9, n. 16, pp. 2526–2533, 2015.

[59] ▲ CAMARA, F., LIMA, A. C., MOREIRA, F. A. "A full frequency dependent line model based on folded line equivalencing and latency exploitation", *Electric Power Systems Research*, v. 154, pp. 352–360, 2018.

[60] ▲ CAMARA, F., LIMA, A. C. "A Multiple Time-Step Formulation of Frequency-Dependent Network Equivalents", *Journal of Control, Automation and Electrical Systems*, v. 29, n. 2, pp. 230–237, 2018.

[61] ▲ LIMA, A. C., CAMARA, F., SALVADOR, J. P., et al. "Frequency-dependent equivalent based on idempotent decomposition and grouping", *Electric Power Systems Research*, v. 189, pp. 106800, 2020.

[62] ▲ "The Vector Fitting Algorithm". In: *Passive Macromodeling*, cap. 7, pp. 225–306, John Wiley and Sons, Ltd, 2015. ISBN: 9781119140931. doi: https://doi.org/10.1002/9781119140931.ch7. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119140931.ch7>.

[63] ▲ ABDEL-RAHMAN, M. A.-A. H. *Frequency dependent hybrid equivalents of large networks*. Tese de Doutorado, National Library of Canada= Bibliothèque nationale du Canada, 2001.

[64] ▲ ABDEL-RAHMAN, M., SEMLYEN, A., IRAVANI, M. R. "Two-layer network equivalent for electromagnetic transients", *IEEE transactions on power delivery*, v. 18, n. 4, pp. 1328–1335, 2003.

[65] ▲ NIE, X., CHEN, Y., DINAVAHI, V. "Real-time transient simulation based on a robust two-layer network equivalent", *IEEE Transactions on Power Systems*, v. 22, n. 4, pp. 1771–1781, 2007.

[66] ▲ MATAR, M., IRAVANI, R. "A modified multiport two-layer network equivalent for the analysis of electromagnetic transients", *IEEE transactions on power delivery*, v. 25, n. 1, pp. 434–441, 2009.

[67] ▲ CLERC, B., MARTIN, C., DENNETIÈRE, S. "Implementation of accelerated models for EMT tools". In: *Proceedings of the International Conference On Power Systems Transients (IPST 2015)*, 2015.

[68] ▲ BRUNED, B., MARTIN, C., DENNETIÈRE, S., et al. "Implementation of a unified modelling between EMT tools for Network Studies". In: *Proceedings of the IPST conference, Seoul, Republic of Korea*, 2017.

[69] ▲ GUSTAVSEN, B., DE SILVA, H. J. "Inclusion of rational models in an electromagnetic transients program: Y-parameters, Z-parameters, S-parameters, transfer functions", *IEEE Transactions on Power Delivery*, v. 28, n. 2, pp. 1164–1174, 2013.

[70] ▲ DOMMEL, H. W. "Digital computer solution of electromagnetic transients in single-and multiphase networks", *IEEE transactions on power apparatus and systems*, , n. 4, pp. 388–399, 1969.

[71] ▲ GUSTAVSEN, B., SEMLYEN, A. "Simulation of transmission line transients using vector fitting and modal decomposition", *IEEE Transactions on Power Delivery*, v. 13, n. 2, pp. 605–614, 1998.

[72] ▲ CHEN, C.-T. *Linear system theory and design*. Oxford University Press, Inc., 1998.

[73] ▲ TAN, L., JIANG, J. *Digital signal processing: fundamentals and applications*. Academic Press, 2018.

[74] ▲ ÅSTRÖM, K. J., WITTENMARK, B. *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[75] ▲ XILINX. *7 Series FPGAs Configurable Logic Block User Guide (UG474)*. 2016.

[76] ▲ XILINX. *Vivado Design Suite Tutorial: High-Level Synthesis (UG781)*. 2014.

[77] ▲ OLIVEIRA, J. J., DE BARROS, H. M., MATTAR, S. A., et al. "Real-Time Cooperative Simulation Between RTDS and HYPERSIM, Test Results for the IEEE 39 Bus System", *XIV Symposium of Specialists in Electric Operational and Expansion Planning*, 2018.

[78] ▲ DE BARROS, H. M., DE OLIVEIRA, J., DICLER, F. "Implementation of Multiport Frequency Dependent Network Equivalents (FDNE) on FPGA Hardware for Efficient Representation of Very Large Power Systems in Real Time". https://www.opal-rt.com/resource-center/document/?resource=L00161_1246, 2020. [Online; accessed 04-October-2021].

[79] ▲ GRAINGER, J. J., STEVENSON, W. D. *Power system analysis*. 2003.

Appendix A

Vector Fitting

Given a set of p frequency responses $Z(s)$ whose values are known for k frequency points $\omega_1, \omega_2, \ldots, \omega_k$ such that:

$$Z_i(j\omega_k) = R_{i,f} + jX_{i,f} \quad , \quad i = 1, 2, \ldots, p; \quad f = 1, 2, \ldots, k \tag{A.1}$$

the transfer function synthesis objective is to find rational functions

$$\hat{Z}_i(s) = \frac{\beta_{i,0} + \beta_{i,1}s + \beta_{i,2}s^2 + \cdots + \beta_{i,n}s^n + \beta_{i,n+1}s^{n+1}}{1 + \alpha_1 s + \alpha_2 s^2 + \cdots + \alpha_n s^n}, \quad i = 1, 2, \cdots, p \tag{A.2}$$

which match the input frequency responses $Z(s)$ minimizing the quadratic mean error.

$$\hat{Z}_i\left(j\omega_f\right) \approx Z_i\left(j\omega_f\right), \quad i = 1, 2, \cdots, p; f = 1, 2, \cdots, k \tag{A.3}$$

The poles of $\hat{Z}_i(s)$ do not carry the i index because in this implementation of Vector Fitting, all fitted functions share the same set of poles, as discussed in the closure to [71].

