HAL
open science

Actor's based diakoptics for the simulation, monitoring and control of smart grids

Davis Montenegro Martinez

To cite this version:

Davis Montenegro Martinez. Actor's based diakoptics for the simulation, monitoring and control of smart grids. Electric power. Université Grenoble Alpes; Universidad de los Andes (Bogotá), 2015. English. NNT : 2015GREAT106 . tel-01260398

HAL Id: tel-01260398

https://theses.hal.science/tel-01260398v1

Submitted on 22 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

préparée dans le cadre d'une cotutelle entre l'*Université Grenoble Alpes* et L'Universidad de los Andes

Spécialité : **Génie Electrique**

Arrêté ministériel : le 6 janvier 2005 - 7 août 2006

Présentée par

Davis / MONTENEGRO MARTINEZ

Thèse dirigée par **Seddik/BACHA**
codirigée par **Gustavo Andrés/RAMOS LOPEZ**

préparée au sein des **Laboratoire G2ELAB**

dans **les Écoles Doctorales l'École Doctorale Electronique, Electrotechnique, Automatique, Télécommunication et Traitement du Signal et le département du génie électrique et électronique de L'Universidad de los Andes**

Diakoptics basés sur les acteurs pour la simulation, la surveillance et la commande des réseaux intelligents

Thèse soutenue publiquement le **« 19 Novembre 2015 »**,
devant le jury composé de :

M, Daniel, HISSEL
Professeur à l'Université de Franche Comté, Président
M, Damien, TROMEUR
Professeur à l'Université Claude Bernard Lyon 1, Examinateur
M, Gilney, DAMM
Maître de conférence à l'Université d'Evry-Val-d 'Essonne, Examinateur
M, Mario Alberto, RIOS MESIAS
Professeur à L'Universidad de los Andes, Examinateur
M, Abdelkrim, BENCHAIB
Supergrid Institute (GE Grid Solutions) HDR\Professeur CNAM, Rapporteur
M, Ionel, VECHIU
Professeur \HDR à l'ESTIA, Rapporteur.
M, Seddik, BACHA
Professeur à l'Université Grenoble Alpes, G2Elab, Directeur de thèse
M, Gustavo Andres, RAMOS-LOPEZ
Professeur à L'Universidad de los Andes, Directeur de thèse
M, Roger, DUGAN
Sr Exécutif technique, EPRI, Invité

Acknowledgments

This Thesis is called Diakoptics based on actors for the simulation, control and monitoring of smart grid applications was developed in cotutel with the « Laboratoire de Génie Electrique de Grenoble » (G2ELab), France, and the Universidad de los Andes, Colombia.

First of all I want to say thanks to god the all mighty, because without him we are nothing and because he gave to my family and to me the strong and discipline to grow personally and intellectually during the development of this thesis.

I want also say thanks to my lovely wife: Luisa, you are a very important part on this process, your love, support and patience make possible all our goals as individuals and family. Also thanks for our beautiful son, who has arrived in the middle of this wonderful experience and gave us a new perspective of life, making us to realize as parents and human beings. This thanks are extensive to my parents Flor and Jorge and to my brothers Giovanny, Gino and Giorgio: guys, without your help, many of the things that I have reached in my life would not be possible.

I want to say thanks to my Colombian advisor, Mr. Gustavo Ramos, he has become a friend giving to me the opportunity to speak and propose ideas in a changing world. Gustavo, thanks for your support and for believe in this project since the beginning; together, we have passed a lot of adventures defending our ideas and proposing new paths to follow for the power industry and to leave our little mark in the world, thanks a lot.

Thanks a lot too to my French advisor, Mr. Seddik Bacha, who received me since my first moment in France as a father. Seddik, without your support, we maybe could not show the potential of our development and without your advice, I would not be able to understand that we are here not only for being the best, but for serving to the people around us with our talents and ideas.

I want to especially acknowledge to Mr. Roger Dugan and Mr. Mark McGranaghan and to the amazing team of the Electric Power Research Institute EPRI, all of you have seeded in me the need for going beyond of what it is established in the power industry, you also gave me the opportunity of being part of your team for a short period, but at the same time showed me the amazing and passionate world of research in this area. Roger, thanks a lot for believe in this initiative and for supporting us with this amazing tool called OpenDSS, which has served for supporting a good number of developments around the world and to give new ideas and innovative paths to engineers worldwide. Mark, you have been also an important and fundamental support for this project for believing is us and for giving us the opportunity to show what we are capable to do. Thanks a lot.

I want to thank to the Universidad Santo Tomas for supporting my formation during this period, particularly, I want to say thanks to the Engineer Adriana Cecilia Paez (Jefe), who has supported this project by giving me the time to finish it and for letting me to be part of the great group of the faculty of Electronics Engineering of this prestigious University. Also, I want to thank to the lab's working team: Giovanni, William, Juancho, Javier, Jason and of course, Leonel Giraldo, to the university's directives and the administrative personnel.

I want to say thanks to all the amazing people I have met during the development of this project. To my friends in the different parts of the world: Julian Fernandez, Jose L. Sanchez, Mariam Ahmed (and

little Luna), Nathalie Sette, Raha Vafaei, Kalle Rauma, Khaled Aslan, Andres Ovalle, Miguel Hernandez, David Celeita, Jason Taylor, Jeff Smith, Gustavo Romero, Fernando Romero, Freddy Garcia, thanks for being such a nice persons with me and my family.

Finally, I want to thank to the Colombian government for supporting this thesis through the program "Becas para doctorados en Colombia" and the "Convocatoria 528, Colciencias". Also, this acknowledgment is extensive to the working team of Colfuturo for their management talent during the development of this thesis.

Dear reader, I wish you a good and nice reading.

Contents

GENERAL INTRODUCTION **12**

CHAPTER 1. INTRODUCTION TO POWER SYSTEMS SIMULATION IN REAL-TIME **13**

CHAPTER 1 INTRODUCTION TO POWER SYSTEMS SIMULATION IN REAL-TIME **14**

SIMULATION FIDELITY 15
Electromagnetic Transients (EMT) simulation *15*
Load flow simulation (phasor simulation) *16*
Hybrid simulation (phasor and EMT simulation) *17*
SIMULATORS AND HARDWARE 17
Analog simulators *18*
Digital simulators *19*
Hybrid simulators *20*
REAL-TIME SIMULATION AND ITS EVOLUTION 21
Addressing real life scenarios through Real-Time simulation *21*
Power-Hardware-In-the-Loop simulators (PHIL) *23*
Controller-Hardware-In-the-Loop simulators (CHIL) *23*
CHALLENGES WHEN SIMULATING REAL-LIFE SCENARIOS 25
The evolution of computing systems *25*
Multithread and vectorization *27*
THE SCOPE OF THIS THESIS 28
CONCLUSIONS 29

CHAPTER2. SIMULATION OF DISTRIBUTION POWER SYSTEMS FOR SHAPING THE FUTURE SMART GRID **30**

CHAPTER 2 ..... **31**

SIMULATION FOR THE DEVELOPMENT OF THE FUTURE SMART GRID 31
THE SOLUTION METHODS FOR LOAD FLOW ANALYSIS IN DS 32
Power flow analysis based on phase frame *32*
Power flow analysis based on sequence frame *36*
Dynamic Simulation *38*
CHALLENGES FOR DS SIMULATION 41
The deterministic computing cycles *41*
Complexity of the simulation *41*
Topology changes *42*
Hybrid power systems *42*
CONCLUSION 42

CHAPTER3. DISTRIBUTION SYSTEM LOAD FLOW ANALYSIS USING DIAKOPTICS **43**

CHAPTER 3 ..... **44**

DISTRIBUTION SYSTEM LOAD FLOW ANALYSIS USING DIAKOPTICS 44
Diakoptics and the piecewise methods *44*
The primitive networks *44*
The orthogonal networks *45*
The interconnected equivalent network *50*
THE MULTILEVEL APPROACH 52
Separating the Connections matrix Z_{CC} *52*
Distributing tasks *54*
Simulation of Hybrid power systems *55*
THE COMPUTATIONAL MODEL OF DIAKOPTICS 57
Creating primitive networks (Islands) *58*
Distributed computation *64*
CONCLUSIONS 65

CHAPTER 4. THE ACTOR MODEL FOR HANDLING PARALLEL AND CONCURRENT COMPUTING 66

CHAPTER 4 67

INTRODUCTION TO THE ACTOR MODEL 67
Homogeneous and Heterogeneous computing systems *68*
Actors and Agents *70*
BUILDING AN ACTOR'S SYSTEM 71
The actor structure *71*
The actor messages *73*
The management of the hardware resources *74*
The standard PC architecture *74*
The Real-Time Architecture *75*
ACTOR'S ENVIRONMENT PROPOSED FOR IMPLEMENTING A-DIAKOPTICS 76
The parent actor architecture *78*
The child actor architecture *80*
The messages architecture *82*
CONCLUSIONS 82

CHAPTER 5. APPLICATIONS OF A-DIAKOPTICS 84

CHAPTER 5 85

APPLICATIONS OF A-DIAKOPTICS 85
THE SIMULATION OF DISTRIBUTION POWER SYSTEMS 85
Distribution System Simulation using Standard PC architectures *86*
The harmonics simulation mode *88*
The sequential-time TSA *90*
Performance of the PC application *91*
Distribution System Simulation using Real-Time architectures *100*
OTHER FUTURE APPLICATIONS 105
DS state estimation *105*
Advanced Distribution Automation *108*
GENERAL CONCLUSIONS 111
REFERENCES 113

ANNEXES AND COMPLEMENTARY MATERIAL……126

List of Figures

Figure 1-1. Cost trend when detecting bugs in different stages of a product development. 14
Figure 1-2. Simulated phenomena using EMT simulation (based on the graph provided in (Watson et al., 2003)) 15
Figure 1-3. Simulated phenomena using phasor simulation (based on the graph provided in (Watson et al., 2003)) 16
Figure 1-4. 39-bus New England AC system with two HVDC links (Xi et al., 2009) 17
Figure 1-5. Test system implemented using an analog simulator (Imai et al., 2002)..... 18
Figure 1-6. Hybrid simulator example 20
Figure 1-7. Uses of Real-time simulators for addressing real world applications. 22
Figure 1-8. HIL Power system simulation categories. 24
Figure 1-9. Classical sequential computing systems 25
Figure 1-10. Multicore computing systems 26
Figure 1-11. Application fields of parallel computing (Jainschigg, 2012)..... 27
Figure 2-1. Solution methods used for power flow analysis in DS 38
Figure 2-2. Dynamic behavior of voltage (p.u.) after the disconnection and reconnection of a 1MW generator from EPRI's circuit 7- 2452 nodes. Step time 4.5 ms. Graph generated with DSSim-PC (Montenegro, 2013). 41
Figure 3-1. Link branch of an electrical system 45
Figure 3-2. System under study 45
Figure 3-3. Y_{BUS} matrix for modeling the proposed system 46
Figure 3-4. Y_{BUS} matrix after switch SW_1 trips 46
Figure 3-5. Y_{BUS} reorganized after switch SW_1 tripping 47
Figure 3-6. Transposed contour matrix in the case presented in Figure 3-5 48
Figure 3-7. Example of reference frame moving using a transformation matrix. In this case, matrix A is the transformation matrix for taking matrix B from the domain YY to the domain XX. 49
Figure 3-8. Equivalent Z_{TT} matrix obtained from the tree admittance matrix Y_{TT} 50
Figure 3-9. The proposed procedure for solving the interconnected equivalent (physical interpretation) 50
Figure 3-10. Two power systems represented using their interconnected equivalents in Diakoptics. F1 and F2 means that both systems are operating at different base frequencies. 56
Figure 3-11. Two power systems working at different base frequencies combined into a single Diakoptics structure 56
Figure 3-12. A control model for interconnecting two power systems working in different base frequency. 57
Figure 3-13. The network represented as a graph and its incidence matrix 59
Figure 3-14. The set of nodes strongly connected and their adjacent matrix. 59
Figure 3-15. Proposed iterative algorithm for detecting and reorganize islands within sparse matrixes 60
Figure 3-16. The proposed system to evaluate the performance of the proposed algorithm..... 61
Figure 3-17. IM obtained for the study case 1 61
Figure 3-18. Y_{BUS} matrix obtained for the case of study 1 61
Figure 3-19. IM obtained for the study case 2 62
Figure 3-20. Y_{BUS} matrix obtained for the case of study 2 62

Figure 3-21. IM obtained for the study case 3, at the left without permutation, at the right according to the order suggested by the ordering algorithm 62
Figure 3-22. Y_{BUS} matrix obtained for the case of study 3 63
Figure 3-23. IM obtained for the study case 4 63
Figure 3-24. Y_{BUS} matrix obtained for the case of study 4 63
Figure 3-25. The hierarchical computation model proposed for implementing Diakoptics 64
Figure 3-26. The interconnected network in Figure 3-2 (a) and its electrical equivalent using Diakoptics (b) 65
Figure 4-1. Project tree for creating an actor using NI LabVIEW 71
Figure 4-2. State Machine structure 72
Figure 4-3. Functional actor core created using NI LabVIEW (running under regular Windows OS) 73
Figure 4-4. Functional actor implemented for RT execution (running under Windows embedded OS) 75
Figure 4-5. New parameters required for actor's RT execution (running under Windows embedded OS) 75
Figure 4-6. Actor framework proposed for implementing the A-Diakoptics methodology 76
Figure 4-7. Hierarchical actor's model proposed for implementing A-Diakoptics 77
Figure 4-8. Expected interactions between actors 77
Figure 4-9. State machine proposed for the parent actor 80
Figure 4-10. State machine proposed for the child actor 81
Figure 5-1. Actor framework proposed for DSSim-PC 86
Figure 5-2. Load model in harmonics mode (Dugan et al., 2014) 88
Figure 5-3. Content of the child actors when working in harmonics mode 89
Figure 5-4.Harmonics meter graphical panel 90
Figure 5-5. Exchange methodology to include new models in the simulation 90
Figure 5-6. Medium-scale (a) and large-scale (b) DS modeled in DSSim-PC using the translator from .DSS to .DSP. Network (*a)* has one layer and network (*b)* has two layers. 92
Figure 5-7. Layers model for representing small-scale, medium-scale and large-scale power systems 93
Figure 5-8. Simulation time for executing 1 iteration on DSSim-PC and OpenDSS 94
Figure 5-9. Testing scenario using DSSim-PC 95
Figure 5-10. Testing scenario using OpenDSS 95
Figure 5-11. Results obtained in test scenario 2 95
Figure 5-12. Results when simulating 10000 iterations with OpenDSS and DSSim-PC (Graphical) 97
Figure 5-13. Computing time improvements (percentage) reached when using A-Diakoptics within PC architectures 98
Figure 5-14. Measurements taken at different points of EPRI's circuit 7 in dynamic simulation, graphic generated with DSSim-PC (Torsional mode oscillations)- 1.5 sec 98
Figure 5-15. Torsional mode oscillations in Voltages (p.u.) provided by DSSim-PC and OpenDSS at node x_1001805 - phase A, 2 sec 99
Figure 5-16. Multirate waveform generated by mixing the data and carrier signals 99
Figure 5-17. Data waveform and combined waveform (data + carrier) for reproducing the voltage signal according to the data generated. 100
Figure 5-18. Actor framework proposed for DSSim-RT 101
Figure 5-19. Heterogeneous computing environment proposed for implementing DSSim-RT 102
Figure 5-20. Performance comparison between PC and RT simulators using A-Diakoptics ... 102

Figure 5-21. Waveform generated with NI cRIO after the separation of a 1MW generator in dynamic mode, the waveform changes in magnitude and angle due to the torsional mode oscillations. Phase A 103
Figure 5-22. Simulation time (1 iteration) of the IEEE 8500 node test system simulated using several cores 104
Figure 5-23. Comparison between the measured times and estimated times using equation 5.2 104
Figure 5-24. Projection for estimating the behavior of the system processing time when the number of cores on the simulation increases 105
Figure 5-25. Latencies by using a communication network to communicate with IEDs 106
Figure 5-26. Classical methodology for gathering data from distributed meters (Agents) 107
Figure 5-27. State estimator architecture using A-Diakoptics 108

List of Tables

TABLE 1-1 19
TABLE 2-1 33
TABLE 2-2 33
TABLE 2-3 34
TABLE 2-4 35
TABLE 2-5 35
TABLE 2-6 36
TABLE 2-7 37
TABLE 2-8 38
TABLE 2-9 39
TABLE 3-1 52
TABLE 3-2 54
TABLE 4-1 69
TABLE 4-2 82
TABLE 5-1 91
TABLE 5-2 91
TABLE 5-3 93
TABLE 5-4 96
TABLE 5-5 97
TABLE 5-6 101

General Introduction

The simulation of power systems is an important tool for designing, developing and assessment of new grid architectures and controls within the smart grid concept for the last decades. This tool has evolved for answering the questions proposed by academic researchers and engineers in industry applications; providing different alternatives for covering several realistic scenarios.

Nowadays, due to the recent advances in computing hardware, Digital Real-Time Simulation (DRTS) is used to design power systems, to support decisions made in automated Energy Management Systems (EMS) and to reduce the Time to Market of products, among other applications.

Power system simulations can be classified in the following categories: (1) Analog simulation (2) off line simulation (3) Fully digital simulation (4) Fast simulation (5) Controller Hardware-In-the-Loop (CHIL) simulation and (6) Power Hardware-In-the-Loop (PHIL) simulation.

The latest 3 are focused on Real-Time Hardware-In-the-Loop (RT-HIL) simulation. These categories cover issues related to Electromagnetic Transients (EMT), phasor simulation or mixed (phasor and EMT). As mentioned above, these advances are possible due to the evolution of computing architectures (hardware and software); however, for the particular case of power flow analysis of Distribution Systems (DS) there are still challenges to be solved.

The current computing architectures are composed by several cores, leaving behind the paradigm of the sequential programing and leading the digital system developers to consider concepts such as parallelism, concurrency and asynchronous events. On the other hand, the methods for solving the dynamic power flow of distribution systems consider the system as a single block; thus they only use a single core for power flow analysis, regardless of the existence of multiple cores available for improving the simulation performance.

Divided into phase and sequence frame methods, these methods have in common features such as considering a single sparse matrix for describing the DS and that they can solve a single frequency simultaneously.

These features make of the mentioned methods non-suitable for multithread processing. As a consequence, current computer architectures are sub-used, affecting simulator's performance when handling large scale DS, changing DS topology and including advanced models, among others real life activities.

To address these challenges this thesis proposes an approach called A-Diakoptics, which combines the power of Diakoptics and the Actor model; the aim is to make any conventional power flow analysis method suitable for multithread processing. As a result, the nature and complexity of the power system can be modeled without affecting the computing time, even if several parts of the power system operate at different base frequency as in the case of DC microgrids. Therefore, the dynamic load flow analysis of DS can be performed for covering different simulation needs such as off-line simulation, fast simulation, CHIL and PHIL. This method is an advanced strategy for simulating large-scale distribution systems in unbalanced conditions; covering the basic needs for the implementation of smart grid applications.

Chapter 1. Introduction to Power Systems Simulation in Real-time

CHAPTER 1. INTRODUCTION TO POWER SYSTEMS SIMULATION IN REAL-TIME 13

CHAPTER 1 INTRODUCTION TO POWER SYSTEMS SIMULATION IN REAL-TIME 14

SIMULATION FIDELITY 15
Electromagnetic Transients (EMT) simulation 15
Load flow simulation (phasor simulation) 16
Hybrid simulation (phasor and EMT simulation) 17
SIMULATORS AND HARDWARE 17
Analog simulators 18
Digital simulators 19
Hybrid simulators 20
REAL-TIME SIMULATION AND ITS EVOLUTION 21
Addressing real life scenarios through Real-Time simulation 21
Power-Hardware-In-the-Loop simulators (PHIL) 23
Controller-Hardware-In-the-Loop simulators (CHIL) 23
CHALLENGES WHEN SIMULATING REAL-LIFE SCENARIOS 25
The evolution of computing systems 25
Multithread and vectorization 27
THE SCOPE OF THIS THESIS 28
CONCLUSIONS 29

Chapter 1 Introduction to Power Systems Simulation in Real-time

For several decades the simulation of power systems has been the main tool for designing, developing and assessing new technologies within the smart grid concept (Caixue, Kaldewey, Povzner, & Brandt, 2006; Ren et al., 2011). This tool is the source for answering the questions proposed by nowadays society by academic researchers and engineers in industry applications (Chang, Liu, Dinavahi, & Ke, 2008; Cheng & Shirmohammadi, 1995), which has generated a constant demand for making evolve these systems in order to cover a wide spectrum of realistic scenarios.

Nowadays, due to the recent advances in computing hardware, Digital Real-Time Simulation (DRTS) is used to design power systems, to support decisions made in automated Energy Management Systems (EMS) and to reduce the Time to Market release of products, among other applications (Dufour & Belanger, 2006; Dufour, Hoang, Soumagne, & El Hakimi, 1996; Fernandes et al., 2012). Moreover, each type of simulation has its own requirements in terms of performance, interface and hardware.

As an expected result of using simulations, the negative impact that new technologies could generate when they are integrated into a working system should be minimized. In fact, several studies reveal that depending on the development stage of a product, detecting an error could represent important economic issues (Soni, 2015). As shown in Figure 1-1, when developing a new technology the detection of bugs in the right moment could symbolize an important success; at the opposite side, if these bugs are not detected this new technology could represent a big economic disaster when trying to enter into the market competition (Harter, Krishnan, & Slaughter, 2000; Shull, Rus, & Basili, 2000).

For these reasons, Real-Time simulation has gained an important part of the simulation market in the recent years. Due to their flexibility and because they provide more information about the performance of a project or product in a test that can be destructive in real life, Real Time simulation is now the preferred tool for manufacturers and for the industries such as aerospace & defense, automotive, electrical and academic research among others (de Jong, de Gelder, Bussink, Verhoeven, & Mulder, 2013; Landry & Pritchett, 2002).

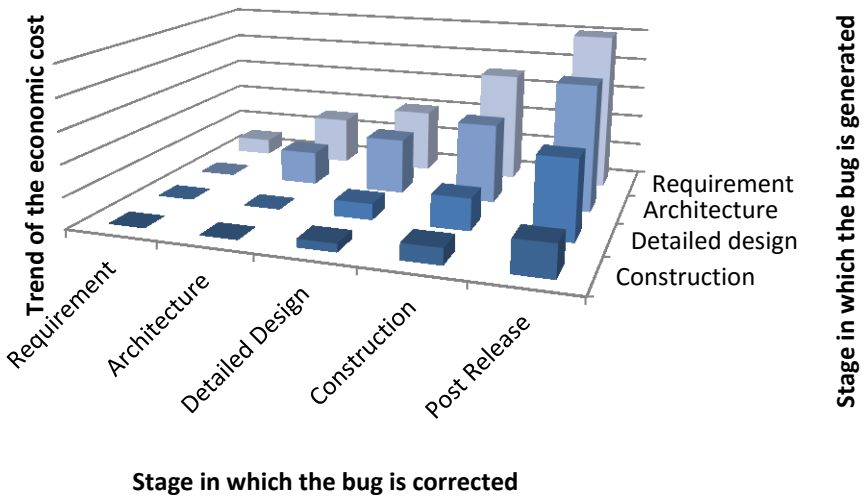


Figure 1-1. Cost trend when detecting bugs in different stages of a product development.

However, the scope of the simulation could vary depending on the type industry and the application field to simulate; these needs define the features of the simulation such as step time, computing power required, dynamic and generated signals. Based on these features, the simulation of power systems addresses several phenomena and their use could be oriented to different functional applications such as design and modeling, rapid prototyping, testing, teaching and training.

Simulation fidelity

In general, the aim of the Power system's simulation is the prediction of the technical limits that can produce stress to the components interconnected within the power system, such as power quality disturbances, frequency problems, voltage stability issues, among others. These phenomena can be predicted using different techniques depending on the scope of the simulation.

Electromagnetic Transients (EMT) simulation

Electromagnetic Transient simulation (EMT) refers to highly accurate simulations where the aim is to reproduce the behavior of the power signal when disturbed by high speed electromagnetic events. This type of simulation is time based and its timescale could vary from microseconds to nanoseconds, which makes of it highly demanding in terms of computational performance (Watson, Arrillaga, & Engineers, 2003).

EMT simulation involves primarily the interactions between the magnetic fields of inductances and the electric fields of capacitors connected to the power system, which can be generated by several causes as shown in Figure 1-2. As can be seen in this Figure, the time scale can drastically change depending on the type of phenomena to simulate; this feature will increase the complexity of the simulation in terms of computational burden and required hardware resources. For this reason, the EMT simulation in Real-Time normally involves small and medium-scale power systems interfaced using control devices such as VSCs in hybrid power systems, distributed generation (Missaoui, Warkozek, Bacha, & Ploix, 2012), HVDC systems (transmission systems), protections, among others (Rogersten, Vanfretti, Wei, Lidong, & Mitra, 2014; Yi, Gole, Wenchuan, Boming, & Hongbin, 2013; Yin, Haixiang, Ying, & Chen-Ching, 2014). The solution of EMT simulation is based on first order differential equations using Kirchhoff's laws for describing the behavior of RLC circuits when excited by a specific stimulus (Mekhtoub, Ibtiouen, Touhami, & Bacha, 2007).

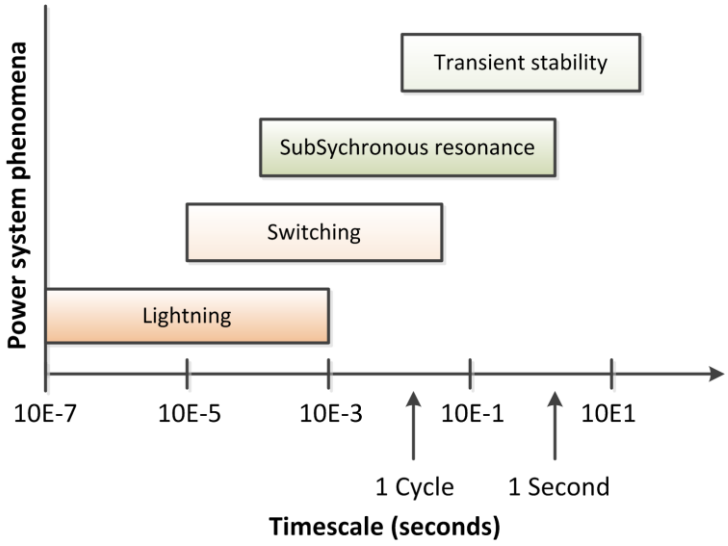


Figure 1-2. Simulated phenomena using EMT simulation (based on the graph provided in (Watson et al., 2003))

Load flow simulation (phasor simulation)

On the other hand, load flow simulation is focused on study the electromechanical transients. These transients are given due to the interactions between the energy stored in rotating machines, and the electrical energy stored in the grid. This type of simulation has a bigger time scale than in the case of EMT, but its field of application and he phenomena modeled demands this treatment.

This simulation is also called phasor simulation because is based on phasors to determinate the state of the power system instantaneously; however, this type of simulation can be translated to the time domain using sequential time simulation, which allows to incorporate the dynamic behavior of the power conversion devices for evolving into a type of simulation called transient stability programs or Transient Stability Analysis programs (TSA) (Sankarakrishnan & Billinton, 1995). This feature also has encouraged the development of simulation models for harmonic studies (Dugan, Arrit, Henry, McDermott, & Sunderm, 2014); complementing the load flow simulation for recreating short and long term variations of power quality (McDermott & Dugan, 2003; Montenegro, Hernandez, & Ramos, 2015; Montenegro & Ramos, 2012; Ramos & Montenegro, 2012).

The time scale of this type of simulation goes from milliseconds to weeks, months or years, but instead of EMT simulation, the scale of the power systems simulated can be large. The load flow simulation has been used mainly for analyzing the behavior of Distribution Systems due to its unbalanced nature; this feature increases the complexity for solving the system at computational level because of the size of the equations, which are modeled as a square matrix using nodes while in transmission systems, the model is balanced and analyzed using buses (Abdel-Akher, Nor, & Rashid, 2005; Stott, 1974).

The phenomena studied with this kind of simulation are shown in Figure 1-3. The control devices for this kind of simulation can be HVDCs, VSCs, generation controllers, protections, prime mover controllers, Load Frequency Controllers (LFC) and in general, the result of operations performed from the EMS.

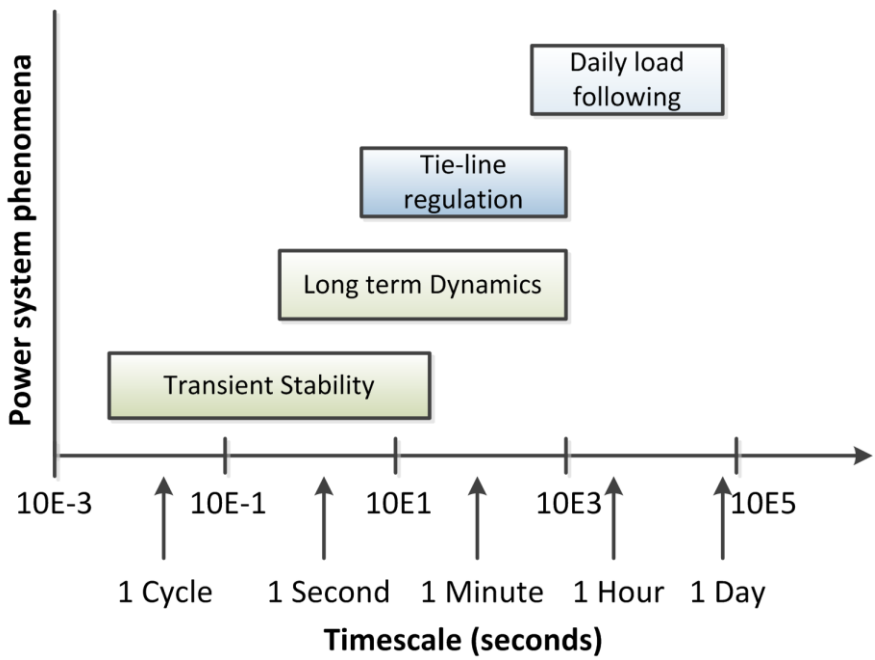


Figure 1-3. Simulated phenomena using phasor simulation (based on the graph provided in (Watson et al., 2003))

Hybrid simulation (phasor and EMT simulation)

Transient Stability Analysis do not require the same accuracy in the time domain than in the case of EMT. In TSA, the solution is based on a single frequency (one frequency per solution) using phasors; however, in the last years, several advances for integrating TSA and EMT simulations have been developed; the aim is to reduce the computational burden by focusing the EMT approach on a certain part of the system such as HVDCs, FACT, and leave the rest of the power system to the TSA approach.

In hybrid simulation is necessary to tolerate the reduced precision for transients that could happen in the EMT simulation, which is the result of the electrical interaction between the EMT and the TSA components and due to EMT is a multi – rate type of simulation. This feature is acceptable for pure AC systems, but when the system is a hybrid AC/DC system large errors can compromise the fidelity of the simulation (Xi, Gole, & Ming, 2009; Yizhong, Wenchuan, Boming, & Qi, 2013).

For solving this challenge, several authors have proposed the integration of a Frequency Dependent Network Equivalent (FDNE) for interfacing the TSA and EMT components; this approach looks for preserving the accuracy of the EMT part by adding equivalent representations (Xi et al., 2009). An example of a system suitable for hybrid simulation is shown in Figure 1-4.

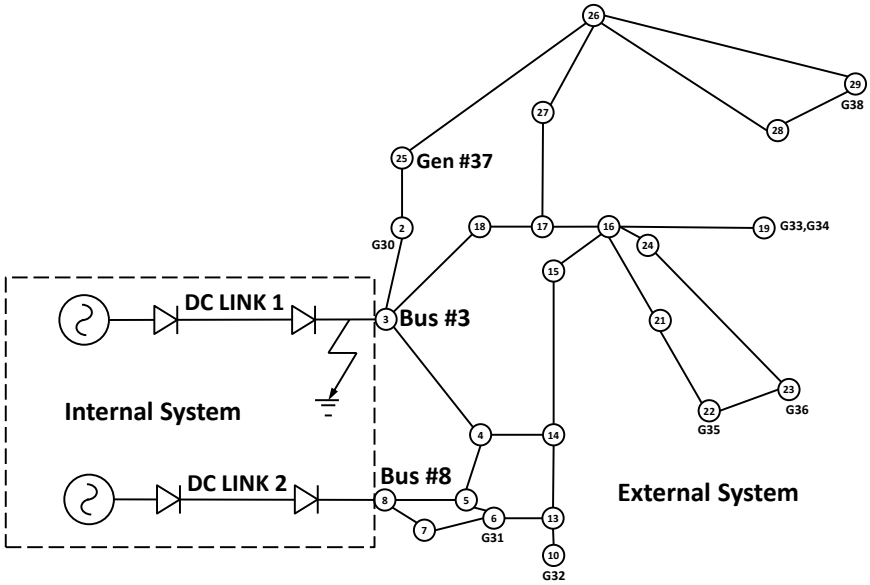


Figure 1-4. 39-bus New England AC system with two HVDC links (Xi et al., 2009)

The simulation fidelity is also affected by the hardware used for recreating the simulated scenario, which has evolved for covering the needs of the industry and academia looking for modularity, flexibility and scalability.

Simulators and hardware

According to the hardware used for recreating realistic scenarios, simulators can be separated into analog and digital simulators. The first case consists of building reduced models that approximates the real system behavior; on the other hand, digital simulators are computing programs used for solving differential equations and recreate the behavior of the power system (Roitman, Watanabe, & Lyra, 1989). Both are described as follows.

Analog simulators

The concept of analog simulators is composed by two different approaches: The construction of reduced models using electrical elements for reproducing the behavior of a real system, and the inclusion of active electronic circuits to overcome speed limits in digital simulators (analog computations) (Nagel, Fabre, Cherkaoui, & Kayal, 2010). Classically, this kind of simulator is the base for practicing in laboratories and teaching due to its similarity to the studied phenomena (Imai, Iizuka, Makino, & Horikoshi, 2002). However, when the study case becomes medium/large-scale with hundreds of elements and complex cases, the costs associated with this kind of simulators can increase drastically and the complexity of the implementation as well.

The advantage when working with analog simulators is that there are no delays when performing the tests. On the other hand, the flexibility of the recreated scenarios using this technique is limited because of the components used are too specific, leading to prefer other simulation techniques (Gombert, 2005):

- The size of the systems that can be represented is limited due to the complexity of the scenarios to be recreated.
- The flexibility, modularity and scalability of these kind of simulators is also limited due to the amount and type of components required for the simulation. In fact, sometimes the stress experienced by these components require to have a replacement for each component depending on the number of iterations required for gathering data (Braham, Schneider, & Metz, 1997).
- The values and type of the components required for recreating a certain scenario are specific, which makes of the simulator uniquely and not suitable for creating other experiments.

An example of a test system implemented using an analog simulator is shown in Figure 1-5(Imai et al., 2002).

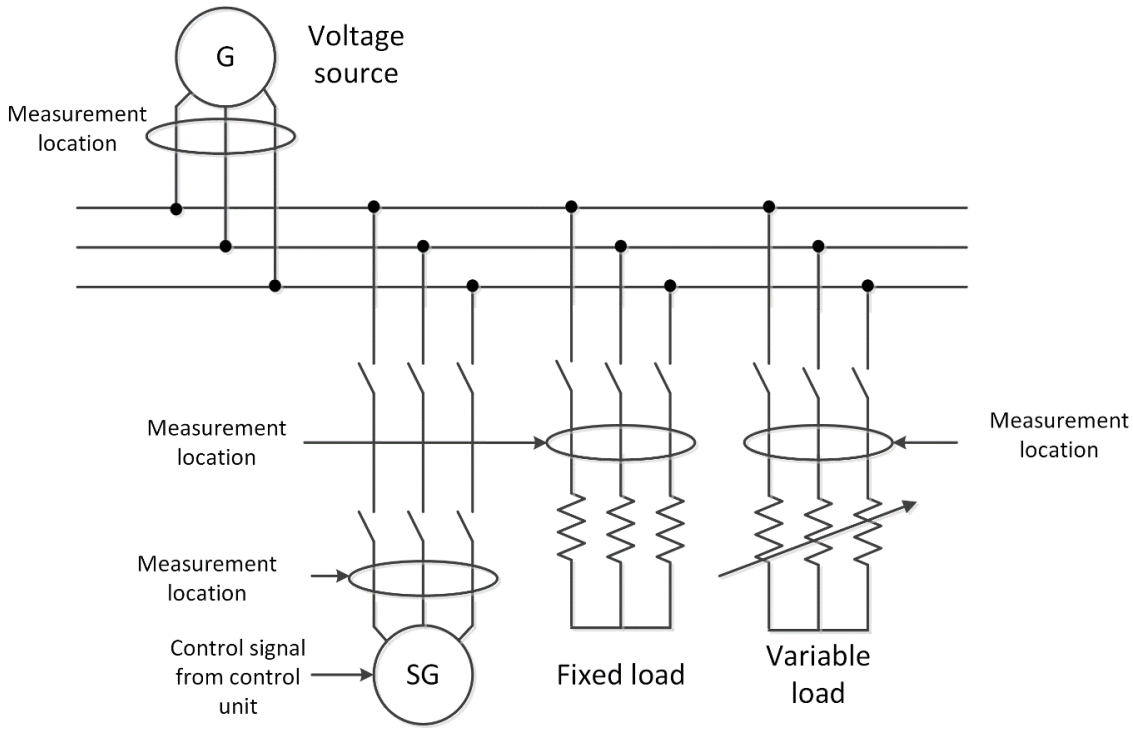


Figure 1-5. Test system implemented using an analog simulator (Imai et al., 2002).

Digital simulators

Digital simulators are computer based and consist in algorithms for solving linear and nonlinear equations in order to reproduce the behavior of the physical system. These simulators can work off-line, on-line and Real-time for recreating different scenarios (Ren et al., 2011; Wang, Guo, Xiao, & Zhao, 2010). The precision and fidelity of these simulators depends on the computing hardware architectures and interfaces. Some of these can be used for collecting data without any interaction with the real world, while some others can be used for interacting with devices under test (DUT) or hardware under test (HUT); closing the simulation loop using Digital to Analog Converters (DAC) or communication interfaces (Dufour & Belanger, 2006; Dufour, Belanger, & Lapointe, 2008).

Driven using software interfaces, the main feature of this kind of simulator is that the simulation becomes flexible, modular and scalable; nevertheless, depending on the hardware characteristics the spectrum of the simulated signals can vary (Ocnasu, 2008; Ren et al., 2011).

Nowadays, this type of simulation is the preferred by the power industry and academia; addressing a wide spectrum of applications and issues related to the development of new control and monitoring devices for building the modern power grids (Zhong et al., 2013). In TABLE **1-1**, a list of the most representative manufacturers of digital simulators for TSA/EMT is presented.

TABLE 1-1
MOST RECOGNIZED SIMULATION TOOLS AVAILABLE NOWADAYS

Company	Product name	Type of simulation	License
DigSilent	Power Factory	Off Line/On-line	Paid
Manitoba Hydro International Ltd.	PSCAD/EMTDC	Off Line	Paid
International Power Electric Technology Co.	CYMDIST	Off Line	Paid
WH Power consultants	RDAP	Off Line	Paid
Matworks	MATLAB SimPower Systems	Off Line	Paid
Battelle Memorial Institute	GridLab-D	Off Line	Freeware
EPRI	OpenDSS	Off Line	Open Source
Stanford University	GridSpice/ GridLab-D	Off Line	Open Source
Operation Technology, Inc.	etap	Off Line/Real-Time	Paid
Opal-RT	ePHASORSim	Real-time	Paid
RTDS	RSCAD-GTNET-PMU	Real-time	Paid
dSPACE	SCALEXIO/RTI	Real-time	Paid

Hybrid simulators

For addressing the limits identified in analog and digital simulators, the hybrid simulator combines their strengths for evolving into an advanced simulation platform. This simulator combine the fidelity of the analog simulators with the models recreated in the discrete time domain using digital systems (Gick, Gallenkamp, & Wess, 1996). Hybrid simulators are used for exchanging signals between the discrete and continuous worlds, and explore the different states that a new device require for addressing the different functional states of the power system (Zhu, Dong, Hu, & Xie, 2013).

For interfacing the digital and analog simulators it is required the inclusion of power amplifiers and DACs; this way, highly complex equipment can be simulated using analog simulators while the rest of the power system is recreated digitally. The literature reports applications such as testing of HVDC interfaces, the interconnection of distributed generators to analog HVDC systems, the testing of dynamic loads simulated digitally and connected to analog power systems, among others. A general example of a hybrid simulator is shown in Figure 1-6.

Depending on the application, the voltages and currents gong from the Digital simulator to the Analog Simulator and vice versa can vary. For example, consider an experiment where a set of controllers are simulated using the digital simulator and the power system with the Analog one. In this case, the voltage and current signals goes from the analog simulator to the digital and if necessary, the Digital part will feedback some processed signals.

Another example could be when a large-scale DS is simulated in the Digital simulator and a small part of the big system is simulated using the Analog simulator, which is used for recreating power quality issues and interconnect real controllers; then, the results of the control actions and the disturbances generated in the Analog simulator are entered into the Digital simulator for feeding back the entire power system in the next iteration. These kind of simulators are Real-time driven, which is the subject of the next section.

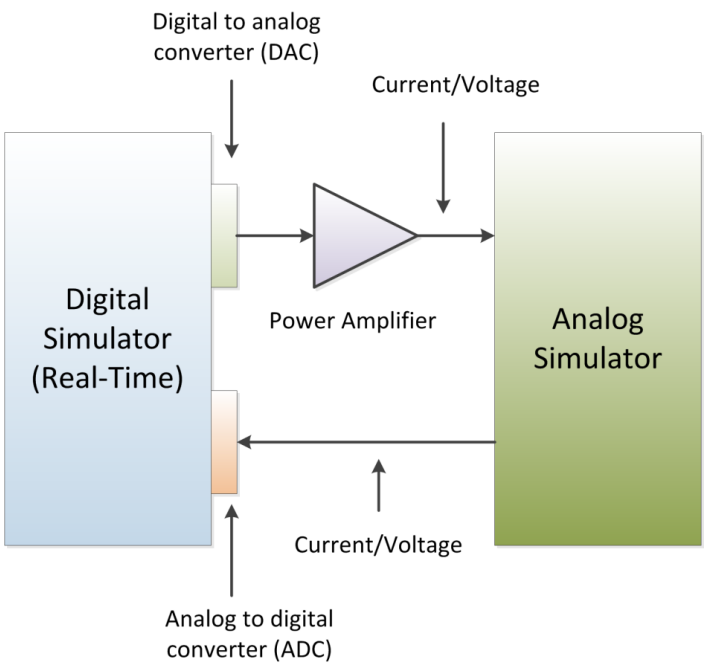


Figure 1-6. Hybrid simulator example

Real-time simulation and its evolution

Real-time simulation is the preferred tool for designing, testing and assessing new technologies and products in the last decades. This kind of simulation allows to recreate the real world using computing hardware, which can be interfaced with real equipment/system for evaluating its performance before being installed in real the real world (Laplante & Ovaska, 2011).

A Real-time (RT) computer system is a computer system where the functional state of this system is not only linked to the compute cycles, but also on the physical time required for performing these computations. The functional states refer to the input/output cycles of the system. So, RT systems are time dependent digital systems that evolve in time according to a fixed period, ruled by a global clock (Kopetz, 2011).

These computers are always part of a larger system called RT systems or cyber-physical systems due to its dependence of physical-time. The general structure of an RT computer is distributed in computer nodes interconnected by a RT communication network (Kopetz, 2011). The time definitions that rule the RT system are described as follows:

- A cut in the timeline is called an instant.
- Any ideal occurrence that happens at an instant is called an event.
- The information that describes the event is called the event information.
- The present (now) is the reference point used for separating the past from the future.
- An interval on the timeline is called duration and is defined by two events (start and terminating)
- Because the system is referenced to a digital clock, the timeline is partitioned into a sequence symmetrically spaced, each space is called granules of the clock, which are delimited by special periodic events called ticks of the clock.

These definitions will be used for describing the operation of the RT simulators and their features for addressing different applications.

Addressing real life scenarios through Real-Time simulation

The advantage of RT simulators is the fact that computations are governed by deterministic computing times, which allows to ensure the response of the system to events produced in the continuous time domain with an accurate approach from the discrete time domain (Proakis & Manolakis, 1995; Tan, 2007). This way, real world scenarios can be recreated for testing, design and assessment of new technologies in the laboratory by offering high scalability, modularity and flexibility.