Assuming that $p >> n$, it would be possible to rewrite Equation A.2 as an overdetermined linear problem of the type $Ax = b$ multiplying both sides with the denominator, and then solve it minimizing the quadratic mean error. However, due to the power series structure of the transfer function, the resulting problem is usually badly conditioned with the columns in A multiplying different powers of the frequency s.

In order to preclude the powers of s and to obtain a well conditioned overdetermined system to be solved in a quadratic mean square sense, a transfer function structure based on partial fractions was proposed.

$$\hat{Z}_i(s) = d_i + \sum_{k=1}^{n} \frac{c_{i,k}}{s - a_k} = h_0 \frac{\prod_{k=1}^{n+1} (s - z_{i,k})}{\prod_{k=1}^{n} (s - a_k)}, \quad i = 1, 2, \cdots, p \tag{A.4}$$

However, the fitting of Eq. A.3 with the partial fraction structure imposes a non-linear regression, which would demand iterative solvers subjected to convergence problems.

An alternative approach was then proposed, in which the fitting with partial fraction transfer function can be reduced to a linear regression. Instead of the approximation A.3, the following was proposed:

$$\sigma(j\omega_f)\, \hat{Z}_i(j\omega_f) \approx \sigma(j\omega_f)\, Z_i(j\omega_f), \quad i = 1, 2, \cdots, p; f = 1, 2, \cdots, k \tag{A.5}$$

in which both equation sides are multiplied by a rational function $\sigma(s)$:

$$\sigma(s) = 1 + \sum_{k=1}^{n} \frac{\tilde{c}_k}{s - \overline{a}_k} = \frac{\prod_{k=1}^{n} (s - y_k)}{\prod_{k=1}^{n} (s - \overline{a}_k)} \tag{A.6}$$

in which the poles $\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_k$ are initially arbitrarily chosen.

The parameters $\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_k$ are fitted to satisfy the condition

$$\sigma(s)\hat{Z}_i(s) = d_i + \sum_{k=1}^{n} \frac{c_{i,k}}{s - \overline{a}_k}, \quad i = 1, 2, \cdots, p \tag{A.7}$$

The transfer function resulted from the multiplication $\sigma(s)\hat{Z}_i(s)$ has the same poles of $\sigma(s)$ itself. This is only possible if the zeros of $\sigma(s)$ cancel the poles of $\hat{Z}_i(s)$, which are the unknowns to be obtained. Then, if the Equation A.7 is imposed to be valid, obtaining the zeros of $\sigma(s)$ solves the problem.

Applying the definition A.6 in the alternative fitting of Equation A.5 yields the following relation (the index i is omitted to simplify the notation):

$$d + \sum_{k=1}^{n} \frac{c_k}{s - \overline{a}_k} = \left(1 + \sum_{k=1}^{n} \frac{\tilde{c}_k}{s - \overline{a}_k}\right) Z(s) \tag{A.8}$$

Equation A.8 is linear, with unknowns c_k, d, $\tilde{c}_k$. By picking sufficiently frequency points yields an overdetermined linear problem of the type $Ax = b$ whose solution gives the complete representation of $\sigma(s)$, but in partial fraction structure. The zeros can be obtained as the eigenvalues of

$$H = A - b\tilde{c}^T \tag{A.9}$$

where A is a diagonal matrix with the starting poles, b is a column vector of ones,

$\tilde{c}^T$ is a row vector with the calculated residues of $\sigma(s)$.

Using the obtained set of poles $a_1, a_2, \ldots, a_k$ as starting poles in an iterative procedure gives more accurate solutions.

Appendix B

Modal Decomposition

The modal decomposition consists of decoupling a three-phase circuit into three decoupled one-phase systems, corresponding to two aerial modes and one ground mode, as described in the Equation B.1 [79].

$$\hat{V} = S^{-1}V, \quad \hat{I} = S^{-1}I, \quad \hat{Z} = S^{-1}ZS \tag{B.1}$$

where

$$S = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \alpha^2 & \alpha \\ 1 & \alpha & \alpha^2 \end{bmatrix}, \quad S^{-1} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 \\ 1 & \alpha^2 & \alpha \end{bmatrix}, \quad \alpha = e^{j\frac{2}{3}\pi} = -0.5 + j\sqrt{3/2} \tag{B.2}$$

The matrix $\hat{Z} = S^{-1}ZS$ is block diagonal:

$$\hat{Z} = \begin{bmatrix} z^0 & 0 & 0 \\ 0 & z^1 & 0 \\ 0 & 0 & z^2 \end{bmatrix} \tag{B.3}$$

where z^0, z^1 and z^2 are zero, positive and negative sequence impedances, respectively.

In the general case, the transformation matrix contains complex elements. However, for balanced three-phase networks composed exclusively of transposed transmission lines and passive elements, the self impedances related to each phase will be equal and the mutual impedances between phases will also be equal, so that

$$Z = \begin{bmatrix} Z_{\text{self}} & Z_{\text{mutual}} & Z_{\text{mutual}} \\ Z_{\text{mutual}} & Z_{\text{self}} & Z_{\text{mutual}} \\ Z_{\text{mutual}} & Z_{\text{mutual}} & Z_{\text{self}} \end{bmatrix} \tag{B.4}$$

Then, the zero, positive and negative sequence impedances are given by

$$z^0 = z_{\text{self}} + 2z_{\text{mutual}} \tag{B.5}$$

$$z^1 = z^2 = z_{\text{self}} - z_{\text{mutual}} \tag{B.6}$$

which can be obtained by an alternative matrix transformation containing only real elements:

$$\hat{V} = M^{-1}V, \quad \hat{I} = M^{-1}I, \quad \hat{Z} = M^{-1}ZM = \begin{bmatrix} z^0 & 0 & 0 \\ 0 & z^1 & 0 \\ 0 & 0 & z^1 \end{bmatrix} \tag{B.7}$$

where

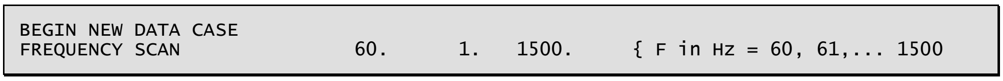
$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix}, \quad M^{-1} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \tag{B.8}$$

Appendix C

Frequency Scan Card configuration in ATP

To obtain the frequency response of the nodal impedance matrix using the ATP tool, several cases must be executed, corresponding to the injection of positive and zero sequence currents.