This tool has been used for recreating a large number of real world situations, the following are documented cases when RT simulators have been used in the power systems industry:

- Design and testing of electric vehicles and storage devices (Dufour & Belanger, 2006; Yuhe, Wenjia, Kobayashi, & Shirai, 2012).
- Design and validation of adaptive protection systems (Abdulhadi, Coffele, Dysko, Booth, & Burt, 2011; Crăciun et al., 2014).
- Harmonics studies (Chang et al., 2008).
- Switching transient studies (Bacha Seddik, 2014; Deokar, Waghmare, & Takale, 2009).
- Design and testing of transmission lines (Dufour et al., 1996).

- Dynamic simulations for transmission and distribution power systems (Jalili-Marandi, Ayres, Ghahremani, Belanger, & Lapointe, 2013).
- The testing of Power Quality monitoring systems (Montenegro et al., 2015; Radil, Ramos, Janeiro, & Serra, 2008).
- Development of interfaces for hybrid power systems (AC-DC links) (Zhu et al., 2013).
- Validation of building management strategies (Missaoui et al., 2012).
- Studies for evaluating the dynamic behavior of interconnected distributed generators (Palensky & Dietrich, 2011; Yizhong et al., 2013).
- Validation of information models for handling information through a network with distributed control devices (Zavoda, 2008; Zhong et al., 2013).
- Studies about peak load shaving using Distributed Energy Resources (Van-Linh, Tuan, Bacha, & Be, 2014).
- To evaluate non-intrusive load monitoring strategies (K. Basu, Debusschere, Bacha, Maulik, & Bondyopadhyay, 2015; Kaustav Basu, Debusschere, Douzal-Chouakria, & Bacha, 2015).
- To evaluate the performance of Real-Time estimation algorithms on DFIG and other rotational machines and stability analysis (A. Djoudi, H. Chekireb, E. Berkouk, & S. Bacha, 2014; A. Djoudi, H. Chekireb, E. M. Berkouk, & S. Bacha, 2014).
- Power optimization studies on Distributed Energy Resources (Taraft, Rekioua, Aouzellag, & Bacha, 2015).
- Demand side optimization studies on Distributed Energy Resources (Thiaux, Dang, Multon, Ben Ahmed, & Tran, 2015).
- And many others…

As can be seen in the previous list, the nature of the scenarios developed using RT simulation can vary significantly even if all of them are part of the same field of application, which has led to classify the RT simulators in two categories for interfacing them with real world devices in a closed simulation loop called Hardware-In-the-Loop (HIL). This concept is illustrated in Figure 1-7.

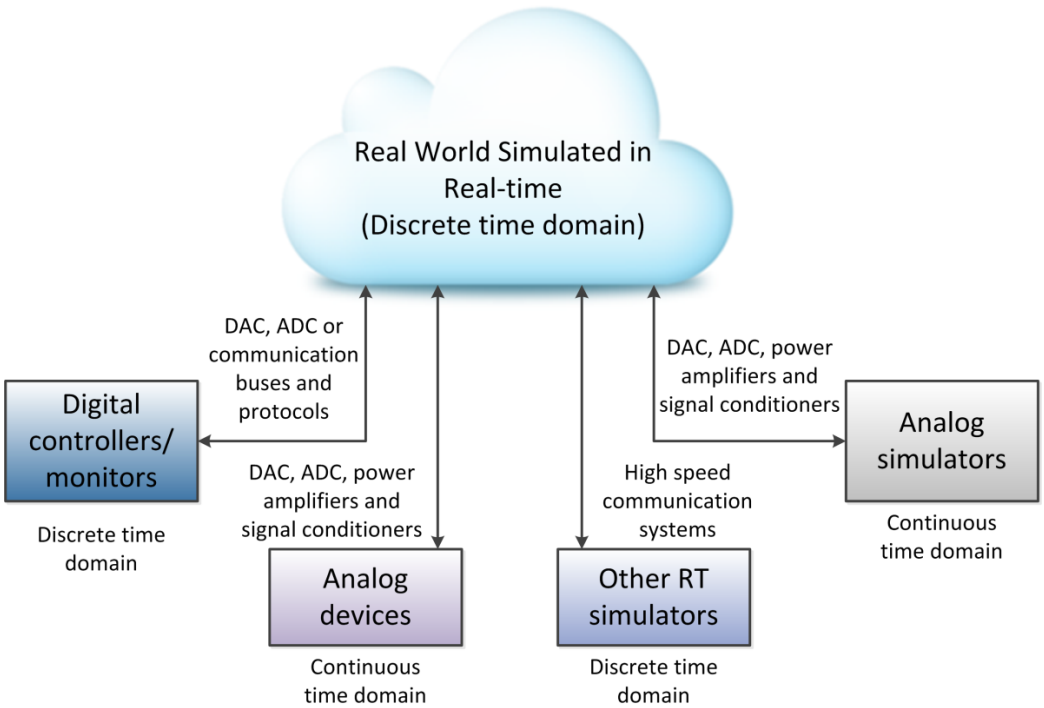


Figure 1-7. Uses of Real-time simulators for addressing real world applications.

Power-Hardware-In-the-Loop simulators (PHIL)

Power-Hardware-In-the-Loop (PHIL) is a Hardware/Software HIL technique where the exchange of information between the RT simulator and the real world is made through analog power signals. The main parts of this hybrid simulator are the power amplifier and the signal conditioning interfaces (Lundstrom, Shirazi, Coddington, & Kroposki, 2013). Because the aim of this technique is to simulate power conditions for driving the external systems to a functional state, the interfaces with the real world generates separate voltage and current signals; thus making easier to test and to evaluate the system response when a certain event occurs (Dargahi, Ghosh, Ledwich, & Zare, 2012; Lundstrom, Mather, Shirazi, & Coddington, 2013). Here, the element under study is the entire power system and its behavior in time.

However, for using this technique and selecting the elements involved in the process, there are several considerations that must be deliberated before implementing an experiment (Ocnasu, 2008):

- The nominal values of the waveforms, which must be considered in selecting properly the magnitudes of the interface and the operation ranges. If these values are not considered undesired effects could be part of the test such as saturation, protections tripping, malfunctioning of loads and connected equipment and the destruction of data acquisition devices.
- The compatibility between the voltage levels between the analog and digital parts must be guaranteed. The gaining of the data acquisition devices and the gain levels of the power amplifiers must be coherent for avoiding accidents.
- The operation range of all the elements must be well defined. With this consideration it is expected to avoid damage on the elements used.
- The bandwidth of each element must be in the range established for the experiment.
- It is important to know the response time of the elements included in the simulation; this way, the step time of the simulation will be selected for avoiding unpredictable behaviors and undesired noise or signal shifting.
- Due to the nature of the simulations where the power flow can change product of a switch tripping to include reversible power conversion devices, distributed generators, among others. For these reasons it is necessary that the power amplifier could work in the four quadrants with a sufficient bandwidth.

Controller-Hardware-In-the-Loop simulators (CHIL)

On the other hand, Controller-Hardware-In-the-Loop (CHIL) is a technique where the interaction with external equipment is made through communication interfaces (Steurer, Bogdan, Ren, Sloderbeck, & Woodruff, 2007). This is because nowadays power networks transport not only power, but also data, which is transmitted using field buses and protocols for driving monitoring and control operations such as feeder reconfiguration, fault detection and isolation, Volt/VAr control, among others (Zavoda, 2008, 2010).

These kind of simulation is widely used for Phasor Measurement Unit (PMU) placement, for assessing power system management activities related to Electric Vehicles (EV), Electronic power converters (Lucia et al., 2011), Storage Devices and distributed generation penetration (Yuhang, Hui, & Foo, 2009). Advanced Distribution Automation (ADA) is as well, another topic studied when using these kind of simulators (Zavoda, 2010).

At a difference of PHIL, the controllers in CHIL are external equipment, replacing the simulated controllers present in PHIL (Yousefpoor, Parkhideh, Azidehak, & Bhattacharya, 2014), which means that the element under study is the controller. The communication protocols most used in these applications are:

- Protocols
  - CAN-Based on CAN bus
  - ModBus (RTU,ASCII, TCP-IP)
  - IEC61850
  - CS104
  - XMPP
- Field Buses
  - Optical Fiber
  - Ethernet
  - GPRS/GSM
  - ZigBee oriented solutions (for closer locations)
  - RS485 (near to the end points)
  - CAN/ASi

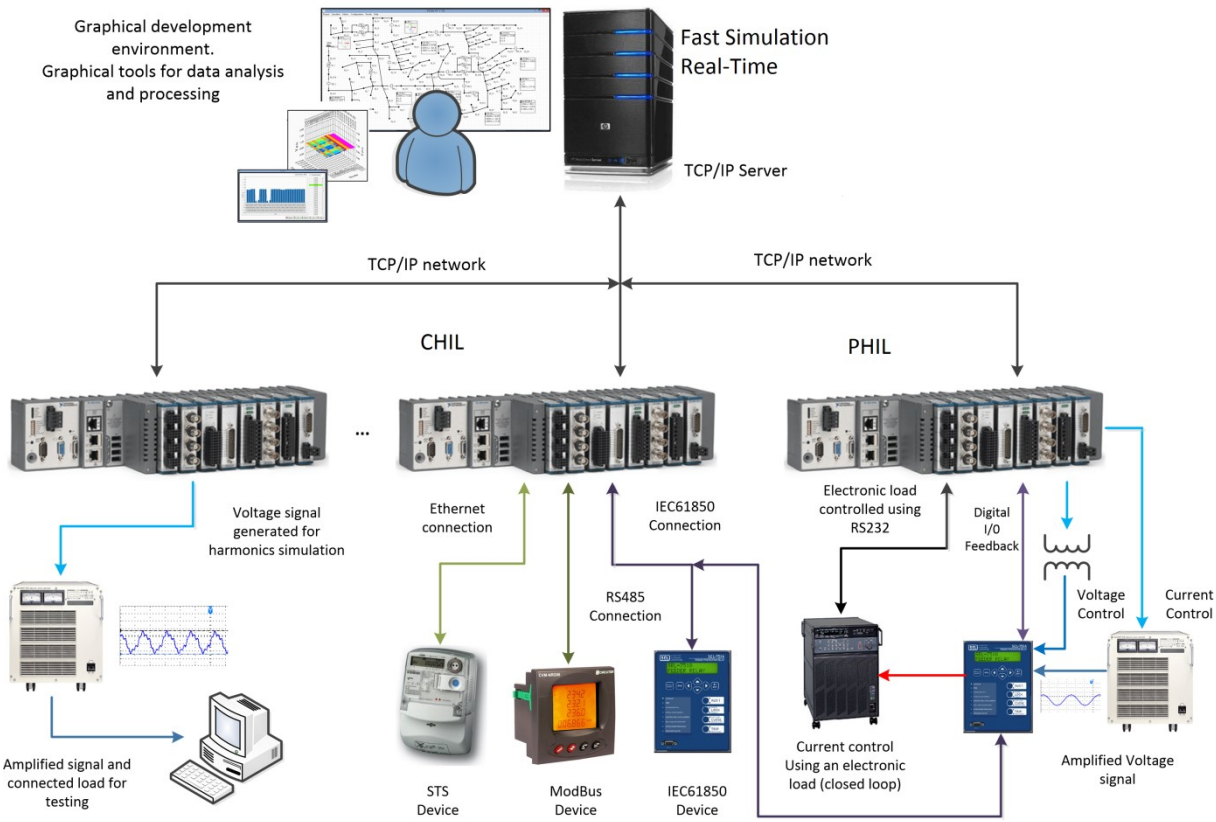


Figure 1-8. HIL Power system simulation categories.

Figure 1-8 presents an application example of the different HIL techniques used for simulating power systems. These developments are possible due to the evolution of the computing architectures, which have evolved through time passing from sequential to parallel systems; involving concepts such as concurrency and asynchronism to the development of RT simulators. These topics are presented in the next section.

Challenges when simulating real-life scenarios

As mentioned above, digital RT simulators are highly dependent of the computing available technology. The features of each simulator are directly associated to the hardware and software architectures used for its development.

For covering the needs in terms of performance and determinism of the simulation platforms described at this point, computer architectures have evolved from large to smaller computing hardware, which is more efficient and faster; however, these computers also require advanced programming structures at software level for exploiting all its power, requesting from RT systems developers to leave behind the paradigm of sequential programing (Abbas & Ahmad, 2002).

The evolution of computing systems

At the beginning of computing science, the computing systems have been designed for working sequentially. This trend prevailed until the beginning of the 21th century, which has led the software development tools to follow a vertical structure for using central-processing-units (CPU) working at fast rates. The features of this type of architecture are (Barney, 2014):

- A problem is decomposed into a sequential set of instructions.
- The instructions are executed sequentially (only one instruction was executed at the same time)
- There is only one processor for handling all the instructions

This computing methodology was used for decades, but conforming the industry and academic researchers needs higher fidelity and lower computing times, hardware manufacturers were forced to develop faster processors for keeping the sequential processing methodology (Diaz, Munoz-Caro, & Nino, 2012b).

However, this hardware acceleration has reached its limits when hardware manufacturers realize that the heat and stress on the components started to become critical. So, new developments for reaching lower computation times and recreate accurately the real world were redirected to distribute tasks, is then when the parallel processing is needed. These concepts are illustrated in Figure 1-9.

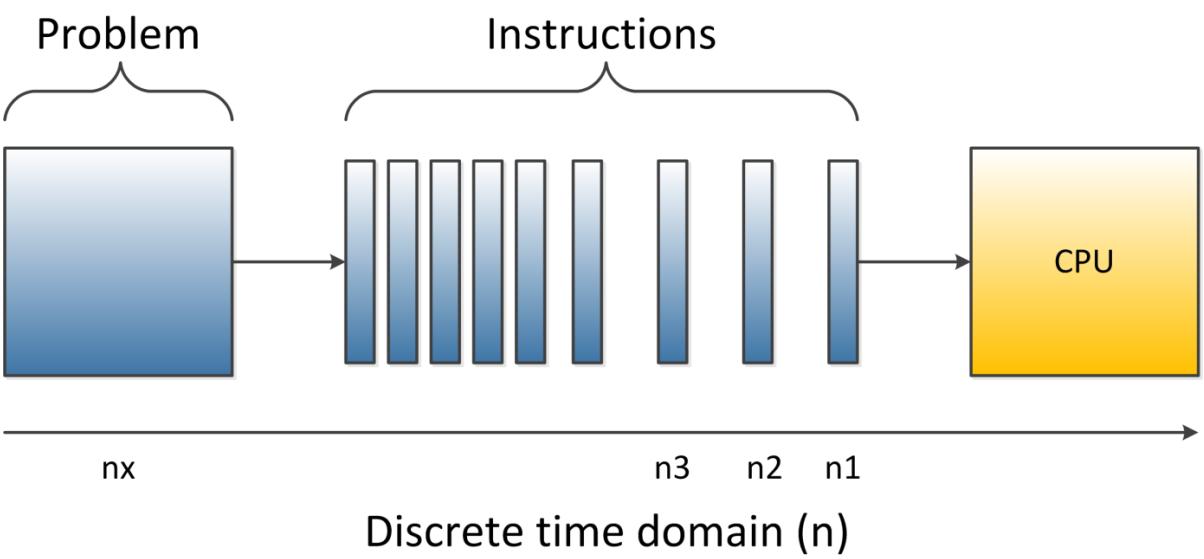


Figure 1-9. Classical sequential computing systems

The first parallel systems were built using computer networks interconnected using communication systems. These systems are known as heterogeneous computing systems due to the diversity of technical specification of the distributed computers (Hewitt, 1976; Hewitt & Barker, 1977). Even if all the computers have the same technical specifications and were manufactured by the same company in the same production batch, there are small differences that make each one unique, requiring to establish controls for guaranteeing synchronism, stability and inconsistency robustness in the same computation granule (Hewitt, 2012).

Then, thanks to the high integration and miniaturization techniques, groups of processors are integrated within the same structure, entering in the multi and many-core computing era (Diaz, Munoz-Caro, & Nino, 2012a). This processors are known as homogeneous computing systems and are very common nowadays, they can be found in every computer no matter the application field. The technical features of this technology are:

- A problem is separated into discrete parts for being handled concurrently.
- Each part is later broken into a sequential set of instructions.
- The instructions are executed sequentially on each core (more than one instruction was executed at the same time)
- Global control/coordination mechanism must be employed for handling the operations performed by each core and coordinate the exchange of data between them.

For making this processing possible the problem to solve should be able to be broken into discrete pieces, considering multiple program instructions at any moment in time and as a consequence, the problem will be solved in less time than with a classical sequential computing system. Nevertheless, many of the computing software keeps being developed using sequential programming structures, which sub-utilize the power of parallel computing due to the program will use a single core for its execution.

Parallel processing is the alternative for recreating the real world, which is massively parallel. In the natural world many events are happening at the same time preserving a temporal sequence, and compared with sequential programming, parallel programming is more adequate for modeling, simulating and explaining real world phenomena (Barney, 2014; Schlesinger, 2010). These concepts are illustrated in Figure 1-10.

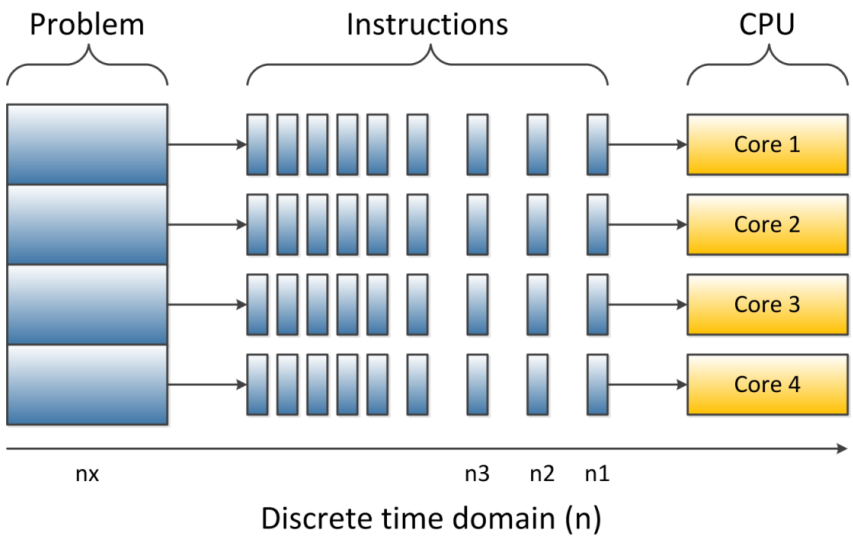


Figure 1-10. Multicore computing systems

For handling parallel computing structures, the processor uses threads, which are also known as virtual processors within the physical processor. A thread can be described as a recipe delivered by parts to different chefs in a kitchen; there is no order for developing each part of the recipe and each chef works independently. Then when each part is ready each chef provides his part for building the total recipe (Inoue & Nakatani, 2010). However, for avoiding a chaotic kitchen, a leader chef is needed for coordinating the kitchen operation. In fact, multicore processing is called multithread, because is the thread the base entity for distributing processes within a core (Jainschigg, 2012).

Additionally, by the introduction of Fast Programmable Gate Arrays (FPGAs), the hardware can be described using hardware interfaces such as Very High speed Hardware Description Language (VHDL), opening the possibility for RT developers to customize their own hardware for developing specialized functions in high deterministic computing times (Brant & Lemieux, 2012; Dufour et al., 2008).

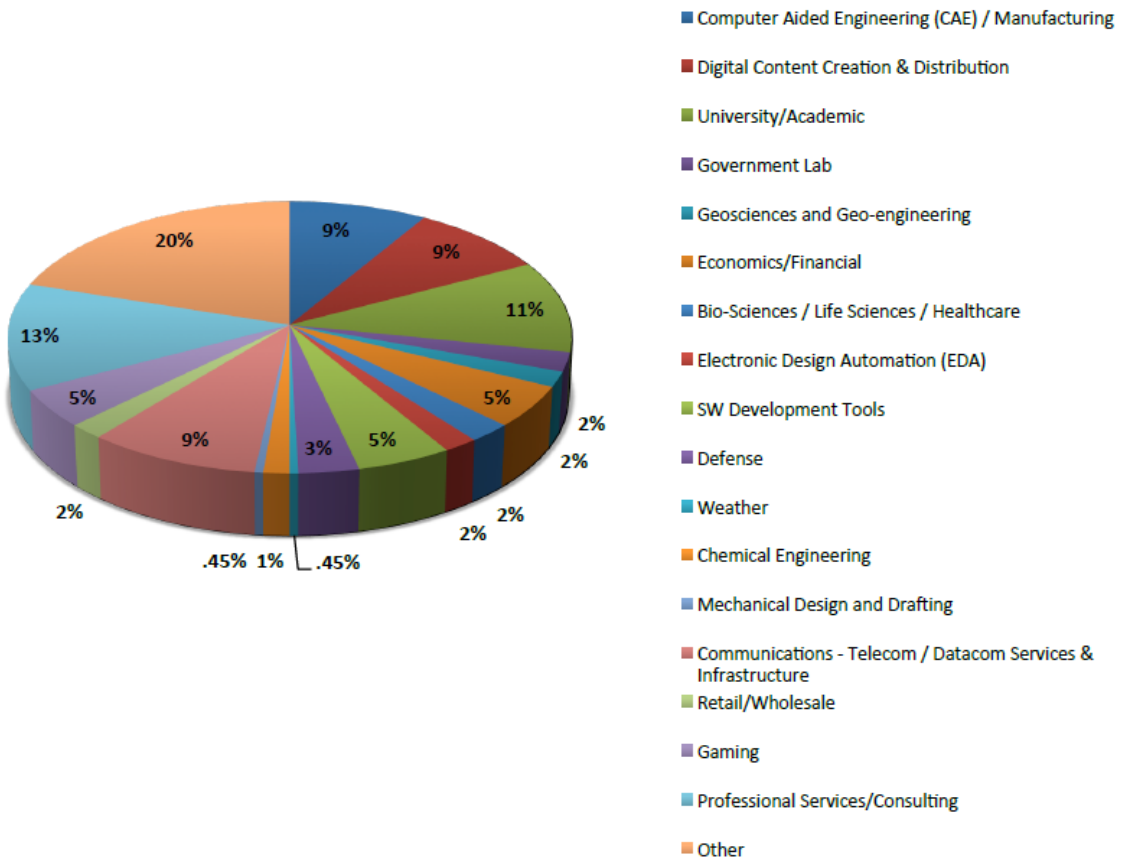


Figure 1-11. Application fields of parallel computing (Jainschigg, 2012)

Parallel programming/computing is widely used in several industries and application fields. Figure 1-11 provides an example of the application sectors where parallel computing is used nowadays (Jainschigg, 2012).