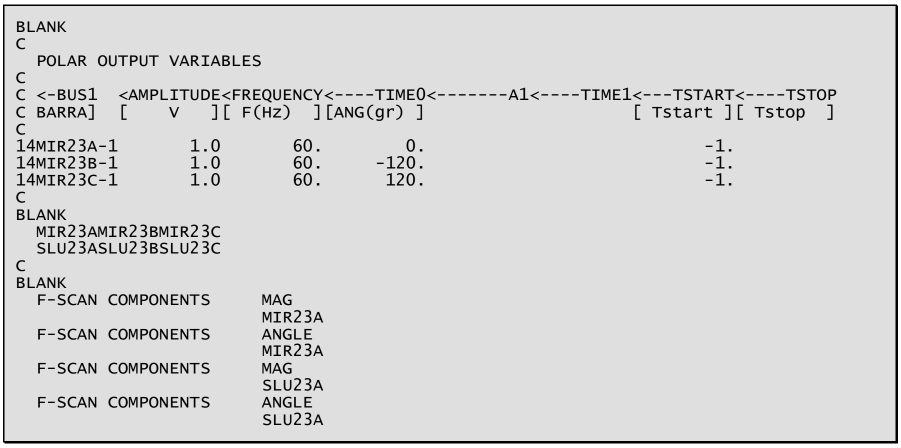
In all cases, the configuration must be done by the FREQUENCY SCAN card, which receives the range of frequencies of interest for the studies to be carried out with the equivalent model, as exposed in the Figure C.1.



```
BEGIN NEW DATA CASE
FREQUENCY SCAN               60.      1.   1500.     { F in Hz = 60, 61,... 1500
```

Figure C.1: Header of the Frequency Scan card.

For two ports equivalents, four cases must be executed. Figures C.2 and C.1 show how the cases must be configured for injecting positive and zero sequence currents in the MIR23 bus, respectively.



```
BLANK
C
  POLAR OUTPUT VARIABLES
C
C <-BUS1  <AMPLITUDE<FREQUENCY<----TIME0<-------A1<----TIME1<---TSTART<----TSTOP
C BARRA]  [    V   ][ F(Hz)  ][ANG(gr) ]                     [ Tstart ][ Tstop  ]
C
14MIR23A-1       1.0       60.        0.                            -1.
14MIR23B-1       1.0       60.     -120.                            -1.
14MIR23C-1       1.0       60.      120.                            -1.
C
BLANK
  MIR23AMIR23BMIR23C
  SLU23ASLU23BSLU23C
C
BLANK
  F-SCAN COMPONENTS     MAG
                        MIR23A
  F-SCAN COMPONENTS     ANGLE
                        MIR23A
  F-SCAN COMPONENTS     MAG
                        SLU23A
  F-SCAN COMPONENTS     ANGLE
                        SLU23A
```

Figure C.2: Positive sequence current injection at MIR23 bus.

```
BLANK
C
  POLAR OUTPUT VARIABLES
C
C <-BUS1  <AMPLITUDE<FREQUENCY<----TIME0<-------A1<----TIME1<---TSTART<----TSTOP
C BARRA]  [    V   ][ F(Hz)  ][ANG(gr) ]                    [ Tstart ][ Tstop  ]
C
14MIR23A-1       1.0       60.        0.                           -1.
14MIR23B-1       1.0       60.        0.                           -1.
14MIR23C-1       1.0       60.        0.                           -1.
C
BLANK
  MIR23AMIR23BMIR23C
  SLU23ASLU23BSLU23C
C
BLANK
  F-SCAN COMPONENTS     MAG
                        MIR23A
  F-SCAN COMPONENTS     ANGLE
                        MIR23A
  F-SCAN COMPONENTS     MAG
                        SLU23A
  F-SCAN COMPONENTS     ANGLE
                        SLU23A
```

Figure C.3: Zero sequence current injection at MIR23 bus.

Figures C.4 and C.5 show how the cases must be configured for injecting positive and zero sequence currents in the SLU23 bus, respectively.

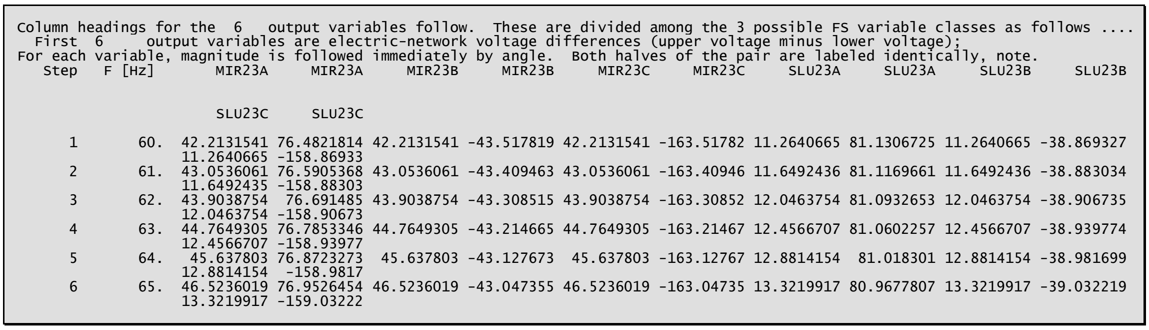
```
BLANK
C
  POLAR OUTPUT VARIABLES
C
C <-BUS1  <AMPLITUDE<FREQUENCY<----TIME0<-------A1<----TIME1<---TSTART<----TSTOP
C BARRA]  [    V   ][ F(Hz)  ][ANG(gr) ]                    [ Tstart ][ Tstop  ]
C
14SLU23A-1       1.0       60.        0.                           -1.
14SLU23B-1       1.0       60.     -120.                           -1.
14SLU23C-1       1.0       60.      120.                           -1.
C
BLANK
  MIR23AMIR23BMIR23C
  SLU23ASLU23BSLU23C
C
BLANK
  F-SCAN COMPONENTS     MAG
                        MIR23A
  F-SCAN COMPONENTS     ANGLE
                        MIR23A
  F-SCAN COMPONENTS     MAG
                        SLU23A
  F-SCAN COMPONENTS     ANGLE
                        SLU23A
```

Figure C.4: Positive sequence current injection at SLU23 bus.

```
BLANK
C
  POLAR OUTPUT VARIABLES
C
C <-BUS1  <AMPLITUDE<FREQUENCY<----TIME0<-------A1<----TIME1<---TSTART<----TSTOP
C BARRA]  [    V   ][ F(Hz)  ][ANG(gr) ]                    [ Tstart ][ Tstop  ]
C
14SLU23A-1       1.0       60.        0.                           -1.
14SLU23B-1       1.0       60.        0.                           -1.
14SLU23C-1       1.0       60.        0.                           -1.
C
BLANK
  MIR23AMIR23BMIR23C
  SLU23ASLU23BSLU23C
C
BLANK
  F-SCAN COMPONENTS     MAG
                        MIR23A
  F-SCAN COMPONENTS     ANGLE
                        MIR23A
  F-SCAN COMPONENTS     MAG
                        SLU23A
  F-SCAN COMPONENTS     ANGLE
                        SLU23A
```

Figure C.5: Zero sequence current injection at SLU23 bus.

After executing the cases in ATP, an output report with ".LIS" extension is generated for each sequence, containing a voltage phasor (module and angle) for each bus phase, for each frequency within the scanned range, as shown in the Figure C.6.



```
Column headings for the  6   output variables follow.  These are divided among the 3 possible FS variable classes as follows ....
  First  6     output variables are electric-network voltage differences (upper voltage minus lower voltage);
For each variable, magnitude is followed immediately by angle.  Both halves of the pair are labeled identically, note.
   Step   F [Hz]     MIR23A     MIR23A     MIR23B     MIR23B     MIR23C     MIR23C     SLU23A     SLU23A     SLU23B     SLU23B


                     SLU23C     SLU23C

      1      60. 42.2131541 76.4821814 42.2131541 -43.517819 42.2131541 -163.51782 11.2640665 81.1306725 11.2640665 -38.869327
                 11.2640665 -158.86933
      2      61. 43.0536061 76.5905368 43.0536061 -43.409463 43.0536061 -163.40946 11.6492436 81.1169661 11.6492436 -38.883034
                 11.6492435 -158.88303
      3      62. 43.9038754  76.691485 43.9038754 -43.308515 43.9038754 -163.30852 12.0463754 81.0932653 12.0463754 -38.906735
                 12.0463754 -158.90673
      4      63. 44.7649305 76.7853346 44.7649305 -43.214665 44.7649305 -163.21467 12.4566707 81.0602257 12.4566707 -38.939774
                 12.4566707 -158.93977
      5      64.  45.637803 76.8723273  45.637803 -43.127673  45.637803 -163.12767 12.8814154  81.018301 12.8814154 -38.981699
                 12.8814154  -158.9817
      6      65. 46.5236019 76.9526454 46.5236019 -43.047355 46.5236019 -163.04735 13.3219917 80.9677807 13.3219917 -39.032219
                 13.3219917 -159.03222
```

Figure C.6: Output report.

Appendix D

Discrete-Time Model Synthesis

The transfer function which results from the Vector Fitting routine has the following state-space realization (The matrices A,B,C and D were previously defined on the Section 2.2):

$$\begin{cases} \dot{X}(t) = AX(t) + BU(t) \\ Y(t) = CX(t) + DU(t) \end{cases} \tag{D.1}$$

where

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} \quad ; \quad U = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_p \end{bmatrix}$$

$$X_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad ; \quad Y = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \end{bmatrix}$$

where n is the number of poles.

Equation D.1 can be rewritten in its integral form:

$$\begin{cases} X(t) = X(t-\Delta t) + \int_{t-\Delta t}^{t} (AX(\tau) + BU(\tau))\, d\tau \\ EU(t) = EU(t-\Delta t) + \int_{t-\Delta t}^{t} (Y(\tau) - CX(\tau) - DU(\tau))\, d\tau \end{cases} \tag{D.2}$$

Applying the Trapezoidal Method to D.2 yields

$$\begin{cases} X(t) = & X(t-\Delta t) + \dfrac{\Delta t}{2}\left(AX(t) + BU(t)\right) \\ & + \dfrac{\Delta t}{2}\left(AX(t-\Delta t) + BU(t-\Delta t)\right) \\ EU(t) = & EU(t-\Delta t) + \dfrac{\Delta t}{2}\left(Y(t) - CX(t) - DU(t)\right) \\ & + \dfrac{\Delta t}{2}\left(Y(t-\Delta t) - CX(t-\Delta t) - DU(t-\Delta t)\right) \end{cases}$$

rearranging:

$$\begin{cases} \left(I - \dfrac{\Delta t}{2}A\right)X(t) = \left(I + \dfrac{\Delta t}{2}A\right)X(t-\Delta t) + \dfrac{\Delta t}{2}B\left(U(t-\Delta t) + U(t)\right) \\ \dfrac{2}{\Delta t}EU(t) = \dfrac{2}{\Delta t}EU(t-\Delta t) + Y(t) - CX(t) - DU(t) \\ \qquad + Y(t-\Delta t) - CX(t-\Delta t) - DU(t-\Delta t) \end{cases}$$