Multithread and vectorization

For handling parallel computing resources there are two methodologies proposed in the literature:

- Multithreaded programming
- Vectorization

Multithreaded programming has existed for decades, in fact, the single-core processors on the early days used threads successfully for handling multiple tasks. For handling threads, the processor halts one thread for allowing the execution of another, which means that a single processor can only run one thread at the same time.

However, nowadays desktop computers can include up to 4 cores on a single chip, server computers up to 24 and so on. Each core can execute 2 threads increasing the number of threads based on the number of cores. The advantage of managing threads this way is that if any thread enters into an infinite loop the rest of the process keeps running with the other threads. So, the success when using multithread programming consists into distribute the threads within the different cores (Cogswell, 2015).

On the other hand, vectorization consist into utilizing properly the registers within the processor, which can be accessed using the assembly language instructions. These registers are big memory spaces incorporated in the processor, for example, for a computer of 256 bits, these registers are 512 bits size in the new Intel processors.

The aim of vectorization is to accommodate the grouped data, such as in the case of arrays or clusters, into a single register of the processor to compute several data using one computation. For example, consider a numeric array of 40000 elements where each element is single precision float (32 bits). If at certain point of the program it is necessary to operate the entire array, this will require 40000 iterations. On the other hand, if the main processor counts with an internal register of 512 bits means that 16 elements of the array of numbers can be stored in this register and be operated at once. As a result, only 2500 iterations will be required for operating the entire array (Cogswell, 2015).

The combination of these two techniques is also known as vectorized multithreaded processing and represents the most used alternative for building RT systems (Jang-Ping & Chih-Yung, 1994; Zhang & Zhao, 2010).

The scope of this thesis

This thesis is focused on making TSA methods suitable for parallel processing; as a result, traditional methods for TSA of Distribution power Systems (DS) will be suitable for multithread processing and exploit the power of many-core computers in desktop computers and RT systems.

As it will be presented in Chapter 2, TSA methods for DS considers the system as a single block, preserving the sequential behavior of the computational model for being processed using a single processor within a homogeneous multicore computing environment.

The TSA in DS is an important subject of study because:

- For developing Smart Grid devices, equipment and management technologies the complexity of the models and their behavior in time gets increased.
- Large-scale systems need to be modeled for nowadays Smart Grid developments without compromising the computation speed and accuracy of the simulation.
- The existing methods for solving DS power flow are not suitable for multithread processing (Explained in Chapter 2).
- The inclusion of the dynamic equations for the modeled loads can represent an important computational burden for middle and large-scale DS, affecting the simulation performance.

- When a topology change is performed in the middle of a simulation, the number of calculations involved in reproducing this change on the DS can be important.
- RT simulation has moved to DS applications due to the development of smart grid and micro grid studies.
- In most cases of distribution network, fast transient behaviors are not the concerns of the studies, and thus TSA may be appropriate to allow the simulation of large scale distribution network. This type of simulation using RT have many applications for distribution systems.

These issues can be addressed by using separation techniques and information models for handling the parallelism and concurrency proposed in this chapter. Literature reveals that these techniques can be adequate for helping in the evolution of the existing simulation methods, thus making them suitable for multithread computing as described in Chapter 3.

By combining the power of Diakoptics (Kron, 1955, 1963) as separation technique and the actor model (Hewitt, 2012) as information model, this thesis will make traditional methods for solving DS power flow, suitable for multithread processing; opening the door for exploiting the power of current computation technologies and improve the performance of existing computational tools.

Conclusions

Nowadays industry and academia utilize RT simulation as the main key for designing, modeling, developing and assessing new technologies to face the challenges proposed by the power industry. Computer technologies have evolved and the methods of analysis as well. However, there are several issues that need to be attended in order to make traditional methodologies compatible with current computing technologies, opening the door for a better use of the existing computing hardware resources at standard PC and RT level.

Chapter2. Simulation of Distribution Power Systems for Shaping the Future Smart Grid

CHAPTER2. SIMULATION OF DISTRIBUTION POWER SYSTEMS FOR SHAPING THE FUTURE SMART GRID 30

CHAPTER 2 31

SIMULATION FOR THE DEVELOPMENT OF THE FUTURE SMART GRID 31
THE SOLUTION METHODS FOR LOAD FLOW ANALYSIS IN DS 32
Power flow analysis based on phase frame 32
Power flow analysis based on sequence frame 36
Dynamic Simulation..... 38
CHALLENGES FOR DS SIMULATION 41
The deterministic computing cycles 41
Complexity of the simulation..... 41
Topology changes..... 42
Hybrid power systems 42
CONCLUSION 42

Chapter 2

Simulation for the development of the future Smart Grid

As presented in chapter 1, the current computing architectures are composed by several cores, leaving behind the paradigm of the sequential programing and leading the digital system developers to consider concepts such as parallelism, concurrency and asynchronous events (Yildirim, Arslan, Kim, & Kosar, 2015; Yildirim, JangYoung, & Kosar, 2012). However, for the particular case of power flow analysis of Power Distribution Systems (DS), there are still challenges to be solved. The methods for solving the dynamic power flow of DS consider the system as a single block; thus they only use a single core for power flow analysis, regardless of the existence of multiple cores available for improving the simulation performance (Balamurugan & Srinivasan, 2011).

These methods were inspired by the developments made for the power flow analysis of the Transmission Systems (TS), which are meshed networks assumed as balanced and with low R/X ratios; making of these suitable for being analyzed using methods based on Gauss-Seidel, Newton-Raphson and its different versions (Dandachi & Cory, 1991; Stott, 1974). On the other hand, the modern DS has special features that differ from transmission systems:

- These are radial or weakly meshed networks with non-transposed conducting lines.
- The load is distributed unbalanced, this requires considering all the phases to perform the load flow analysis.
- The short length of feeder lines and the presence of very high frequency power electronic converters, make the real-time simulation of a comparatively large modern distribution networks much more challenging
- The high penetration of non-conventional loads within the DS is a normal situation nowadays. Distributed Energy Resources (DER) such as wind turbines (WT), photovoltaic cell arrays (PV) and fuel cells change the conventional radial configuration of the grid (Arritt & Dugan, 2011; Dugan, Arritt, McDermott, Brahma, & Schneider, 2010; Ning et al., 2013; Ustun, Ozansoy, & Zayegh, 2011); making the DS not adequate for being analyzed by using the methods mentioned above (Balamurugan & Srinivasan, 2011).

These features represent additional challenges for analyzing DS, which are increments in computational burden for digital simulation:

- Bigger systems mean more memory.
- Topology changes can drive to an important number of iterations for reshaping the system's model.
- Large-scale systems handled by a single core will result in non-optimal implementations when the simulations are performed in multicore computer architectures.

For analyzing the DS considering these features, two reference frames are reported in the literature: The phase frame and the sequence frame. The phase frame refers to the techniques that consider all the calculations based on using the 3 phase a-b-c (T. H. Chen, Mo-Shing, Hwang, Kotas, & Chebli, 1991; Mo-Shing & Tsai-Hsiang, 1991). Instead, the sequence frame separates the DS in three phasor networks, which are solved separately and then are superposed to find the total solution (M. Z. Kamh & Iravani, 2010).

The solution methods for load flow analysis in DS

For solving the DS load flow (in general) the solver considers the system's Y matrix and a vector of injected currents. This vector contains the currents injected by the substation and other power conversion devices.

The general structure is as follows:

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} Y_{11} & 0 & \cdots & 0 \\ 0 & Y_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Y_{nn} \end{bmatrix}^{-1} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \tag{1}$$

The structure presented in (1) is the linear approach for solving the power flow of the DS; however, this partial solution must be complemented with the non-linear components, which are calculated by correcting the currents injected by the power conversion devices (Brown, Carter, Happ, & Person, 1963). This is an iterative procedure and the methods for calculating these new currents could vary depending on the selected frame.

Some of these methods can be used for simulating the behavior of DER and are compatible with radial and weakly meshed topologies; on the other hand, some of these can be used just to cover basic analysis by demanding strict radial topology.

Power flow analysis based on phase frame

The phase frame refers to the algorithms that consider three phase power flow analysis for the DS. In this frame the size of the Y matrix that describes the power system is defined by the following expression:

$$\#\text{Rows Y} = \sum_{n=1}^{m} N_n \tag{2}$$

In (2) m corresponds to the number of phases of the system and Nn the number of nodes on the bus. The nodes refer to the number of phases connected to the bus m. For dealing with these features, this frame count with several approaches.

These approaches can handle DER and diverse network topologies in balanced or unbalanced conditions. The *forward and backward sweep* algorithm (W.H. Kersting, 2001; W. H. Kersting & Dugan, 2006) counts with three approaches: The current summation approach (Chang, Chu, & Wang, 2007), the power summation approach (Ghosh & Das, 1999) and the admittance summation approach (Rajičić & Taleski, 1998).

These approaches are inspired in Kirchhoff's Current Law (KCL) and Kirchhoff's Voltage Law (KVL); the aim is to determinate the grid voltage by injecting currents and compensate them until reaching convergence. The general algorithm for applying the forward and backward sweep is presented in TABLE 2-1(W.H. Kersting, 2001).

These approaches have also demonstrated to be suitable for DER specially with the power summation approach; where the generators are replaced by power injection units and the convergence is evaluated in terms of power (Moghaddas-Tafreshi & Mashhour, 2009). However, they represent a considerable computational burden when the topology of the grid is not strictly radial.

TABLE 2-1
A GENERAL ALGORITHM FOR APPLYING THE FORWARD AND BACKWARD SWEEP METHOD

Step	Description	Step	Description
1	Assume three phase voltages at the end nodes. The normal assumption is to use the nominal voltages.	6	Repeat steps 3 to 5 to reach the reference node. Consider all junction nodes
2	Compute the current at the end node using the assumed voltage.	7	Compare the calculated voltage at the reference node with the specified voltage of the source.
3	With this current, apply KVL to calculate the voltage on the precedent nodes.	8	If voltage difference is not within tolerance, use the specified voltage of the source and go forward to the next node using KCL, calculate the new voltage for the next nodes.
4	If there is a "junction" node (with lateral branches, apply steps 1 to 3 on the lateral branches and use the total current to compute the voltage at the junction node. This voltage will be the most recent voltage at this node.	9	The backward sweep continues to reach the end nodes. This action will complete the first iteration.
5	Apply KCL for calculating the current flowing from preceding nodes. Then compute Voltage on these nodes.	10	Repeat steps 1 to 8 using the new end voltages until reach voltage convergence.

On the other hand, the *compensation algorithm*, which is based on the *forward and backward sweep* algorithm, proposes to find compensation currents to complement the injection current vector (Zhu & Tomsovic, 2002). In this method the line sections in the radial network are ordered by layers away from the root node (substation); this order will allow to separate the DS in two matrixes: The breakpoint impedance matrix and the PV sensitivity matrix (for PV nodes).

With the complement injection current vector the compensation algorithm looks for opening loops for guaranteeing the radial topology of the system to solve (Cheng & Shirmohammadi, 1995). This algorithm can handle DG and weakly meshed networks. The general algorithm for applying the compensation method is shown in TABLE 2-2.

TABLE 2-2
A GENERAL ALGORITHM FOR APPLYING THE COMPENSATION METHOD

Step	Description	Step	Description
1	Read data, adjust network to radial representation and organize by layers.	5	Solve the system for finding the new injected currents vector J using the following equation: $[Z_B][J]^u = [V]^u$ Then, go to step 3
2	Form the breakpoint impedance matrix and the PV sensitivity matrix. Establish the maximum allowed errors e1 and e2.	6	If the PV node positive sequence voltage mismatch is lower than e2 go to step 8, otherwise, go to step 7.
3	Solve the 3 phase power flow.	7	Calculate the reactive current injection to eliminate the voltage mismatch using: $I_{iqx}^{(y)} = \left\|I_{iq}\right\|^{(y)} e^{j(90°+\delta_{vix}^{(y)})}$ ∂_{vix} are the voltage angles, and x is the phase (a, b, c) of the PV node. Go to step 3
4	If the Breakpoint impedance power flow mismatch is lower than e1 go to step 6, if not,	8	End

go to step 5.

Other algorithms like Implicit Z_{BUS} *Gauss* (T. H. Chen et al., 1991) and *Modified Newton* (or *Newton like*) (Garcia, Pereira, Carneiro, da Costa, & Martins, 2000) uses the sparse bifactored Y_{BUS} Matrix and equivalent current injections to solve the network equations. These methods recognize the good convergence performance of the Classic Newton-Raphson method; however, also identifies its problems when handling DS:

- Classic Newton-Raphson requires the Jacobian matrix with a rank approximately 4 times the Y_{BUS} to be recalculated on each iteration.
- Additionally, the Jacobian matrix cannot be decoupled of the smaller X/R of DS.

Implicit Z_{BUS} is based on the principle of superposition applied to the system bus voltages; these voltages are assumed to arise from two different contributions: the source voltage (substation) and the equivalent current injection. The convergence behavior of this method depends directly from the number of voltages specified for solving the system; this feature is handled by modeling each cogeenrator as a P-Q specified bus. As a result, the convergence rate is comparable to the Newton-Raphson approach. The general algorithm for this method is presented in TABLE 2-3 (T. H. Chen et al., 1991).

TABLE 2-3
A GENERAL ALGORITHM FOR APPLYING THE Z_{BUS} GAUSS METHOD

Step	Description	Step	Description
1	Calculate the initial Bus voltage estimation, component modeling and Build Y_{BUS} matrix.	5	Apply the voltage superposition algorithm by adding the deviation to the calculated voltage. $V^{(K+1)} = V_{NL} + DV^K$
2	Find the optimal order for the Y_{BUS} matrix, factorize this matrix and start the iteration counter K=1.	6	If the voltage converge goes to step 8, if not, go to step 7.
3	Compute the bus injected current I_n^K For all the elements connected to the grid (Power conversion devices PCD and power delivery devices PDD).	7	K = K+1, go to step 3
4	Calculate the voltage deviation due to the PCD and PDD. $I_n^K = \|L\|\|U\|DV^K$	8	Calculate bus voltages, Power flow, current flow, losses. End.

Direct methods such as *Bus Injection to Branch Current/Voltage (BIBC/BICV)* (Jen-Hao & Chuo-Yean, 2002), are focused in the factorization of the Y matrix for obtaining a block triangular form (BTF). This factorization is the base for solving faster the power system and reach convergence using a lower amount of iterations. The convergence criterion can be based on the voltages of the system, the currents and power. These algorithms can handle DG, weakly meshed and meshed networks (Davis & Natarajan, 2010; Kuzmin, Luisier, & Schenk, 2013). This type of solver also considers the total solution as the combination of partial results.

In the *Bus Injection* method the reference sources like substation and voltage sources are presented using their Thevenin equivalents, which are integrated into the Y_{BUS} matrix and to the current injection vector and then, the solution is found using a factorization algorithm. After the first iteration, the current injection vector is updated by including the currents injected by PCD and PDD. The general algorithm for applying the *Bus Injection* method is presented in TABLE 2-4.

TABLE 2-4
A GENERAL ALGORITHM FOR APPLYING THE BUS INJECTION METHOD

Step	Description	Step	Description
1	Build the Y_{BUS} matrix and the current injection vector.	5	Compute voltage mismatches.
2	Solve the system's voltage by factoring the Y_{BUS} matrix into its BTF. $$[V] = [Y_{BUS}]^{-1}[I]$$	6	Verify voltage convergence. If the system converges go to step 6, if not, go to step 2.
3	Calculate the complementary currents using the power and voltage at each node for each PCD and PDD.	7	Calculate bus voltages, Power flow, current flow, losses.
4	Update the current injection vector placing the new currents in the affected nodes.	8	End.

Other methods like the *loop impedance* change the reference bus frame for finding the power flow solution. In this method the topological characteristics of the network are combined with the concept of loop impedance matrix (T.-H. Chen & Yang, 2010). This approach has been extended to weakly meshed networks and the main feature is the elimination of the bus admittance matrix, which allows to find fast convergence for large-scale DS radially built (Stagg & El-Abiad, 1968). However, this advantage disappears when analyzing meshed networks, due to the increment of the number of iterations required for reaching convergence. The general algorithm for applying the *Loop impedance* method is shown in TABLE 2-5.

TABLE 2-5
A GENERAL ALGORITHM FOR APPLYING THE LOOP IMPEDANCE METHOD

Step	Description	Step	Description
1	Build the Z_{BUS} matrix using the primitive impedance matrix for the branches and the current injection vector.	6	Compute the bus voltage for PCD and PDD.
2	Form the K matrix using basic graph theory (T.-H. Chen & Yang, 2010).	7	Compute voltage mismatches. $$\Delta\bar{V}_{BUS}^{K} = \bar{V}_{BUS}^{(K-1)} - \bar{V}_{BUS}^{(K)}$$ If converges go to step 9, if not, go to step 8.
3	Calculate $$K^T[Z_{BUS}]K$$	8	Update Bus voltage. K = K + 1. Go to step 5
4	Set iteration number in 1.	9	Calculate bus voltages, Power flow, current flow, losses.
5	Calculate the injected current by PDC and PDD.	10	End.

These methods are well appreciated because practically every element connected to the power system can be easily represented within it. Using the bus admittance matrix, the relationships between the Y primitive matrixes of the interconnected elements can be modeled for static and dynamic analysis.

Power flow analysis based on sequence frame

Instead of the phase frame, the sequence frame decomposes the unbalanced DS into its positive, negative and zero sequences for being solved in a decoupled form. The advantage of this decomposition is that the positive sequence network is balanced, which allow using the single phase Newton-Raphson method for solving the network (Abdel-Akher, Nor, & Rashid, 2005; Lo & Zhang, 1993). Finally, the decoupled solutions are coupled again and this process keeps until reaching convergence, which is performed by evaluating the voltage instead of power.

In this approach the decoupled representation is made by separating the system into one three phase network with three phase line segments and unbalanced laterals with two and single phase line segment. These laterals are decoupled and replaced with equivalent current injections, which allows to complement the three phase networks defined above by combining classic Newton-Raphson method with backward-forward sweep for solving the unbalanced laterals. The algorithm for performing this solution is described in (Balamurugan & Srinivasan, 2011).

The criterion for evaluating the solution's convergence is the voltage mismatch. The advantage about this method is that it reduces the computational burden by simplifying the problem. The decoupled solution allows also the distributed analysis of the power system. This method can handle DER and the system topology can be radial or weakly meshed.

The general algorithm for applying the *sequence frame* solution method approach proposed by (Abdel-Akher et al., 2005) is presented in TABLE 2-6. Another approach proposed by (M. Kamh & Iravani, 2011; M. Z. Kamh & Iravani, 2010) is presented in TABLE 2-7. This last approach looks for reducing the computational burden, to include synchronous based generators and inverter based DERs.

TABLE 2-6
A GENERAL ALGORITHM FOR APPLYING THE SEQUENCE FRAME METHOD (ABDEL-AKHER ET AL., 2005)

Step	Description	Step	Description
1	Calculate the equivalent current injection for each unbalanced lateral using backward sweep in phase components.	5	Update the voltage of the unbalanced laterals using forward sweep in phase components.
2	Perform the power flow analysis using the main three phase networks in sequence reference frame. Include the lateral current injection.	6	Calculate power mismatch.
3	Use Stardard Newton-Raphson method or fast decoupled methods for solving the positive sequence network.	7	If convergence is reached go to step 8, if not, go to step 1.
4	Represent negative and zero sequence networks using nodal equations. After these calculations, the root voltages of the unbalanced laterals are known.	8	Calculate bus voltages, Power flow, current flow, losses. End.

The studies for new developments within the smart grid concept require more than one feeding node or voltage controlled source. Moreover, the structure of the DS may not be strictly radial, the modern distribution networks are mixed type networks. These features have led to consider mixed solution types for addressing the challenges proposed for smart grid studies. Additionally, non-conventional solution modes such as harmonics regimes (Dugan, Arrit, Henry, McDermott, & Sunderm, 2014) have been developed for creating power flow solvers that includes more than one frequency.

TABLE 2-7
A GENERAL ALGORITHM FOR APPLYING THE SEQUENCE FRAME METHOD (M. KAMH & IRAVANI, 2011; M. Z. KAMH & IRAVANI, 2010)

Step	Description	Step	Description
1	Decompose the power system into a three phase section with single phase laterals. The single phase laterals are not equipped with single phase DERs.	4	Calculate voltage mismatch instead of power mismatch. I convergence is reached go to step 5, if not, go to step 2.
2	Each phase lateral and its associated loads are lumped and represented as a single phase load at the node. This procedure is done with all nodes where laterals connect to the three phase networks.	5	Extract the phase voltages of the head nodes for all single phase laterals.
3	Perform three phase sequential power flow as shown in TABLE 2-6.	6	Perform the single phase backward/forward sweep power flow for each single phase lateral.

These frames have been used for building several applications oriented in power system design, testing new products/control strategies, academic research, monitoring and development of new technologies. Depending on the application area, currently there are recognized manufacturers and products as presented in Chapter 1. A summary of the frames and methods applied to DS is shown in Figure 2-1. TABLE 2-8 presents a summary of the method's features in phase and sequence frame (Balamurugan & Srinivasan, 2011).