Solving for $X(t)$ and $Y(t)$ yields:

$$\begin{cases} X(t) = \left(I - \dfrac{\Delta t}{2}A\right)^{-1}\left(I + \dfrac{\Delta t}{2}A\right)X(t-\Delta t) \\ \qquad + \dfrac{\Delta t}{2}\left(I - \dfrac{\Delta t}{2}A\right)^{-1}B\left(U(t-\Delta t) + U(t)\right) \\ Y(t) = -Y(t-\Delta t) + CX(t) + CX(t-\Delta t) \\ \qquad + \left(D - \dfrac{2}{\Delta t}E\right)U(t-\Delta t) + \left(D + \dfrac{2}{\Delta t}E\right)U(t) \end{cases}$$

rearranging:

$$\begin{cases} X(t) = \hat{A}X(t-\Delta t) + \hat{B}U(t-\Delta t) + \hat{B}U(t) \\ Y(t) = -Y(t-\Delta t) + CX(t) + CX(t-\Delta t) \\ \qquad + \left(D - \dfrac{2}{\Delta t}E\right)U(t-\Delta t) + \left(D + \dfrac{2}{\Delta t}E\right)U(t) \end{cases}$$

where

$$\hat{A} = \left(I - \frac{\Delta t}{2}A\right)^{-1}\left(I + \frac{\Delta t}{2}A\right)$$
$$\hat{B} = \left(I - \frac{\Delta t}{2}A\right)^{-1}\frac{\Delta t}{2}B$$

Replacing $X_m(t)$ in the expression of $Y_m(t)$ yields:

$$\begin{cases} X(t) =\hat{A}X(t-\Delta t) + \hat{B}U(t-\Delta t) + \hat{B}U(t) \\ Y(t) = -Y(t-\Delta t) + \left(C\hat{A} + C\right) X(t-\Delta t) \\ \qquad + \left(C\hat{B} + D - \dfrac{2}{\Delta t}E\right) U(t-\Delta t) \\ \qquad + \left(C\hat{B} + D + \dfrac{2}{\Delta t}E\right) U(t) \end{cases}$$

The voltage at terminal buses $Y(t)$ can be decomposed into a historic component $Y_{Hm}(t)$ and an instantaneous component proportional to the input $U(t)$:

$$\begin{cases} X(t) =\hat{A}X(t-\Delta t) + \hat{B}U(t-\Delta t) + \hat{B}U(t) \\ Y_H(t) = -Y(t-\Delta t) + \left(C\hat{A} + C\right) X(t-\Delta t) \\ \qquad + \left(C\hat{B} + D - \dfrac{2}{\Delta t}E\right) U(t-\Delta t) \\ Y(t) =Y_H(t) + \left(C\hat{B} + D + \dfrac{2}{\Delta t}E\right) U(t) \end{cases}$$

Replacing $-Y_m(t-\Delta t)$ for the expression of $Y_m(t)$ yields:

$$\begin{cases} X(t) = \hat{A}X(t-\Delta t) + \hat{B}U(t-\Delta t) + \hat{B}U(t) \\ Y_H(t) = -Y_H(t-\Delta t) + \left(C\hat{A} + C\right) X(t-\Delta t) - \dfrac{4}{\Delta t}EU(t-\Delta t) \\ Y(t) = Y_H(t) + \left(C\hat{B} + D + \dfrac{2}{\Delta t}E\right) U(t) \end{cases}$$

Finally, the difference equations to be inserted in the time-domain simulator to be solved at each time-step are:

$$\begin{cases} X(t) = \hat{A}X(t-\Delta t) + \hat{B}U(t-\Delta t) + \hat{B}U(t) \\ Y_H(t) = -Y_H(t-\Delta t) + \hat{C}X(t-\Delta t) \\ Y(t) = Y_H(t) + \hat{D}U(t) \end{cases} \tag{D.3}$$

where

$$\begin{aligned} \hat{A} &= \left(I - \frac{\Delta t}{2}A\right)^{-1} \left(I + \frac{\Delta t}{2}A\right) \\ \hat{B} &= \left(I - \frac{\Delta t}{2}A\right)^{-1} \frac{\Delta t}{2}B \\ \hat{C} &= C\hat{A} + C \\ \hat{D} &= C\hat{B} + D \end{aligned}$$

Appendix E

RTDS results

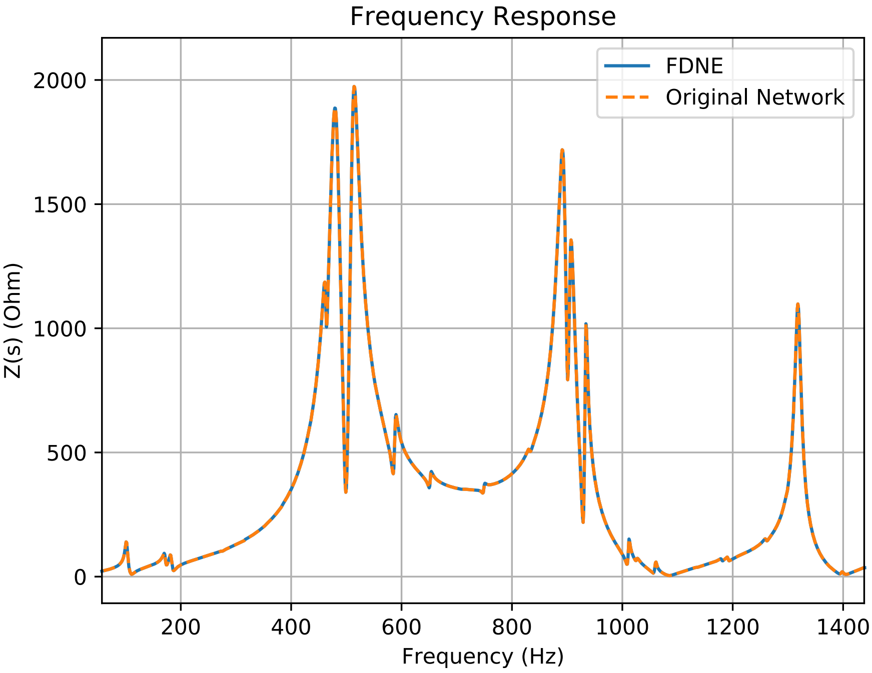


Figure E.1: Z_{12} amplitude results.

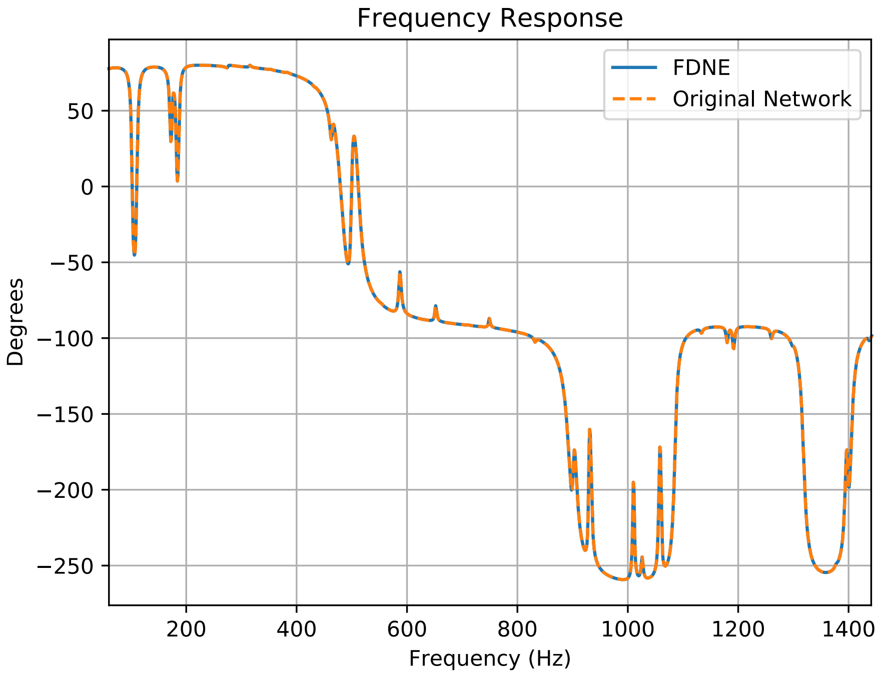


Figure E.2: Z_{12} phase results.

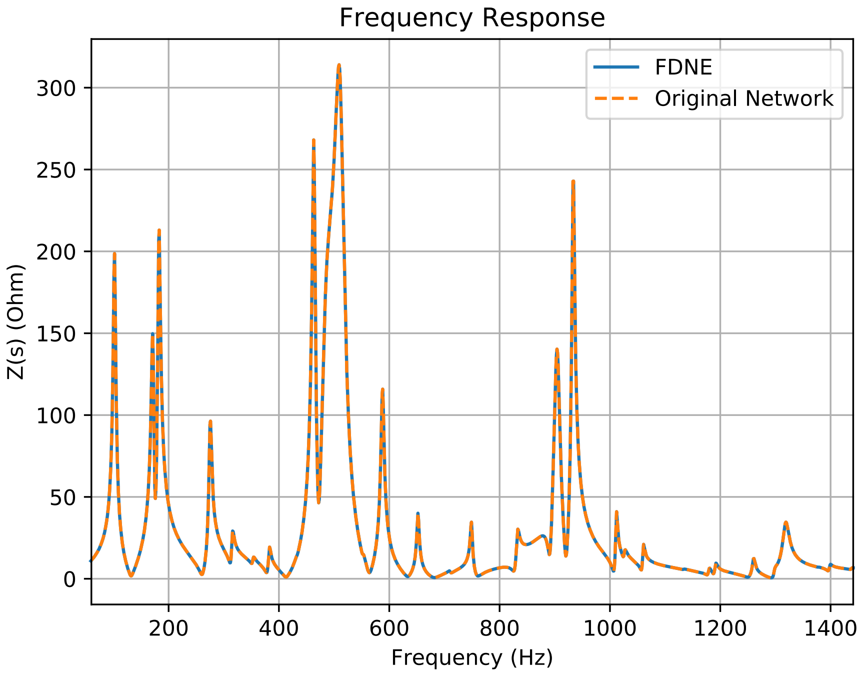


Figure E.3: Z_{13} amplitude results.

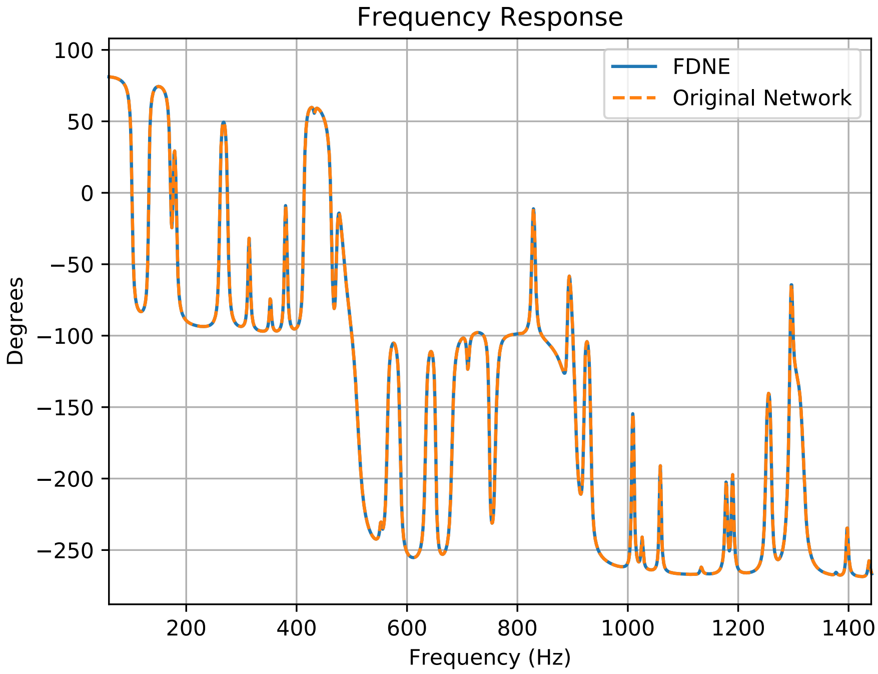


Figure E.4: Z_{13} phase results.