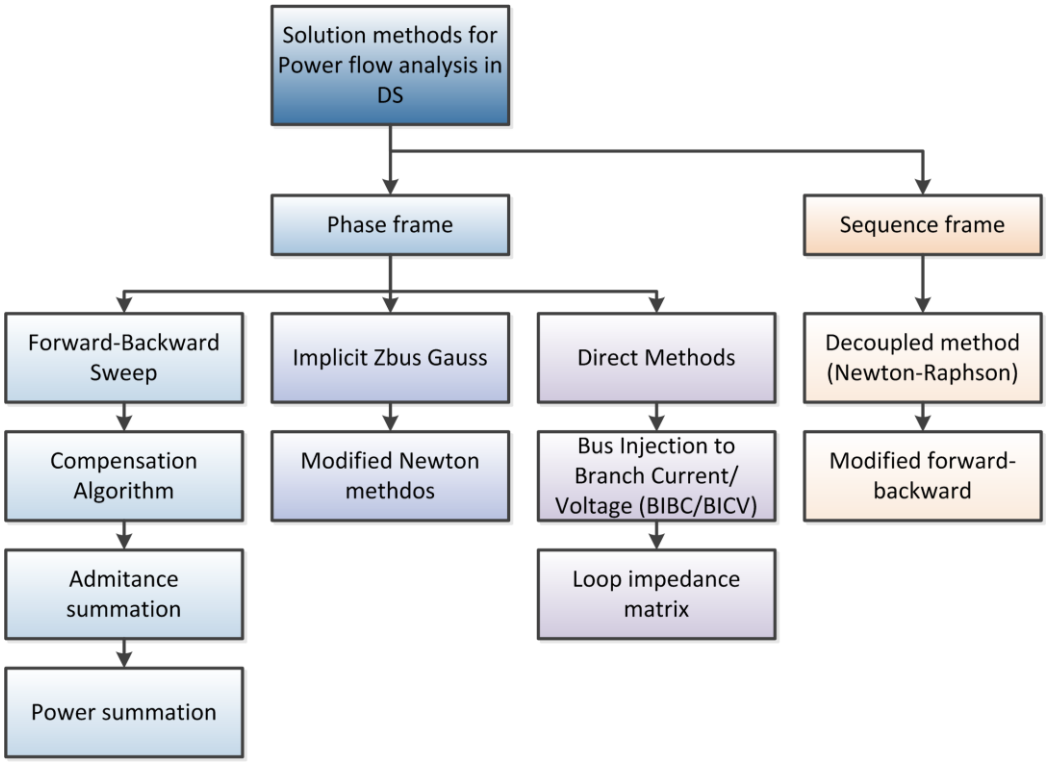


Figure 2-1. Solution methods used for power flow analysis in DS

TABLE 2-8
SUMMARY OF METHODS USED FOR POWER FLOW ANALYSIS IN THE DS (BALAMURUGAN & SRINIVASAN, 2011)

Algorithm	Frame	Topologies Covered	Covers Unbalanced	DER	Convergence Criterion
Forward/ Backward Sweep	Phase	Radial	Yes	No	Voltage Mismatch
Compensation	Phase	Radial/Weakly Meshed	Yes	Yes	Power Mismatch
Implicit Z_{BUS}	Phase	Radial/Weakly Meshed/ Meshed	Yes	Yes	Voltage Mismatch
Newton Like	Phase	Radial/Weakly Meshed/ Meshed	Yes	Yes	Voltage/Current/ Power Mismatch
BIBC/BICV	Phase	Radial/Weakly Meshed	Yes	Yes	Load Current Mismatch
Loop Impedance	Phase	Radial/Weakly Meshed	Yes	No	Voltage Mismatch
Sequence Frame 1	Sequence	Radial/Weakly Meshed	Yes	No	Voltage Mismatch
Newton Based	Sequence	Radial/Weakly Meshed/ Meshed	Yes	Yes	Positive sequence voltage Mismatch

Dynamic Simulation

The large-scale penetration of DERs on DS impacts the dynamic behavior of all PCD interconnected through this system, which has motivated to perform transient analysis on the distribution feeders (Nagarajan & Ayyanar, 2014). This kind of analysis, performed traditionally in the time domain,

represents an important computational burden, thus limiting the size of the system modeled and making emphasis only in certain parts of the DS through equivalent circuits.

However, Transient Stability Analysis (TSA) allows for modeling small-scale, medium-scale and large-scale DS considering the dynamic features of PCD in low computational times. These features are integrated to the solution algorithms using a mathematical approach based on differential algebraic equations (DAE) (Ishchenko, Myrzik, & Kling, 2006). These are integrated in every simulation step for determining the phase angle of voltage on the next simulation step. As a result, the PCD are sensitive to switch tripping operation, allowing to perform dynamic studies such as determining controllers bandwidth, to evaluate the effect of multiple voltage correction devices, anti-islanding studies, among others (Tianshu, Hongwei, Peng, & Qixun, 2008).

There are several methods for performing the integration of DAE; depending on the application needs, there are a large number of methods for covering specific needs. The improved Euler-Cauchy method or Heun's method has been used for addressing this problem in several simulators such as OpenDSS (Dugan & McDermott, 2011) and ePhasorSim (Jalili-Marandi, Ayres, Ghahremani, Belanger, & Lapointe, 2013).This method count with two steps: The predictor and the corrector (Electric Power Research Institute, 2013a).

The predictor step offers algorithm developers the opportunity to insert a predictor formula at the beginning of the process. It may be different from the corrector step only in that it updates the initial value of the state variables for this time step based on derivatives at previous answer (Electric Power Research Institute, 2013a, 2013b). The predictor and corrector equations are as follows:

$$\boldsymbol{y_{n+1}^{*} = y_n + hf(x_n, y_n)} \qquad \textbf{2.1}$$

$$\boldsymbol{y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{*})]} \qquad \textbf{2.2}$$

Equation 2.1 corresponds to predictor step and equation 2.2 describes the corrector step. In these equations x_n, x_{n+1} are the independent variables, y_n, y_{n+1} are the dependent variables. y_n is the initial value defined by the previous calculated state and y_{n+1} is the unknown value that need to be calculated. This method gives extra stability to the results due to the correction step (Electric Power Research Institute, 2013a). The algorithm for applying the improved Euler-Cauchy method is presented in TABLE 2-9.

TABLE 2-9
ALGORITHM FOR APPLYING THE IMPROVED EULER-CAUCHY INTEGRATION METHOD (NAGARAJAN & AYYANAR, 2014)

Step	Description
1	Formulate the differential equations and the ordinary differential equations separately. $$y' = f(x, y)$$ $$0 = g(x, y)$$
2	Solve the ordinary differential equations using a numerical integration technique. $$y' = f(x, y)$$
3	With the new calculated state of *y* solve the algebraic equations $$0 = g(x, y)$$
4	Increase the time step, proceed to the next iteration until reaching the final simulation time

Nowadays, there exist several models for representing the dynamic behavior of DER (Apostolov, 2005; Boemer, Gibescu, & Kling, 2009; Stewart et al., 2012); these models follow the same pattern of the integration methods for being compatible with these techniques. As an example, OpenDSS models the dynamic behavior of synchronous generators based on the speed and angle of the shaft as follows:

$$\frac{dV}{dt} = \frac{PShaft - PTerm - Dv}{M} \quad 2.3$$

$$\frac{d\theta}{dt} = v \quad 2.4$$

Where,

- v = shaft speed (velocity) relative to the synchronous speed
- θ = shaft, or power, angle (relative to the synchronous reference frame)
- *Pshaft* = shaft power
- *Pterm* = terminal power out
- *D* = damping constant
- *M* = mass

Other authors have considered more sophisticated devices such as three phase voltage inverters (Areerak, 2011; Nagarajan & Ayyanar, 2014; Stewart et al., 2012), which can be represented using DAE. The following example corresponds to the representation of a three phase voltage inverter in the dq reference frame:

$$L\frac{di_d}{dt} = e_d - R_{id} - v_d + \omega L i_q \quad 2.5$$

$$L\frac{di_q}{dt} = e_q - R_{iq} - v_q - \omega L i_d \quad 2.6$$

Where,

- L is the grid inductance and the filter inductance of the three phase inverter
- R is the resistance value used for filtering the harmonics of the inverter
- e_x represents the voltage at phase x (d or q)
- v_x represents the voltage at phase x (d or q)
- ωL_{ix} are the cross coupling terms (d and q)

Equations 2.1 and 2.2 must be fixed for being compatible with the DAE that describe the behavior of a specific model. For example, the trapezoidal integration formula applied to integrate the angle θ in 2.4 is as follows:

$$\theta_{n+1} = \theta_n + \frac{\Delta t}{2}\left[\left.\frac{d\theta}{dt}\right|_n + \left.\frac{d\theta}{dt}\right|_{n+1}\right] \quad 2.7$$

Comparing equation 2.2 and 2.7 it is simple to identify the parts modified on the equation. In this case, h corresponds to the time step and the other arguments refers to the differential equations for solving θ. As a result of including these DAE to the models for solving the system power flow, the dynamic behavior of the entire network can be simulated, revealing the effects in different parts of the DS when DER are connected and disconnected to the system in different instants of the simulation (events). Figure 2-2 shows the dynamic behavior of the voltage (p.u.) after a separation of 1 second from the DS

(EPRI's circuit 7 – 2452 nodes). The measurements were taken at different points of the DS and the step time is 4.5ms.

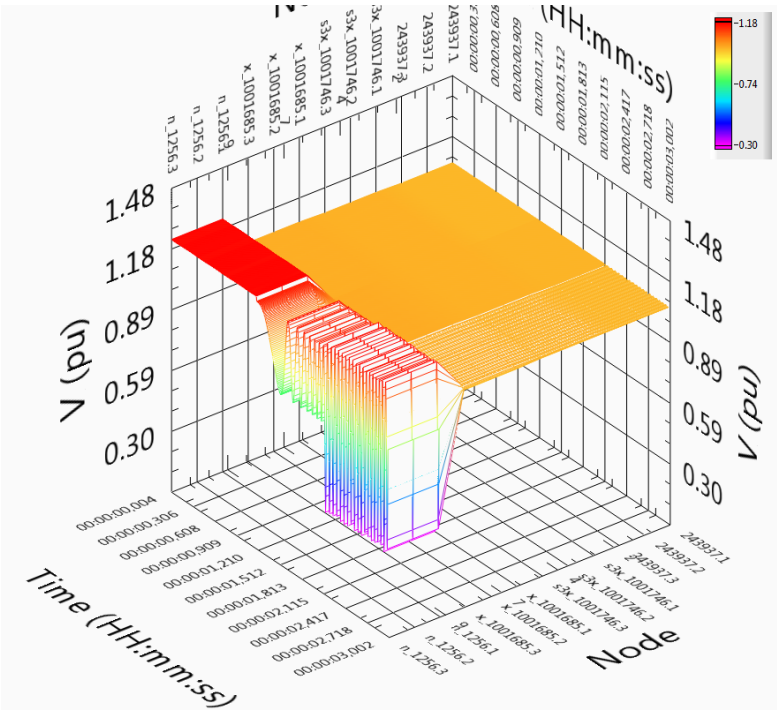


Figure 2-2. Dynamic behavior of voltage (p.u.) after the disconnection and reconnection of a 1MW generator from EPRI's circuit 7- 2452 nodes. Step time 4.5 ms. Graph generated with DSSim-PC (Montenegro, 2013).

Challenges for DS simulation

The frames presented above contemplate the system as a single block, this feature represents several disadvantages related with the performance of the simulator and the use of available computing resources. These disadvantages are described as follows:

The deterministic computing cycles

When the number of nodes of the DS increases also the computing time, which makes difficult to guarantee deterministic computing cycles for RT-HIL simulation. When considering the computing problem as a single unit, normally it must be executed using a single processor (core); this means that the memory assignment is limited to the memory space linked to the local core by the Operative System (OS), using just a part of the available hardware resources in a sequential form.

Complexity of the simulation

The simulation of sophisticated control devices interacting within the DS can add an important computational burden; thus affecting the performance of the simulation depending on the number and nature of the devices (control, monitoring). Nowadays, there are more and more control devices interacting within the DS, their dynamic behavior and the way this interaction takes place could lead to important computation cycles that affects directly the total simulation time.

Topology changes

Topological changes need an important computation time when being considered within the simulation. Depending on the number of phases of a switch/line, the number of cells to update when they trip can be significant. The power flow analysis within the smart grid concept considers topology changes all the time. Activities such as feeder reconfiguration and fault detection and isolation demand that the simulation can react to these changes with speed and accuracy, especially for long sequential time simulations (like yearly simulations) and RT-HIL simulation.

Hybrid power systems

These methods can solve a single frequency simultaneously, limiting the complexity of the power system's model and the scope of the simulation. The Y matrix built for describing the power system is calculated using only one frequency, which lead to code special simulation modes such as harmonics (Dugan et al., 2014) using sequential programming. As a result, the total computing time of the simulation increases as a function of the number of frequencies to be simulated.

Conclusion

The existing methods for power flow analysis of DS are not compatible with multicore hardware architectures, which is the consequence of asynchronous evolution processes. This last sentence proposes that while computing hardware evolves into multicore processing, the methods for power flow analysis have evolved considering classical sequential programming, instead of keeping the same direction of the hardware.

In this thesis, these challenges will be addressed by using our approach, called Diakoptics Based on Actors (A-Diakoptics) as mentioned in Chapter I. The results are going to be integrated into the open source simulator called OpenDSS from EPRI (Dugan et al., 2014; Dugan et al., 2010; Dugan & McDermott, 2011) for evaluating the contribution to improve existing simulation tools. This simulator was chosen due to the following features:

1. The code is well coded. It was conceived in term of classes and objects, making easier the code reading and modification.
2. The COM interface is a good alternative for interfacing external software without compromising the simulator performance when accessing it using early bindings instead of late bindings, which is the default method active in most programming languages (Rogerson, 1997).
3. It includes a good number of models and control devices and there are several study cases documented.
4. Includes simulation modes such as harmonics that cannot be found in other simulation tools.
5. The solver of OpenDSS uses direct methods for performing power flow analysis.

These features make of OpenDSS a good candidate for applying A-Diakoptics and to evaluate how its performance gets improved. Additionally, the test cases and applications proposed by this thesis will be supported by the robustness and documented cases available with the simulation tool.

Chapter3. Distribution System Load Flow Analysis using Diakoptics

CHAPTER3. DISTRIBUTION SYSTEM LOAD FLOW ANALYSIS USING DIAKOPTICS.....43

CHAPTER 3 44

DISTRIBUTION SYSTEM LOAD FLOW ANALYSIS USING DIAKOPTICS 44

Diakoptics and the piecewise methods 44

The primitive networks 44

The orthogonal networks 47

The interconnected equivalent network 50

THE MULTILEVEL APPROACH 52

Separating the Connections matrix Z_{CC} 52

Distributing tasks 54

Simulation of Hybrid power systems 55

THE COMPUTATIONAL MODEL OF DIAKOPTICS 57

Creating primitive networks (Islands) 58

Distributed computation 64

CONCLUSIONS 65

Chapter 3

Distribution system load flow analysis using Diakoptics

Diakoptics and the piecewise methods

Diakoptics is a method used for the decomposition of large-scale networks in a set of independent sub-networks. This method and its variations are well known as the piecewise methods (Kron, 1963), which were designed for Transmission Systems (TS) and their applications include economic dispatch, power flow analysis and transient stability analysis. Diakoptics was formally proposed by Kron in the 55's (Kron, 1955) and since then several authors have proposed different interpretations for the algorithm.

One of these interpretations is Z-Diakoptics, which was used for performing power flow analysis using a distributed network of computers. This was proposed by Happ in the 70's (Happ, 1971) and proposes the decomposition of the system using equivalent impedance matrixes. However, when the computing architectures became more powerful the piecewise methods became irrelevant; this because one computer was enough for performing power flow analysis on TS.

Nowadays, the challenges proposed for smart grid studies in small-scale, medium-scale and large-scale Distribution Systems (DS) make of Diakoptics an alternative for evolving classical Transient Stability Analysis (TSA) methods into suitable for multithread processing. This method allows considering the primitive sub - network as independent systems (Klos, 1982). This is possible by decomposing the interconnected network into primitive and orthogonal networks.

Primitive networks are the sub-networks formed when separating the interconnected system; orthogonal networks are those that interconnect the primitive networks. The primitive networks do not regard the frontier effects produced by the interconnection links, they are declared such as standalone networks. On the reverse, the orthogonal networks include the information about the network view from the link branch.

Using these two representations of the original system, the new model is assembled by complementing it using graph theory for defining how these independent sub-networks are interconnected; as a result, the new representation of the original system will be composed by primitive networks, orthogonal networks and a particular type of matrix called the *contours matrix* (also called *tensors (Kron, 1963)*, *links*, among other names).

The primitive networks

The primitive networks are those networks represented using the primitive reference frame defined in Diakoptics (Happ, 1980). In this frame, an electrical network can be described as a set of branches interconnected, which have electrical currents flowing through them and voltages causing this flow in their terminals.

The primitive networks represent the building blocks of any network based on the branches in the primitive (Happ, 1966), which define the "torn model" of a bigger network to be specified (Happ, 1971), which means that a primitive network can have any number of branches, can be composed by one or several branches for being transformed into another interconnected network. This network is used as reference network because it is the representation of a basic unit of a larger network.

This network represents an easier network for analyzing and formulate than a large-scale network, preserving the same principles of electrical networks, but useful for making emphasis in parts of the system of special interest.

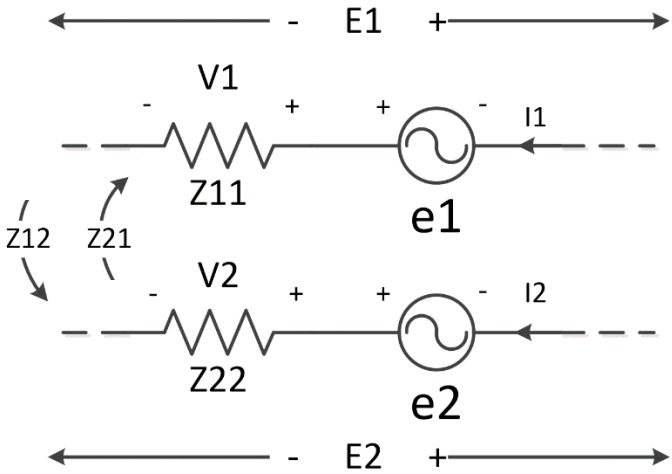


Figure 3-1. Link branch of an electrical system

Consider the link branches shown in Figure 3-1. As can be seen in this figure two link branches are modeled one near to the other; these branches can be mathematically modeled as follows:

$$\begin{bmatrix} V1 \\ V2 \end{bmatrix} = \begin{bmatrix} Z11 & Z12 \\ Z21 & Z22 \end{bmatrix} \begin{bmatrix} I1 \\ I2 \end{bmatrix} \qquad 3.1$$

Where Z11 and Z22 are the self-impedances of the branches, and Z12 and Z21 are the mutual impedances between them. However, in this representation it is not clear how these branches are connected; in fact, it is not possible to know if they are interconnected. This preliminary conclusion is used for explaining how two parts of the same system can be separated by neglecting the effect of the link branches between them. This means that this basic representation must be complemented with other representations for describing the relationships between them.

Example 3.1

Consider the radial power system shown in Figure 3-2.

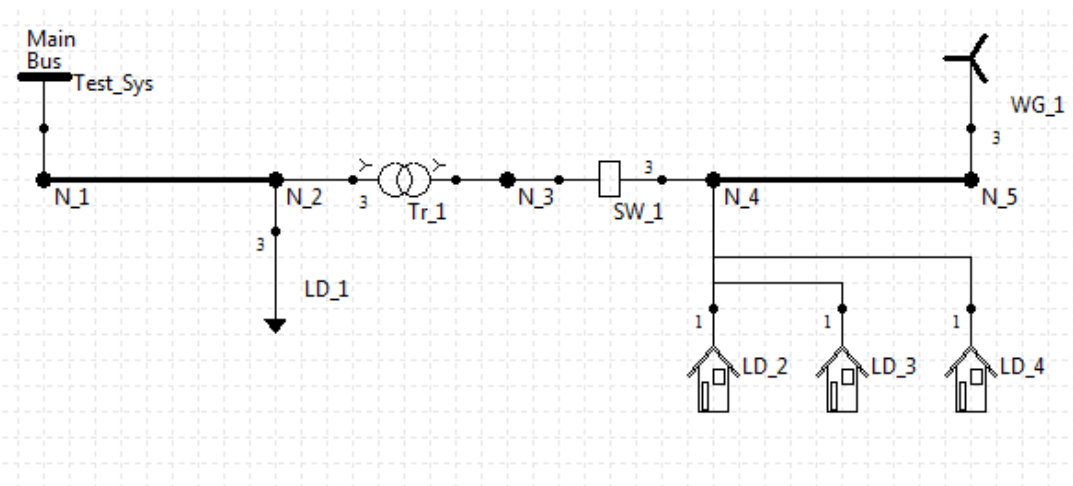


Figure 3-2. System under study

The Y_{BUS} matrix that defines the system presented in Figure 3-2 is shown in Figure 3-3.

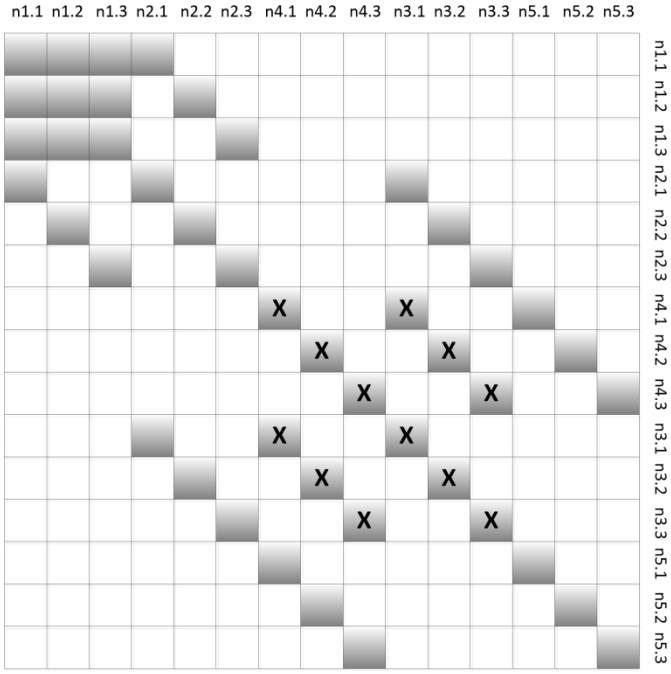


Figure 3-3. Y_{BUS} matrix for modeling the proposed system

In Figure 3-3 the colored cells are different from 0 and the cells marked with *X* correspond to the cells where the values of the primitive Y of the switch *SW_1* are included. This switch is modeled as a 3 phase line with a very low resistance (Dugan & McDermott, 2011). Let suppose that switch SW_1 trips separating the proposed power system in two parts, the resulting Y_{BUS} matrix will be as shown in Figure 3-4.