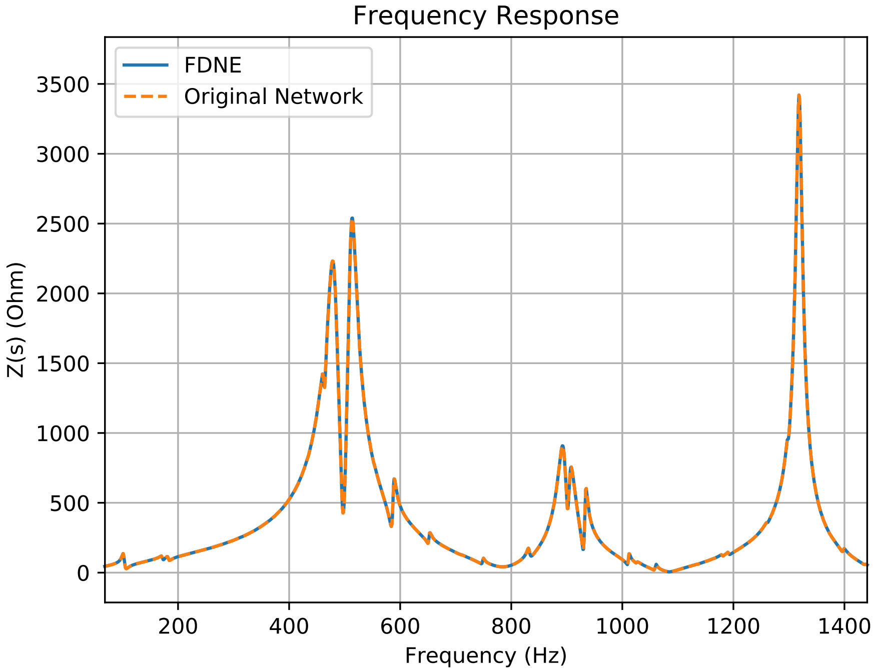


Figure E.5: Z_{22} amplitude results.

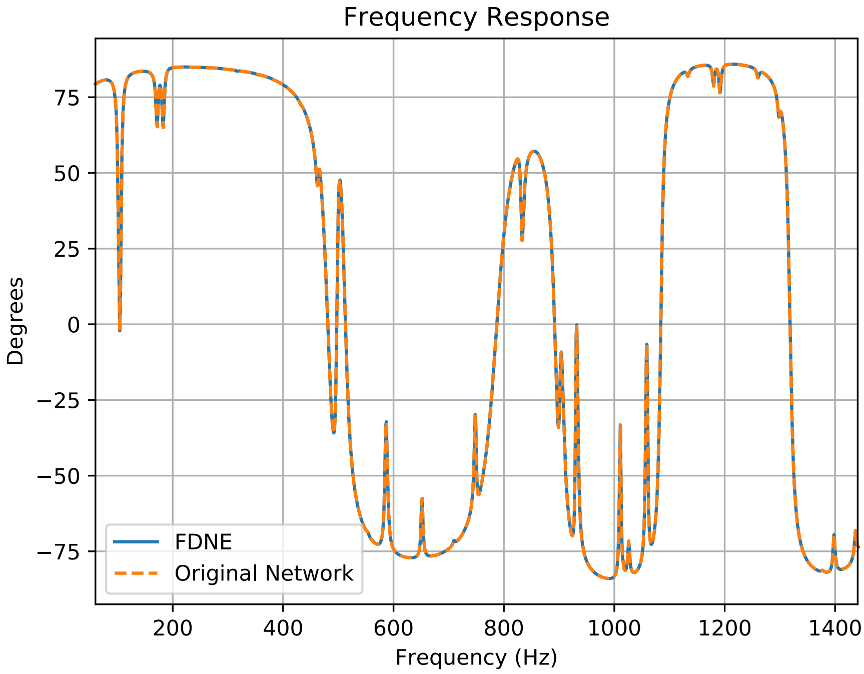


Figure E.6: Z_{22} phase results.

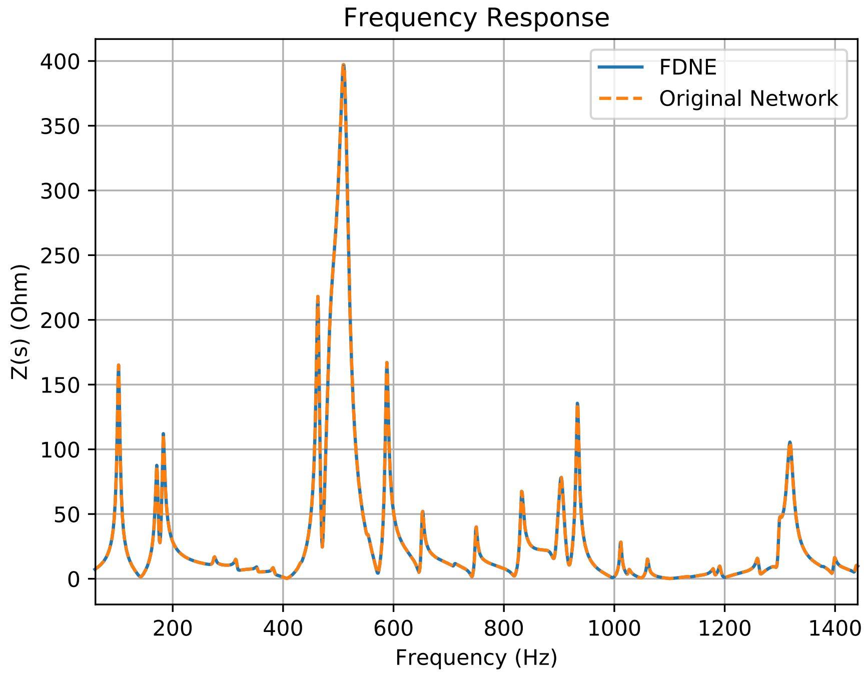


Figure E.7: Z_{23} amplitude results.