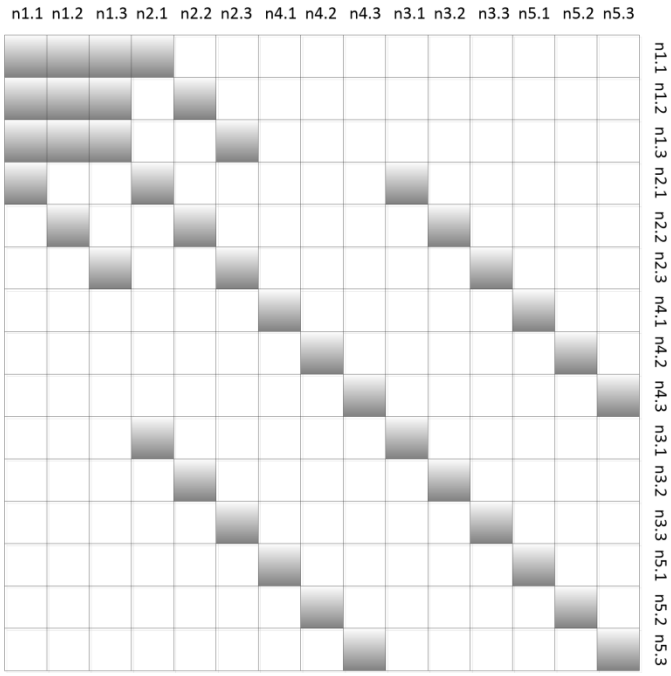


Figure 3-4. Y_{BUS} matrix after switch SW_1 trips

However, the order of the columns and rows in the resulting matrix shown in Figure 3-4 does not allow visualizing the separation clearly; in fact, is evident that the matrix distribution can be optimized for making evident how the system is separated into two sub-systems with strongly interconnected elements. The new distribution proposed for the Matrix in Figure 3-4 is shown in Figure 3-5.

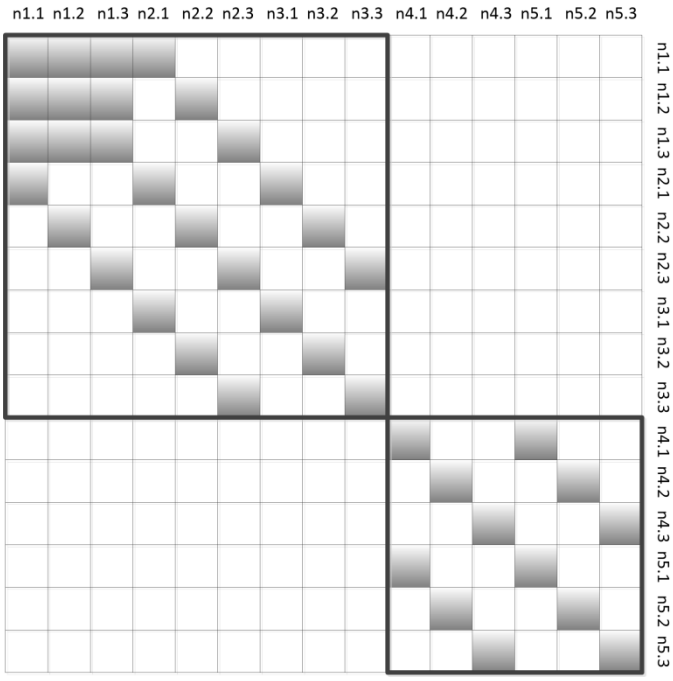


Figure 3-5. Y_{BUS} reorganized after switch SW_1 tripping

As can be seen in Figure 3-5, the proposed ordering allows visualizing how the radial power system presented in Figure 3-3 gets separated into two sub-networks. Additionally, in this new representation of the network the new values of the SW_1 have been removed from the mathematical representation, it is like if the switch was removed from the system creating two independent sub-systems.

These two independent sub-systems are the primitive networks of the interconnected network. Nevertheless, if it is desired to keep separating the system in more parts for considering all the possible topologies of the system, it can be done; the only requirement is to remove the lateral effects of the link branches between the primitive networks. The new matrix containing the primitive networks is called the tree's matrix (Y_{TT} or Z_{TT}^{-1}) (Happ, 1967; Uriarte & Butler-Purry, 2006).

The orthogonal networks

After separating the power system into several primitive networks it is necessary to identify how these networks are interconnected; this will allow to transform the voltages and currents of the primitive network to those of another network through the transforming tensors and vice versa (de Hon & Arnold, 2010; Happ, 1966; Mrozowski, Niedzwiecki, & Suchomski, 1995). The tensors are defined using graph theory: each tensor has direction and connects nodes of different primitive networks for representing the interconnected network. Each connection is represented as a 1 or -1 between the interconnected nodes. *Tensors* are also called *contours*. The transposed contour matrix for describing the connections between the primitive networks generated in Figure 3-5 is shown in Figure 3-6.

In Figure 3-6, the contour matrix has three contours due to the three phase nature of the switch SW_1, which is the link branch between the two primitive networks. This matrix is the transformation matrix for passing from the tree's matrix (primitive networks) to the interconnected network. As a result of

this transformation, a new network is formed called the orthogonal network (Olobaniyi, Nouri, & Ghauri, 2012).

$C^T =$

	n1.1	n1.2	n1.3	n2.1	n2.2	n2.3	n3.1	n3.2	n3.3	n4.1	n4.2	n4.3	n5.1	n5.2	n5.3
C_1							-1			1					
C_2								-1			1				
C_3									-1			1			

Figure 3-6. Transposed contour matrix in the case presented in Figure 3-5

The orthogonal networks are based on the concurrent existence of open and close paths of currents and voltages (Ney & Le Maguer, 2002). These paths flow through the designated contour: this way, the number of independent paths in a connected network is equal to the number of link branches in the network. This representation considers that currents and voltages are superimposed upon the assigned contours; as a consequence, there will be open-path and close-path voltages and currents. The variables associated to the contours will be known as orthogonal variables representing the orthogonal reference frame (Kron, 1955).

These variables reflect the system's topology (open and close paths), which satisfy Kirchhoff Current Law (KCL) without the need for specifically invoking this law (Happ, 1971). The orthogonal network will be different from the interconnected network but electrically equivalent to it.

As shown in Figure 3-5, the tree's matrix is referenced in the *node to node* frame, this is, connects all the nodes of the system using a link branches represented by admittances. However, in equation 3.1 the expected relationship between current and voltage is ruled by an impedance matrix *Z*. This means that for transforming the tree's matrix into the electrical equivalent of the interconnected system this matrix must be turned from its admittance representation to its impedance equivalent (Brown, Carter, Happ, & Person, 1963).

This transformation from admittance to impedance (Y_{TT} to Z_{TT}) cannot be performed directly because as mentioned above, all primitive networks are independent and electrically isolated. So, the equivalent impedance matrix will be obtained by calculating the equivalent impedance matrix of each primitive matrix, and replacing the admittance sparse matrix with its new dense equivalent.

On the other hand, it is convenient to keep the sparse representation of the admittance matrix for solving the primitive network using direct methods and for saving memory when handling large-scale networks (Davis & Natarajan, 2010; Kuzmin, Luisier, & Schenk, 2013). These considerations generate the following rules for solving the interconnected network:

- For solving primitive networks the sparse admittance matrix will be used.
- For finding the orthogonal matrix of the primitive networks, the impedance representation will be used.

However, for describing the relationships between the contour matrix and the trees matrix it is necessary to take them to the same reference frame. The contour matrix is referenced in the *contour to node* frame, connecting nodes through the proposed contours as shown in Figure 3-6, while the trees matrix is referenced to the *node to node* frame.

Therefore, the order on that these matrices are operated will define the destination reference frame, which can inferred using matrix algebra as shown in Figure 3-7. In this Figure transformation matrix A is used for taking matrix B from reference frame YY to reference frame XX.

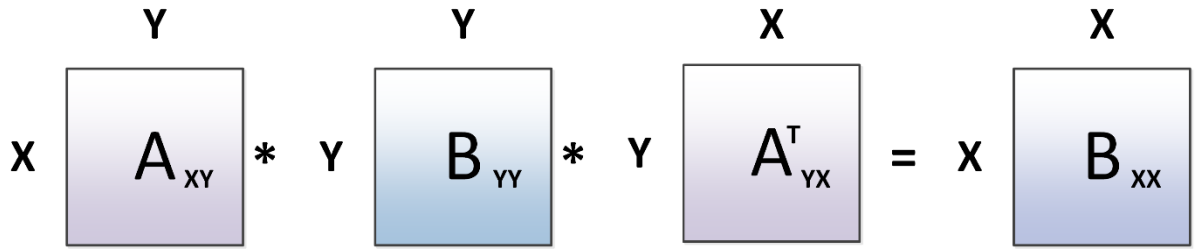


Figure 3-7. Example of reference frame moving using a transformation matrix. In this case, matrix A is the transformation matrix for taking matrix B from the domain YY to the domain XX.

The same principle can be used to describe the relationship between the contours matrix and the trees matrix; in this case, the contours matrix will be the transformation matrix for taking the trees matrix from the isolated domain to the interconnected domain. As a result, it is expected that the obtained matrixes will describe how the interconnections between the primitive matrixes through their link branches are, which can be obtained as follows.

$$\boldsymbol{Z_{TT} = C_{TC} Z_{TT} (C_{TC})^T} \qquad \textbf{3.2}$$

$$\boldsymbol{Z_{CC}{}' = (C_{TC})^T Z_{TT} C_{TC}} \qquad \textbf{3.3}$$

$$\boldsymbol{Z_{CT} = (C_{TC})^T Z_{TT}} \qquad \textbf{3.4}$$

$$\boldsymbol{Z_{TC} = Z_{TT} C_{TC}} \qquad \textbf{3.5}$$

As can be seen in equation 3.2 this transformation keeps the original matrix within the same reference frame, which makes unnecessary this transformation; nevertheless, orthogonal networks obtained in equations 3.3, 3.4 and 3.5 are referenced into new reference frames.

Comparing these new matrixes with the structure proposed in equation 3.1, each one of these can be associated to be a part of this model. Due to the obtained frame in Equations 3.4 and 3.5, these equations can be interpreted as the mutual impedances between the contours (link branches) and the primitive networks.

The self-impedances matrix of the primitive networks is the tree's matrix. However, for equation 3.3 the components of this orthogonal matrix only include the mutual impedances between the primitive networks, which means that the diagonal of this matrix must be complemented for including the information about the link branches as follows:

$$\boldsymbol{Z_{CC} = (C_{TC})^T Z_{TT} C_{TC} + Z_{LLD}} \qquad \textbf{3.6}$$

In Equation 3.6 Z_{LLD} is a diagonal matrix where the diagonal components are the self-impedances of the link branch, complementing the connections orthogonal matrix Z_{CC} (Happ, 1967). The equivalent Z_{TT} matrix obtained from the tree admittance matrix Y_{TT} presented in Figure 3-5, is shown in Figure 3-8.

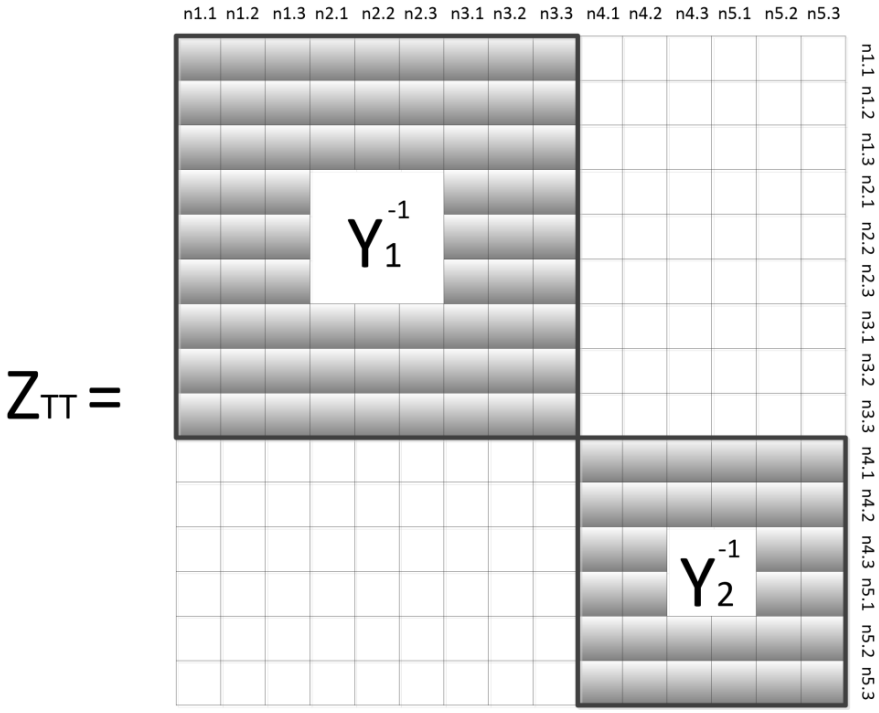


Figure 3-8. Equivalent Z_{TT} matrix obtained from the tree admittance matrix Y_{TT}

The interconnected equivalent network

Considering the orthogonal networks built above and the structure proposed in equation 3.1, the interconnected equivalent network can be proposed as follows:

$$\begin{bmatrix} \boldsymbol{E_T} \\ \boldsymbol{e_C} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Z_{TT}} & \boldsymbol{Z_{TC}} \\ \boldsymbol{Z_{CT}} & \boldsymbol{Z_{CC}} \end{bmatrix} \begin{bmatrix} \boldsymbol{I_0} \\ \boldsymbol{i_C} \end{bmatrix} \qquad \mathbf{3.7}$$

This new structure involves the currents injected to each primitive network considered independent from the rest of the system, and the currents flowing through each link branch for complementing the partial solutions delivered by the primitive networks. However, at this point the question is: how to find these partial solutions for obtaining the total solution.

For solving this question the procedure shown in Figure 3-9 is proposed.

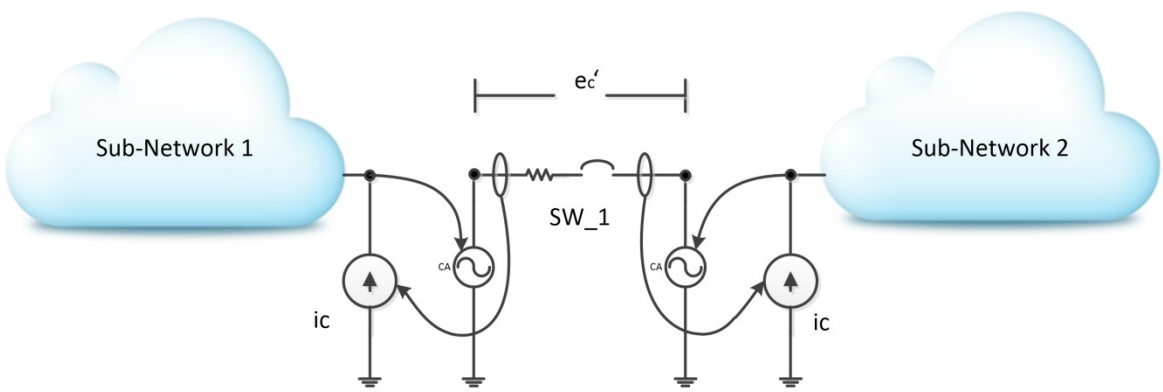


Figure 3-9. The proposed procedure for solving the interconnected equivalent (physical interpretation)

In this procedure the following steps are proposed:

- To solve the primitive networks independently
- With the partial solution delivered for each primitive network, excite the terminals of the link branches.
- Once the link branches are excited, the voltages at the terms of these branches can be calculated.
- Finally, add the partial solutions with the voltages provided by the link branches to find the total solution.

In equation 3.7 the vector I_0 corresponds to the currents injected to each primitive matrix, i_c is the vector of currents flowing through the link branches, E_T is the vector containing the voltages calculated for solving the interconnected network and e_c is the voltage on the link branch. Nevertheless, because this structure is based on current injection e_c will be equal to 0.

Considering the structure presented in equation 3.7 the voltage on the link branches is given by:

$$\boldsymbol{e_c = Z_{CT} I_0 + Z_{CC} i_c} \qquad \textbf{3.8}$$

And the current passing through the branch will be given by:

$$\boldsymbol{i_c = Z_{CC}^{-1}(e_c - Z_{CT} I_0)} \qquad \textbf{3.9}$$

However, as mentioned above, e_c is equal to 0, which makes that i_c will be equal to:

$$\boldsymbol{i_c = Z_{CC}^{-1}(-Z_{CT} I_0)} \qquad \textbf{3.10}$$

This means that the voltage in the terms of the link branch will be given by:

$$\boldsymbol{e_c{}' = -Z_{CT} I_0} \qquad \textbf{3.11}$$

The total solution of the system E_T will be given by:

$$\boldsymbol{E_T = Z_{TT} I_0 + Z_{TC} i_c} \qquad \textbf{3.12}$$

By replacing *each* from equation 3.10 into 3.12 for the new expression will be:

$$\boldsymbol{E_T = Z_{TT} I_0 - Z_{TC} Z_{CC}^{-1} Z_{CT} I_0} \qquad \textbf{3.13}$$

This way, and considering that the total solution is composed of two parts, equation 3.13 can be rewritten as follows:

$$\boldsymbol{E_T = Z_{TT} I_0 - Z_{TC} Z_{CC}^{-1} Z_{CT} I_0 = E_0 + E_1} \qquad \textbf{3.14}$$

Where E_0 corresponds to the voltages provided by the partial solutions of the primitive networks, and E_1 are the complementary voltages when considering the link branches between the primitive networks. This formulation is the original formulation made by Happ for the Z-Diakoptics approach (Happ, 1967). However, for the approach proposed in this work the solution of the primitive networks is not obtained as proposed in equation 3.14.

As mentioned above, to solve the primitive networks the matrix to be used is tree admittance matrix Y_{TT}. As a consequence, the left part of the equation 3.13 can be solved using several methods

depending on the size of the primitives. In this work, the proposed solution for the primitive networks is to solve each primitive separately using direct methods, which will deliver an updated vector I_0 that will include the compensation currents of the PCD. This consideration affects the solution of the complementary voltages of equation 3.13, due to I_0 is different in time than the vector used for solving the primitive networks. Assume that n the time instant where some event occurs, because of the difference between the vector used for solving the primitive networks and the complementary voltages (I_0 in equation 3.13), equation 3.13 can be rewritten as follows:

$$\boldsymbol{E_T = Z_{TT} I_{0(n)} - Z_{TC} Z_{CC}^{-1} Z_{CT} I_{0(n+m)}} \qquad \textbf{3.15}$$

Where m is the time instant where the complementary voltages are calculated. The difference between both is that $I_{0(n)}$ is the original vector of injected currents without the contribution of PCD; on the other hand, $I_{0(n+m)}$ is the same vector plus the complementary current injections delivered by the PCD. This means that both calculations, E_0 and E_1, happens in different time instants. The proposed algorithm for solving the interconnected equivalent network is shown in TABLE 3-1.

TABLE 3-1
GENERALIZED ALGORITHM FOR APPLYING DIAKOPTICS

Step	Description
1	Solve each primitive network independently using $I_{0(n)}$. Save the excitation vector (currents) of each solution for building the vector $I_{0(n+m)}$. If the primitive network has no excitation, send the load's characteristics on the network for adding the new currents in step 6.
2	Find the excitation voltage sources for Z_{CC}. $e_{c'} = -Z_{CT} I_{0(n+m)}$
3	Find the contour currents using the orthogonal network Z_{CC}. $i_c = Z_{CC}^{-1} e_{c'}$
4	Find the complementary voltages E_1. $E_1 = Z_{TC} i_c$
5	Calculate the total solution. $E_T = E_0 + E_1$
6	If some primitive network has no excitation, re-calculate the vector I_0 according to equation (5) and go back to step 2. Repeat this procedure until the desired convergence is reached.

The Multilevel Approach

Depending on the size of the connections matrix Z_{CC}, to perform the steps 2 to 5 of the algorithm presented in TABLE 3-1 could represent a bottleneck in terms of computation time. However, sue to the linear nature of the model this matrix can be separated in many parts as needed for being processed distributive (Happ, 1980).

Separating the Connections matrix Z_{CC}

The connections matrix Z_{CC} is a square matrix in the reference frame *contour to contour*. As described in the equation 3.6, this matrix includes the information about the mutual and self-impedances of between link branches and primitive networks as follows:

$$Z_{CC} = \begin{array}{c|ccccc} & c_1 & c_2 & c_3 & \cdots & c_k \\ \hline c_1 & x & x & x & \cdots & x \\ c_2 & x & x & x & \cdots & x \\ c_3 & x & x & x & \cdots & x \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_k & x & x & x & \cdots & x \end{array} \quad 3.16$$

Consider to perform the procedure described in TABLE 3-1 but using only the first column of matrix 3.16. This procedure will be as follows:

$$Z_{CC(1)} = C_1^T Z_{TT} C_1 + Z_{LLD(1)} \quad 3.17$$

$$Z_{CT(1)} = C_1^T Z_{TT} \quad 3.18$$

$$Z_{TC(1)} = Z_{TT} C_1 \quad 3.19$$

This way, the new structure will be a partial solution of the system. This partial solution can be written in terms of the total solution of the system:

$$\begin{bmatrix} E_T \\ 0 \end{bmatrix} = \begin{bmatrix} Z_{TT} & Z_{TC(1)} \\ Z_{CT(1)} & Z_{CC(1)} \end{bmatrix} \begin{bmatrix} I_0 \\ i_C \end{bmatrix} \quad 3.20$$

By assuming that $I_{(n)} = I_{(n+m)}$, equation 3.20 can be rewritten as:

$$E_{T(1)} = Z_{TT} I_0 - Z_{TC(1)} Z_{CC(1)}^{-1} Z_{CT(1)} I_0 \quad 3.21$$

As can be seen in equation 3.12, this first partial solution is composed by the injected current vector and the equivalent partial impedance created when selecting the first column of the original Z_{CC}. This partial impedance is given by:

$$Z_{TT(1)} = Z_{TT} - Z_{TC(1)} Z_{CC(1)}^{-1} Z_{CT(1)} \quad 3.22$$

So, this partial impedance will be the starting point to keep separating the interconnected equivalent system. Consider the same procedure described from equations 3.17 to equation 3.22 but using $Z_{TT(1)}$ as the tree's matrix of the system, and using the column 2 of the original Z_{CC} matrix.

The partial solution of the system will be given by:

$$Z_{CC(2)} = C_2^T Z_{TT} C_2 + Z_{LLD(2)} \quad 3.23$$

$$Z_{CT(2)} = C_2^T Z_{TT} \quad 3.24$$

$$Z_{TC(2)} = Z_{TT} C_2 \quad 3.25$$

$$\begin{bmatrix} E_T \\ 0 \end{bmatrix} = \begin{bmatrix} Z_{TT(1)} & Z_{TC(2)} \\ Z_{CT(2)} & Z_{CC(2)} \end{bmatrix} \begin{bmatrix} I_0 \\ i_C \end{bmatrix} \quad 3.26$$

Then, for the next partial solution the partial tree's matrix can be described as follows:

$$\boldsymbol{Z_{TT(2)} = Z_{TT(1)} - Z_{TC(2)} Z_{CC(2)}^{-1} Z_{CT(2)}} \qquad \mathbf{3.27}$$

The general expression for n columns can be written:

$$\boldsymbol{Z_{TT(n)} = Z_{TT(n-1)} - Z_{TC(n)} Z_{CC(n)}^{-1} Z_{CT(n)}} \qquad \mathbf{3.28}$$

The total solution of the system will be given by the sum of all the partial solutions

$$\boldsymbol{E_T = \sum_{l=1}^{k} \left(Z_{TT(l-1)} I_{0(n)} - Z_{TC(l)} Z_{CC(l)}^{-1} Z_{CT(l)} I_{0(n)} \right)} \qquad \mathbf{3.29}$$

However, by considering that the injected current vector I0 is updated in time, the equation 3.29 can be rewritten as:

$$\boldsymbol{E_T = \sum_{l=1}^{k} \left(Z_{TT(l-1)} I_{0(n)} - Z_{TC(l)} Z_{CC(l)}^{-1} Z_{CT(l)} I_{0(n+m)} \right)} \qquad \mathbf{3.30}$$

Distributing tasks

This approach can be used for distributing tasks in many parts as needed for balancing the computational burden. This separation generates a model for parallel computing considering as basic unit the primitive network. This means that the number of solvers required for finding the solution of the system will depend on the number of primitive networks.

If there exists a number of independent solves equal to the number of existing primitive networks, the connections matrix Z_{CC} can be separated in the same number of parts. Additionally, for reducing the amount of information sent between these independent solvers, the steps 2 and 6 of the algorithm presented in TABLE 3-1 must be modified.

Instead of considering the load's information about the primitive network when this has no excitation, the solution of these primitives must consider the previous solution (voltage) for finding the PCD currents. This approach will reduce the number of iterations required for solving the primitive network and the interconnected system. During the first iteration, the value of the voltage for non-excited primitive networks will be the nominal value of the elements (V_0). With these values it will be possible to update the injected current vector including the contribution of these type of networks as follows:

$$\boldsymbol{\frac{Pow_{nom}}{V_{calc}} - \frac{Pow_{nom}}{(V_{nom})^2} V_{calc}^{\ *} = \Delta I} \qquad \mathbf{3.31}$$

As a result, when including the considerations mentioned above, the algorithm presented in TABLE 3-1 can be reformulated as shown in TABLE 3-2.

TABLE 3-2
ALGORITHM FOR APPLYING MULTILEVEL DIAKOPTICS

The compilation stage	
Step	Description
1	Compile the interconnected power system. Enumerate the switches and link branches for opening and to create the tree's matrix. With this matrix, then, obtain the connections matrix Z_{CC} and the mutual impedance matrixes Z_{CT} and Z_{TC}. Separate the connection matrix and the mutual impedance matrixes in many parts as primitive networks exist in the tree's matrix. Finally, send this information to the parallel solvers of the concurrent computing system.

The simulation stage	
Step	Description
2	Solve each primitive network using $I_{0(n)}$ for obtaining an updated $I_{0(n+m)}$ and E_0. If the primitive network has no injected currents assigned in vector $I_{0(n)}$ and the iteration number is equal to 0, use the nominal values of the loads for calculating the contribution to $I_{0(n+m)}$. Otherwise, if there are not assigned currents to the primitive network and the iteration number is different from 0, use the previously calculated voltage values $V_{0(n-1)}$ for this primitive network.
3	Send the new $I_{0(n+m)}$ to the independent solvers for calculating the complementary voltages. Define an empty complex array E_1 for storing the partial results. The size of this array is the same than E_0.
4	Follow steps 2 to 5 from algorithm in TABLE 3-1. Add the calculated partial solution to the vector E_1.
5	Verify that all the independent solvers have finished.
6	Calculate the total solution. $E_T = E_0 + E_1$

This algorithm considers the existence of independent solvers working in parallel dividing the solution process in two stages:

- The solution of the primitive networks for obtaining the vector E_0 and the updating process of the vector $I_{0(n+m)}$.
- The calculation of the complementary voltages for building the vector E_1.

These stages will be the base for proposing the computational model of this approach.

Simulation of Hybrid power systems

For solving several frequencies at once, this Diakoptics approach can be used by considering the independence of the primitive networks as a determinant factor. Consider two systems working at different frequency coupled by a digital control device such as VSC (Dasgupta & Agnihotri, 2009). There are three different possibilities for describing the interaction between these systems:

- System A delivers power to system B
- System B delivers power to system A
- Both systems are electrically isolated.

Considering the situations exposed above means that one of the two systems will see the other as a load; on the contrary, at the other side the alien system will be seen as a current source. The dynamic of the load/current source can be handled locally according to the desired control algorithm.

Then, if both systems are decomposed using Diakoptics their representation will be as shown in Figure 3-10. As can be seen in this figure, two power networks working in different base frequency can be represented independently using Diakoptics; however, both systems can be represented into a single structure by considering the independence of the tree's matrix and their complementary structures Z_{CC}, Z_{CT} and Z_{TC}.

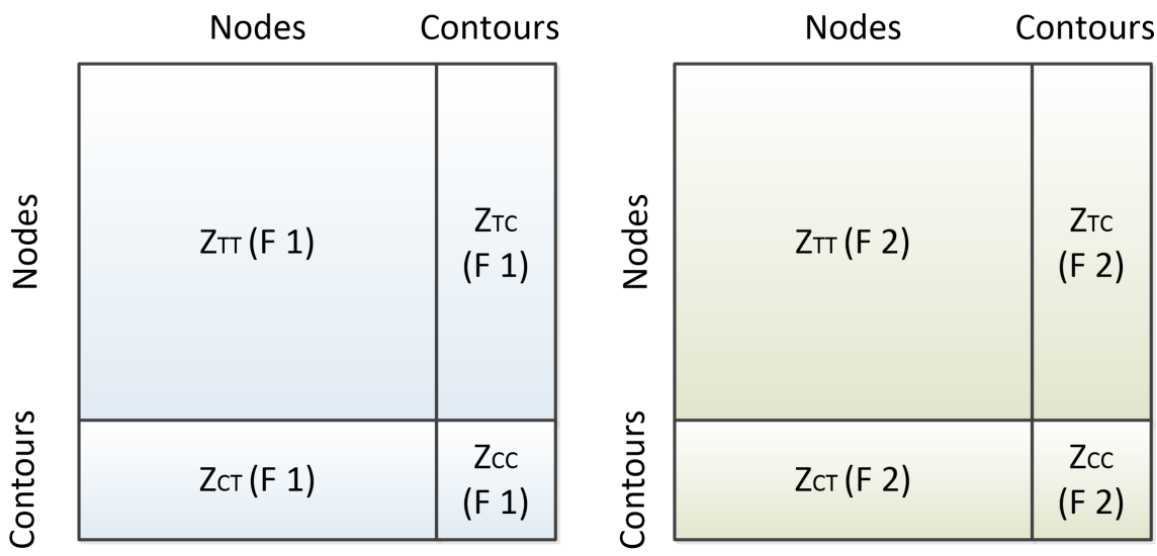


Figure 3-10. Two power systems represented using their interconnected equivalents in Diakoptics. F1 and F2 means that both systems are operating at different base frequencies.

The combination of these two systems into a single structure is shown in Figure 3-11. Depending on the control action simulated, it must be added a current source/load to each subsystem for interfacing them according to the possibilities mentioned above.

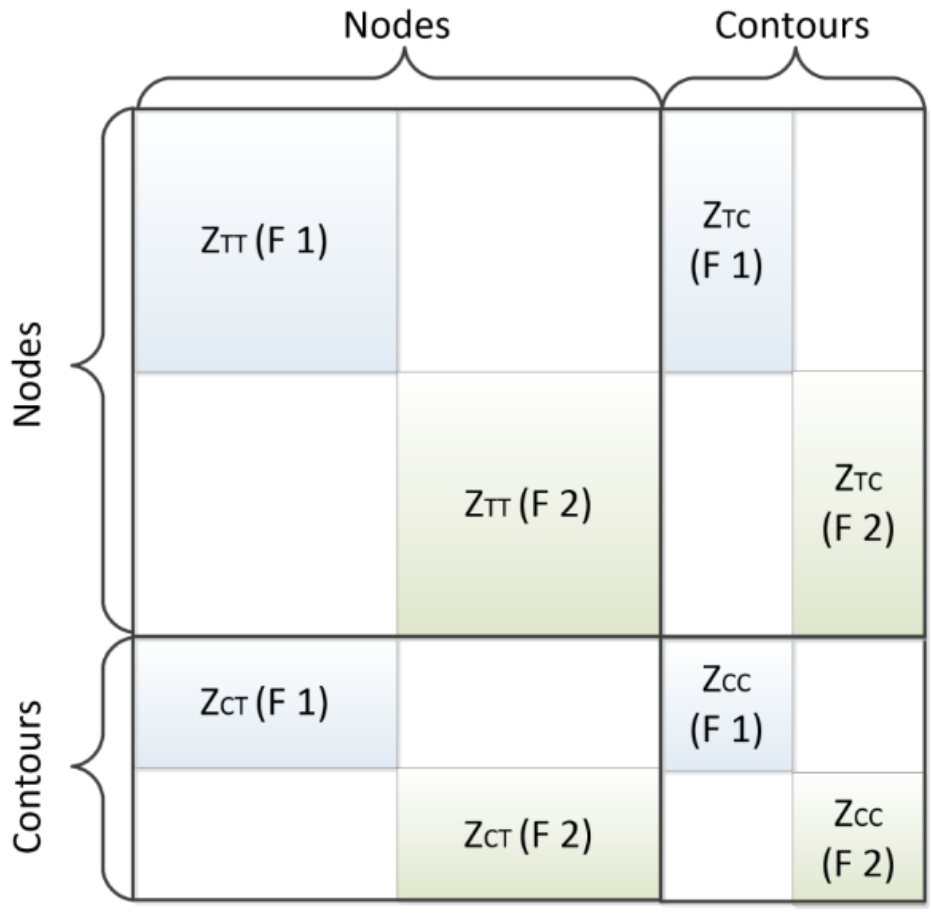


Figure 3-11. Two power systems working at different base frequencies combined into a single Diakoptics structure

This coordination process can be performed by a control algorithm, which will control the value of the load at the side of the power delivering system, and the value of the current source at the side of the power consuming system. This relationship is shown in Figure 3-12.

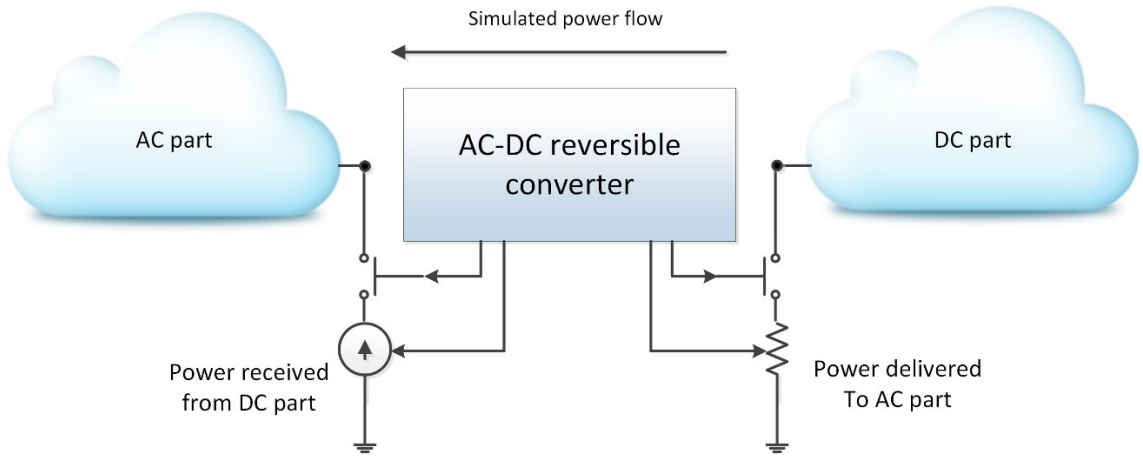


Figure 3-12. A control model for interconnecting two power systems working in different base frequency.

As can be seen in Figure 3-12, a connection between an AC system and a DC microgrid is being controlled by using an AC-DC reversible converter. The DC microgrid is delivering power to the AC system, which is represented as a DC load on the DC side. On the other hand, the power delivered by the DC microgrid to the AC system is modeled as a current source.

The control strategy can based on voltage for defining dynamically the values of the load and the current source. Additionally, when these systems are electrically isolated, the controller separates them by using switches.

It is expected that the control model of the AC-DC converter will be handled locally by one of the independent solvers proposed above.

The computational model of Diakoptics

This approach for Diakoptics proposes the existence of several independent solvers for solving the entire power system using a parallel computing system. This type of computation was implemented by Happ in the 80's when using an heterogeneous computing system (several computers) communicated using serial ports at 1200 bps (Happ, 1980).

However, modern technology allows to incorporate several cores in the same chip creating a homogeneous computing environment as discussed in chapter 1; this feature will help to improve the computing times, the improve the synchronism (the global clock is the same for all cores) and to communicate using high speed communication interfaces such as queues, UDP and TCP Ethernet connections (Agha & Panwar, 1992; Instruments, 2013; Montenegro, Ramos, & Bacha, 2014).

As a result, Diakoptics can be implemented using the existing computing technology within a single computer, which means that the asynchronous events, concurrency and parallelism of the proposed model can be handled for PC and RT purposes looking for different scopes in terms of performance for each case.

Creating primitive networks (Islands)

The first step in translating an interconnected network into the Diakoptics interconnected equivalent is to build the tree's matrix, which is composed by several primitive networks. The aim is to consider all the possible topologies (or at least those of interest) when separating the interconnected network. For this reason, the first action over the interconnected model will be open all switches (reclosers, fuses, etc.) present in the system.

Due to the sparse nature of the Y_{BUS} matrix (Y matrix) and because normally this matrix is built for an algorithm following the user commands, this matrix must be analyzed and reordered for making it adequate for being considered as a tree's matrix. In this analysis, the primitive network will be called *islands*, which are the subject of study for this part of the work.

The detection of islands within sparse matrixes has been a topic of interest for several years. Due to the correct location of these, to propose a convenient order of the matrix elements is possible. This order is highly important for solving linear equation sets by using direct methods (Gupta & George, 2010).

These methods are based in graph theory for detecting strongly connected elements within the matrix, which are reordered for producing less fill-in and computational work for solving the system (Amestoy, Davis, & Duff, 1996). The problem to solve is as follows:

$$\mathbf{Ax} = \mathbf{b} \qquad 3.32$$

The aim is to make A suitable for finding b using factorization methods such as Cholesky, LU, among others (Ashcraft, 1995); avoiding matrix inversion for computing direct solutions of large sparse systems. These techniques have been extended for covering unsymmetrical systems, which results in a problem NP-Complete (Yannakakis, 1981); demanding the use of heuristics for finding the adequate permutation.

For finding the reordering required the methods mentioned above only the structure of the matrix is used; this way, the computational burden is reduced using simpler representations such as matrixes branch to node.

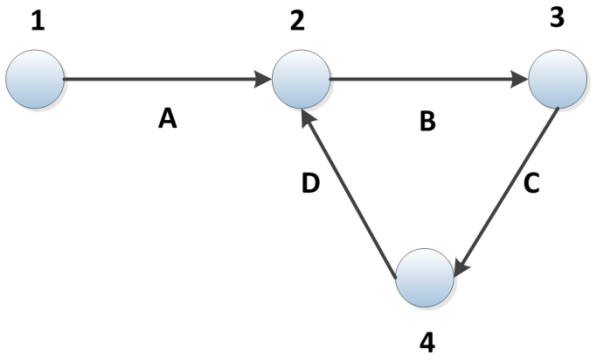
The first version of these algorithms is registered in literature in the 60's. This algorithm is a symmetric equivalent of the Markowitz method (Markowitz, 1957),and is called optimal ordered triangular factorization (Tinney & Walker, 1967). Then, this method was improved by adding graph theoretical models (Björck, 1996) and are called the minimum degree algorithms.

Since then, there are several works around this topic looking for improving the computational burden and memory use; but keeping the original idea of selecting a group of nodes of minimum degree.

However, this work deals with the problem of island location when the DS topology changes (Maheshwarapu, 1998). This is a different focus to the ones presented by the algorithms mentioned above, which are focused into searching the block triangular form (BTF) for solving the sparse linear system.

For topological analysis, power systems can be represented with a directed graph (Balakrishnan & Ranganathan, 2000; Diestel, 2006). These graphs describe a set of nodes connected by edges, which have associated a direction, which are represented as an incidence matrix where the cells are occupied by 0 or 1 depending on if a branch is connected to a node; the sign of the digit defines its direction.

An example of a network and its incidence matrix is presented in Figure 3-13.



	1	2	3	4
A	1	-1		
B		1	-1	
C			1	-1
D		-1		1

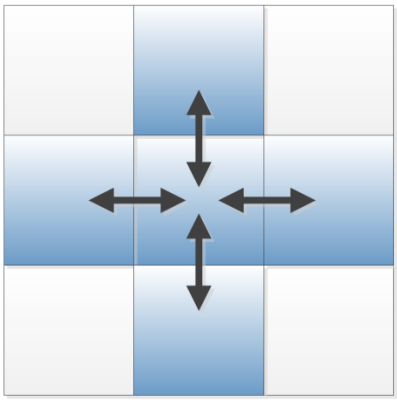
Figure 3-13. The network represented as a graph and its incidence matrix

However, due to the radial nature of DS the incidence matrix is non-symmetrical. This representation can be translated into another called the adjacency matrix using the following expression:

$$\mathbf{Ad} = \mathbf{I}^{\mathbf{T}}\mathbf{I} \qquad 3.33$$

Where *Ad* is the adjacency matrix and *I* is the incidence matrix. The adjacency matrix is symmetrical and describes how the nodes are interconnected. The diagonal of this matrix corresponds to the number of nodes linked to the diagonal node. The other cells will have a value equal to -1 if they are connected to the node corresponding to the row/column.

The adjacent relationship between nodes indicates if they are strongly connected. If any cell of a certain row/column of the adjacency matrix is different from 0 means that the nodes are interconnected. This concept is illustrated in Figure 3-14.



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

Figure 3-14. The set of nodes strongly connected and their adjacent matrix.

So, by inspecting each column it is possible to determine if a certain set of nodes are connected to a reference. This procedure is iterative and can be performed using columns or rows due to the matrix symmetry.

The proposed strategy consists into sweep each column searching for interconnected cells. Simultaneously, an index value is assigned for representing each cell interconnected in a matrix full with the same size of the adjacent matrix.

The index value will increment in 1 when the search for interconnected elements is finished for the selected column. As a result, a matrix filled of indexes describing the islands will be obtained. This matrix is called the islands matrix.

The proposed algorithm for finding islands is shown in Figure 3-15. In this algorithm variables *X*, *Y* and *Index* will deliver important information. Variables *X* and *Y* contain the coordinates of the cell evaluated. Once the algorithm finishes the variable Index will contain the number of Islands found.

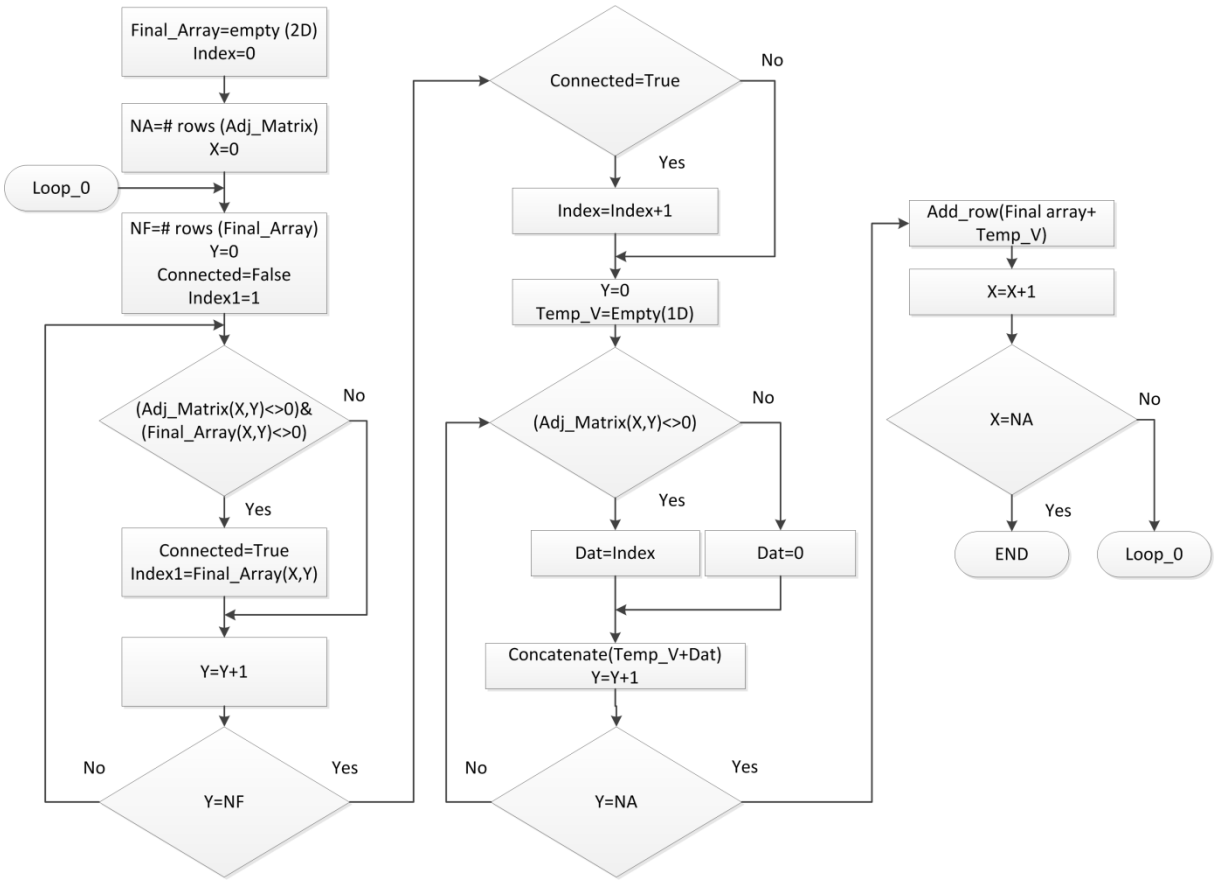


Figure 3-15. Proposed iterative algorithm for detecting and reorganize islands within sparse matrixes

Moreover, the diagonal of the islands matrix (*IM*) describes the location of the island's components. By extracting this vector a permutation can be proposed for grouping each island cells. Then, this permutation can be applied to the Y_{BUS} matrix of the DS for identifying clearly each island.