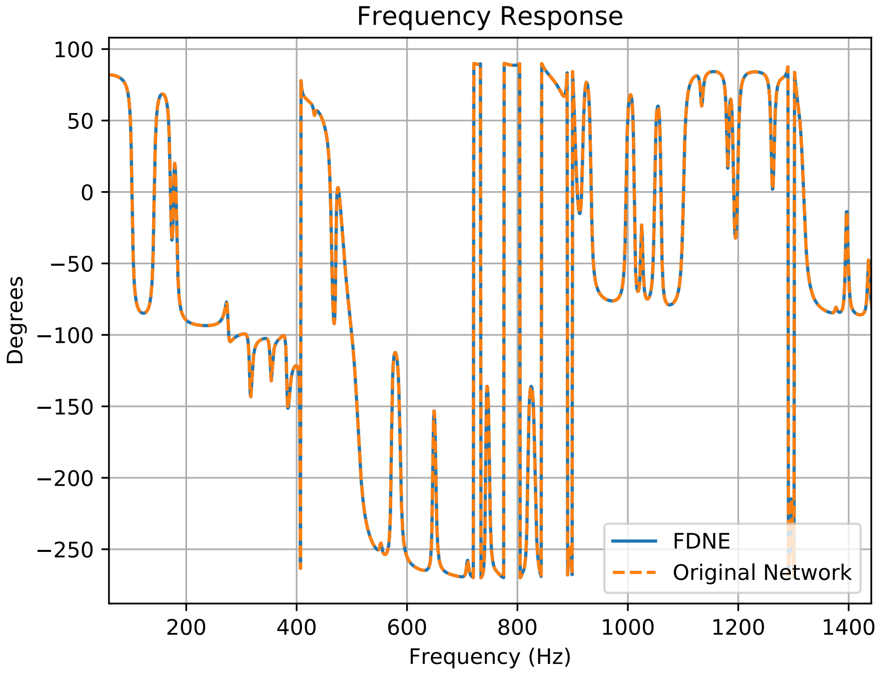


Figure E.8: Z_{23} phase results.

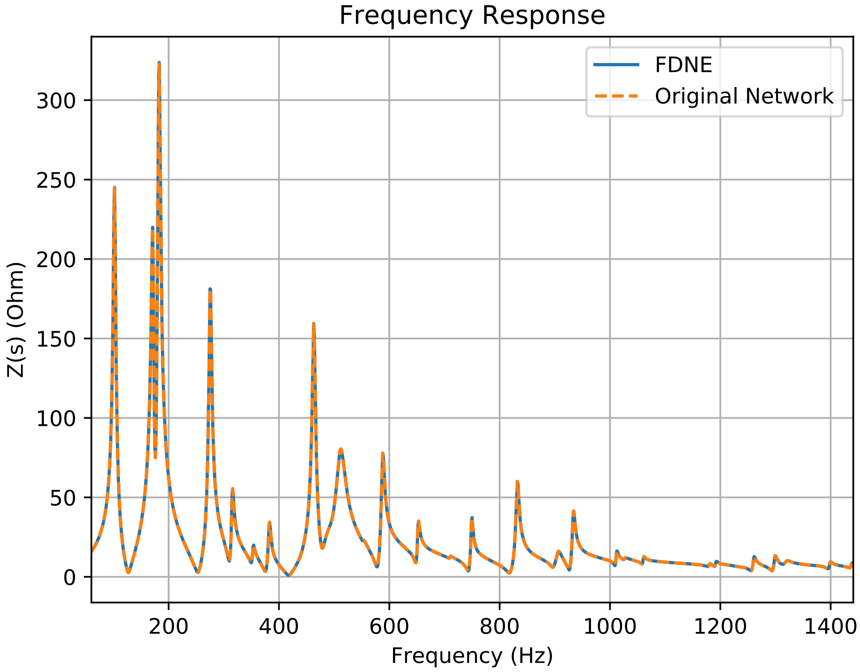


Figure E.9: Z_{33} amplitude results.

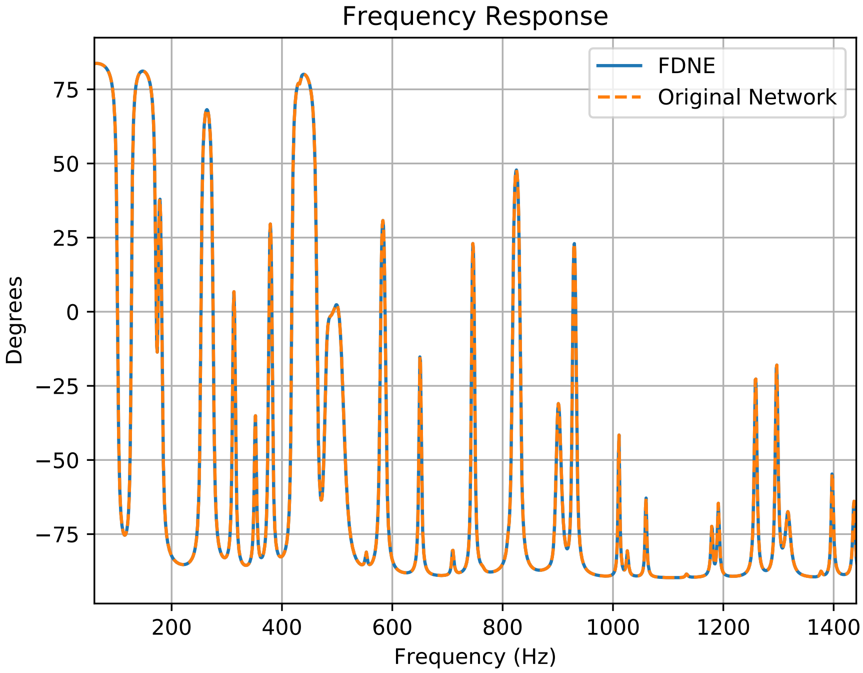


Figure E.10: Z_{33} phase results.