The permuted diagonal of the IM can be used for locating each island; this way, they can be extracted for independent analysis or for being used in multithread algorithms. To evaluate the performance of the proposed algorithm, consider the radial system shown in Figure 3-16. This system includes a distributed generator, three residential loads and two industrial loads. Because there are two Reclosers installed on the DS, there are four possible configurations for the system. These configurations are going to be explored for evaluating the performance of the algorithm.

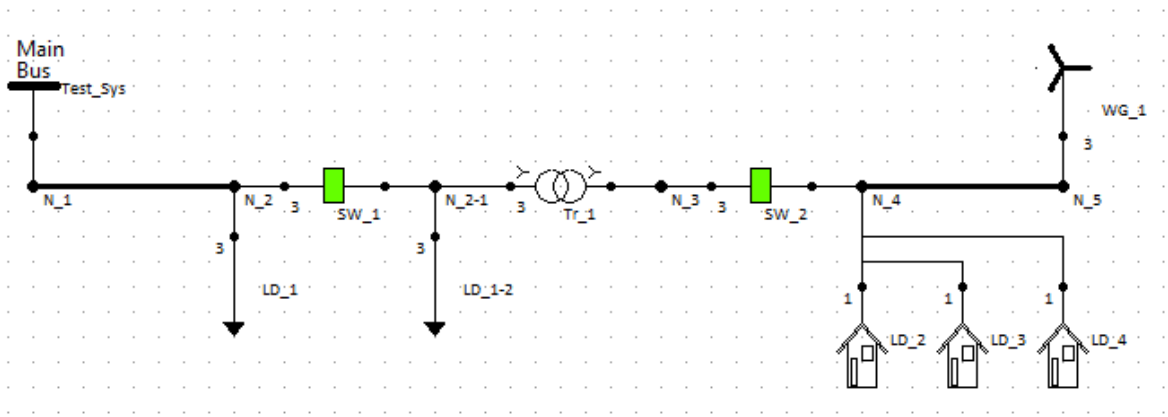


Figure 3-16. The proposed system to evaluate the performance of the proposed algorithm

This system is implemented using DSSim-RT, which provides the incidence matrix of the proposed DS. The basic version of OpenDSS does not provides this data; but using DSSim-RT/PC, the user can access this data by using the command 0x1D through the TCP server included (Montenegro, 2013).

The first configuration corresponds to the interconnected network, which means that both Reclosers are closed. The IM obtained from this network after applying the algorithm is shown in Figure 3-17. As can be seen, the order of the nodes does not correspond to the order in Figure 3-16 from left to right. However, because the system is interconnected the permutation suggested is equal to the actual distribution.

	n1	n2	n4	n3	n5	n2-1
n1	1	1	0	0	0	0
n2	1	1	0	0	0	1
n4	0	0	1	1	0	1
n3	0	0	1	1	1	0
n5	0	0	0	1	1	0
n2-1	0	1	1	0	0	1

Figure 3-17. IM obtained for the study case 1

As a consequence, the Y_{BUS} matrix remains the same as seen in Figure 3-18. In this figure, the colored cells correspond to the non-zero values and the rest are zero.

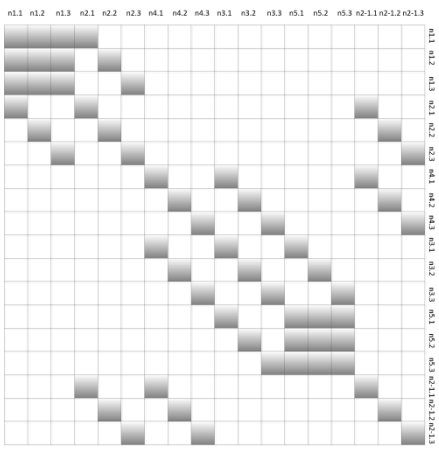


Figure 3-18. Y_{BUS} matrix obtained for the case of study 1

The next configuration happens when the Recloser SW_1 (left) trips, thus separating the system in two islands. Then after executing the algorithm with this configuration, the IM obtained is as shown in Figure 3-19. The suggested permutation remains equal the original organization of the Y_{BUS} matrix, since both islands have their components well grouped. The resulting Y_{BUS} Matrix is shown in Figure 3-20.

	n1	n2	n4	n3	n5	n2-1
n1	1	1	0	0	0	0
n2	1	1	0	0	0	0
n4	0	0	2	2	0	2
n3	0	0	2	2	2	0
n5	0	0	0	2	2	0
n2-1	0	0	2	0	0	2

Figure 3-19. IM obtained for the study case 2

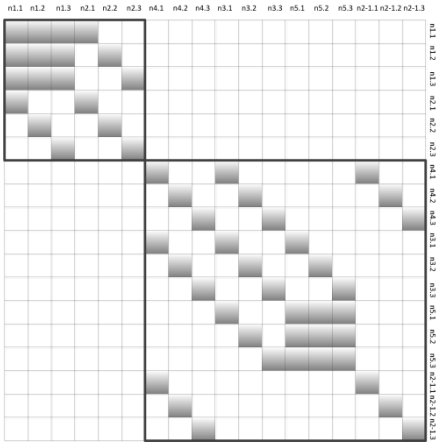


Figure 3-20. Y_{BUS} matrix obtained for the case of study 2

However, in the third configuration the Recloser SW_2 (right) trips and SW_1 closes. As a result, the system is separated in two islands again, but the components of the islands are separated as shown in Figure 3-21a. The node n2-1 is part of the island 1, but its location within the matrix is far from the other components of the island. As consequence the algorithm proposes the permutation: 0, 1, 2, 5, 3 and 4. Then, row and column 5 must be relocated into row and column 3. As a result, the IM presented in Figure 3-21a is reconfigured as shown in Figure 3-21b.

	n1	n2	n3	n4	n5	n2-1
n1	1	1	0	0	0	0
n2	1	1	0	0	0	1
n3	0	0	1	0	0	1
n4	0	0	0	2	2	0
n5	0	0	0	2	2	0
n2-1	0	1	1	0	0	1

(a)

	n1	n2	n3	n2-1	n4	n5
n1	1	1	0	0	0	0
n2	1	1	0	1	0	0
n3	0	0	1	1	0	0
n2-1	0	1	1	1	0	0
n4	0	0	0	0	2	2
n5	0	0	0	0	2	2

(b)

Figure 3-21. IM obtained for the study case 3, at the left without permutation, at the right according to the order suggested by the ordering algorithm

As can be seen in this figure both islands are identified and grouped. Moreover, by checking the diagonal of the matrix IM after its reordering it is possible to locate the islands within the matrix. The final form of the Y_{BUS} matrix for case 3 is shown in Figure 3-22. According to this figure, island 1 goes from nodes n_1 to n_2-1 while island 2, goes from nodes n_4 to n_5.

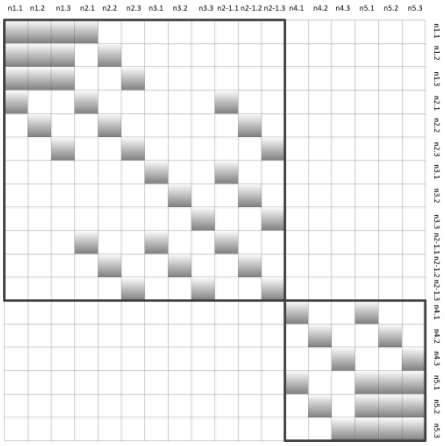


Figure 3-22. Y_{BUS} matrix obtained for the case of study 3

In the last configuration both reclosers are opened creating 3 islands, the resulting IM matrix is shown in Figure 3-23. Again, because the island components are separated the algorithm suggests a permutation. This permutation is the same as in the case 3 due to the distribution of the matrix. The new Y_{BUS} matrix after the permutation is shown in Figure 3-24.

	n1	n2	n3	n4	n5	n2-1
n1	1	1	0	0	0	0
n2	1	1	0	0	0	0
n3	0	0	2	0	0	2
n4	0	0	0	3	3	0
n5	0	0	0	3	3	0
n2-1	0	0	2	0	0	2

Figure 3-23. IM obtained for the study case 4

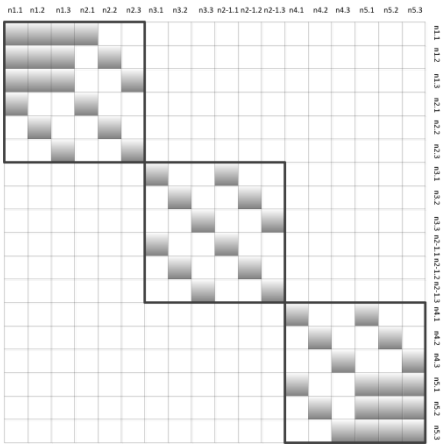


Figure 3-24. Y_{BUS} matrix obtained for the case of study 4

As a result, the YBUS matrix obtained in each case of study will be the tree's matrix for applying Diakoptics, and the number of islands will be the number of primitive networks.

Distributed computation

Diakoptics and its implementation using the algorithm proposed in TABLE 3-2 suggest the occurrence of asynchronous events within the parallel solver. The first stage is the compilation of the interconnected network, which will require a dedicated program for calculating and organizing the different components of the Diakoptics structure. This stage of the computing model cannot be part of the solution stage, which suggests that this stage will make part of a higher level program.

The second stage is the solution (simulation) stage, which is expected to be dedicated for guaranteeing performance. However, in the proposed model several concurrent solvers could be solving primitive networks with different size, due to the topological location of the switches within the interconnected network; this situation suggests that the computation time of each solution will be different (multi-rate), thus requiring a synchronism coordinator for this part of the process.

As a result, it is expected that the simulation time step will be equivalent to the time required for solving the largest primitive network. The size of the largest primitive can be fixed for ensuring the length of the step size in RT architectures, which will depend also on the number of physical computing resources available.

But for solving the primitive networks and to find the complementary voltages of the total solution, the concurrent solvers require the data about primitive network and the partial components of Z_{CC}, Z_{CT} and Z_{TC}. These values are calculated in the compilation stage and there are two levels up from the solvers (compiler and synchronism coordinator). This means that an extra communication time is required for passing this information from the compiler to the solvers. However, if the compilation stage is closer to the solvers, this extra time can be reduced.

With this analysis about the computational model of Diakoptics, a three level hierarchical model is proposed for its execution. This model is presented in Figure 3-25Figure 3-1.

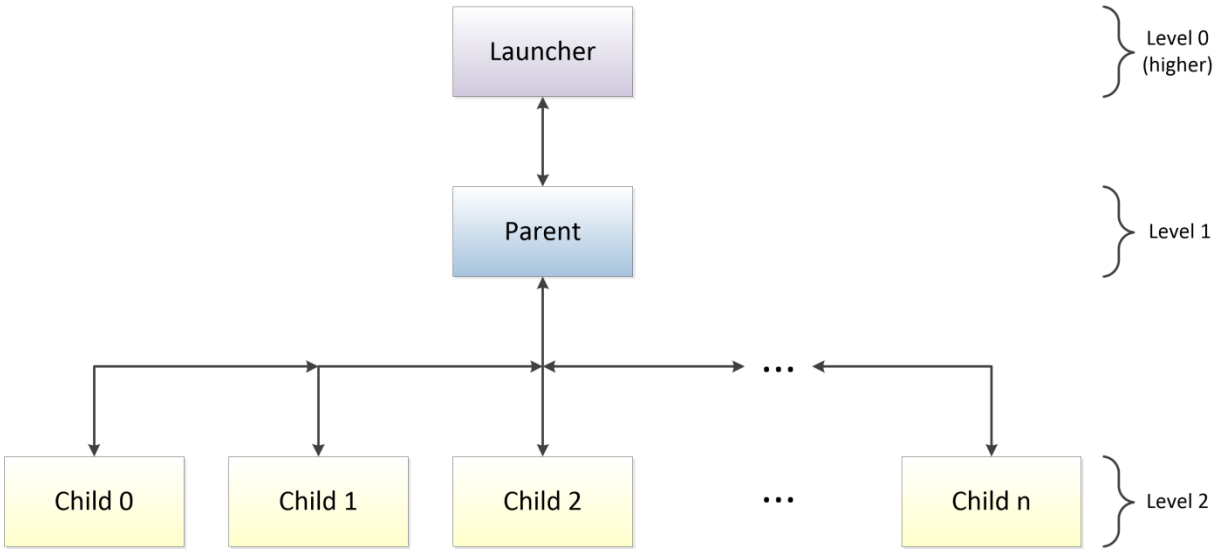


Figure 3-25. The hierarchical computation model proposed for implementing Diakoptics

In this model, the higher hierarchy level is the launcher, which will start the object named parent. The parent will compile the interconnected network and to determinate the number of children (solvers) required for solving the equivalent system, the priority and core of destination of each child according to the size of the primitive network associated, and will control the execution of the simulation in each

iteration. Finally, the children are objects that can be cloned many times as needed, they are responsible for solving the primitive networks, and to perform all the calculations commanded by the parent. The actions of the children will be always ruled by the parent.

Conclusions

Diakoptics is an advanced algorithm for representing interconnected into an orthogonal reference frame; this equivalent representation allows solving the interconnected system power flow in a distributed way, making of the selected solution method suitable for parallel processing using the existing computing hardware.

The Diakoptics equivalent of the interconnected network requires a less amount of cells for updating the state (open/close) of a link branch as shown in Figure 3-26.

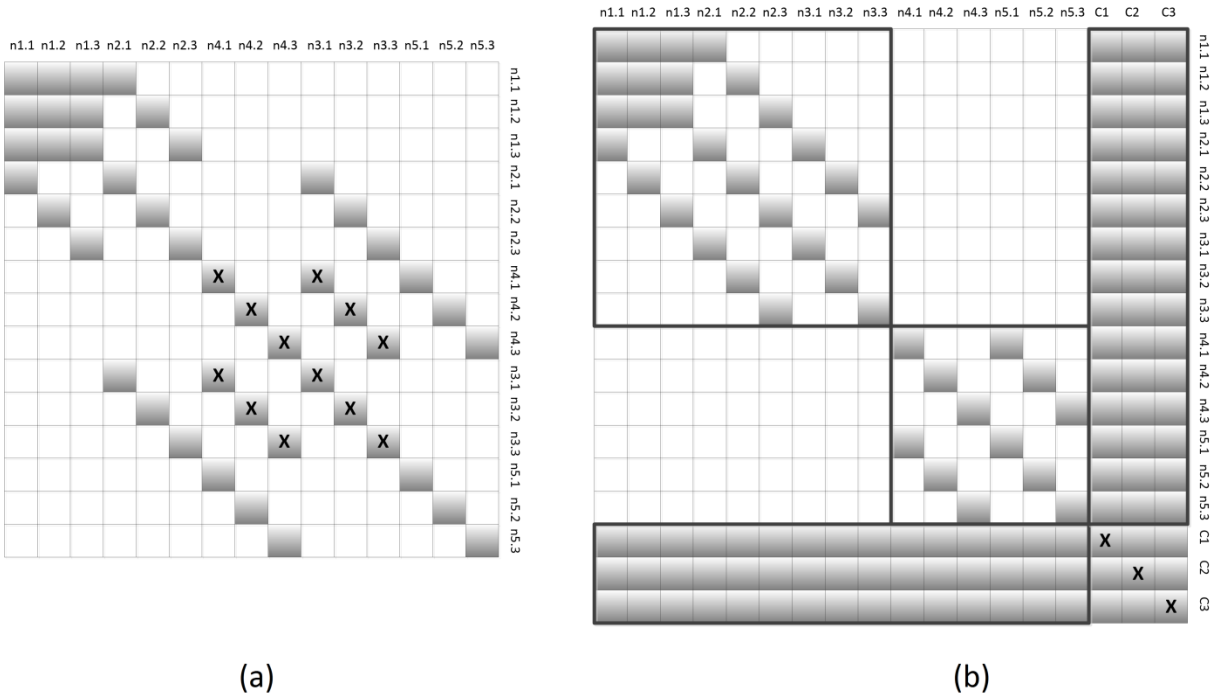


Figure 3-26. The interconnected network in Figure 3-2 (a) and its electrical equivalent using Diakoptics (b)

As can be seen in Figure 3-26, the colored cells are the cells different from 0 and the cells marked with *X*, are the cells required for representing the state of the Switch SW_1. In Figure 3-26 it is clear that there is a significant reduction of the number of cells required for changing the state of SW_1.

Given the modularity and scalability of the electrical equivalent built using Diakoptics, several non-conventional scenarios can be modeled for TSA, opening the door for simulating a wide spectrum of applications for smart grid studies.

The computational model proposed includes concurrency, asynchronous events and parallelism. For implementing this model according to these parameters, traditional sequential programming cannot be used. Is then when the actor model enters for building the method Diakoptics based on actors (A-Diakoptics).

Chapter 4. The Actor model for handling parallel and concurrent computing

CHAPTER 4. THE ACTOR MODEL FOR HANDLING PARALLEL AND CONCURRENT COMPUTING 66

CHAPTER 4 67

INTRODUCTION TO THE ACTOR MODEL 67
Homogeneous and Heterogeneous computing systems 68
Actors and Agents 70
BUILDING AN ACTOR'S SYSTEM 71
The actor structure 71
The actor messages 73
The management of the hardware resources 74
The standard PC architecture 74
The Real-Time Architecture 75
ACTOR'S ENVIRONMENT PROPOSED FOR IMPLEMENTING A-DIAKOPTICS 76
The parent actor architecture 78
The child actor architecture 80
The messages architecture 82
CONCLUSIONS 82

Chapter 4

Introduction to the actor model

The Actor model is a mathematical theory that treats "Actors" as the universal primitives of concurrent digital computation (Hewitt, 2012). This model has been used as a framework for a theoretical understanding of concurrency, and as the theoretical basis for several practical implementations of concurrent systems. This model proposes the computational representation theorem for establishing that an actor's system is a closed system as follows:

$$\boldsymbol{Denote_S \equiv \lim_{i\to\infty} Progression_S^i(\perp_S)} \qquad \textbf{4.1}$$

Where S is the closed system, $\perp S$ is the initial behavior and the behavior approximation function is $Progression_S$. This way, all the possible behaviors of S can be represented including the unbounded non deterministic behaviors (Copeland, 2008).

The actor model was proposed by Hewitt in the 70s (Hewitt, 1976). Later this model was used for RT computation and for integrate heterogeneous computing systems (Baker, 1978). Later on, Agha proposed the first framework (Agha & Panwar, 1992) for this model and since then it is incorporated to many programming languages.

This model gives to the parallel system inconsistency robustness, which is a frequent problem found when handling parallel tasks asynchronously. Inconsistency robustness is "information system performance in the face of continually pervasive inconsistencies" and is the result of asynchronous events accessing common resources (variables, data banks, equipment) within a parallel system. If these events are not handled properly the flow of information between the different concurrent processing unit (PUN) can be inconsistent, which can generate a chaotic response of the system especially if the handled system is multirate (Laddomada et al., 2011; Ni & Xiao, 2010), because each PUN could have different information about the same variable in the same time instant.

In this model everything is treated as an actor (like in object oriented programming OOP) (Hae-woo, Hanwoong, Hyunok, & Soonhoi, 2011); however, as difference from the OOP philosophy, which is executed sequentially, actors are inherently concurrent. Each actor is a computational entity that in response to a message it receives it can:

- Send a finite number of messages to other known actors
- Create a finite number of new actors
- Define its next functional state depending on the content of the received message

However, there is no sequential order specified for these actions, which means that they can be carried out in parallel. Actors can be described as Queue-Driven State Machines (QDSM) that exchanges data with other actors by using messages. The messages are sent by using dedicated queues or high speed communication interfaces, which are addressed to guarantee the high speed communication between these. Actors can be defined as objects within a virtual environment. As objects, they can be cloned automatically and executed independently in parallel (Baker, 1978).

The features of the actor model are (Hewitt, Meijer, & Szyperski, 2012):

- Inherent concurrency of computation within and among actors

- Dynamic creation of actors
- Inclusion of actor addresses in messages
- Interaction only through direct asynchronous message passing with no restriction on message arrival order.

These features require that for sending a message an actor must know if the destination actor exists, which is the key feature of the actor model for giving inconsistency robustness to a concurrent system. As a result, each actor has an address that can be implemented using a variety of methods (Hewitt, 2012):

- Direct physical attachment
- Memory or disk addresses
- Network addresses
- Email addresses

These addressing methods can be implemented using homogeneous/heterogeneous computing systems, which allows to consider the application of this model in a large number of applications such as:

- Electronic email and instant messaging (twitter for example, is an actor based system)
- Web Services can be modeled with Simple Object Access Protocol (SOAP) endpoints modeled as Actor addresses
- Objects with locks modeled as a serializer, which means that the object have an alternate procedure for granting the continuous arriving of messages.

For creating actors these must consider several parts; these include the number of states that define the behavior of the actor in time. The formal definition of an actor is as follows (Jie, Eker, Janneck, Xiaojun, & Lee, 2004):

$$\boldsymbol{A = (S, T, I, Q)} \qquad \textbf{4.2}$$

Where S is the set of the actor states, T is the set of actions of the actor, I is the set of initial/active states and Q is the set of quiescent states (normally I is the complement of Q).

Homogeneous and Heterogeneous computing systems

As mentioned in chapter 2, homogeneous computing systems refers to the interactions performed by several PUN working within the same environment (Engblom, 2008); this sentence taken to nowadays computing systems could be rewritten as: a group of PUN interacting within the same processor. Nonetheless, the homogeneous computing systems concept goes beyond the construction of many cores within a single chip; in fact, the homogeneous computing environments guarantee the same storage representation and the same results for a data operation in all PUNs. As a consequence, when transmitting this data to other PUNs the data is intact and its representation will be the same for all PUNs (Bhattacharyya et al., 2004).

On the other side, many of the simulation structures discussed in earlier chapters suggest that even if the simulation processing is taken in a homogeneous environment, the interfaces for exchanging information between the different parts of a Hybrid simulator as waveforms or data makes the system

heterogeneous. Moreover, in chapter 3 a hierarchical computing model was proposed for addressing the implementation of the Diakoptics algorithm, which demands independent behavior of each PUN for guaranteeing concurrency within the solver's structure, making of this a heterogeneous computing system (Ptolemaeus, 2013). For example, consider that the simulation is performed in a regular computer using a RT operative system (OS) and that the calculations are done using single precision floating point format (32 bit); then, this data is sent to FPGA for being exported as waveform and the FPGA will retrieve data acquired from the environment. Within the FPGA, the representation of the same data can be fixed point for saving local memory space, which means that the data must be transformed in the interface between the RT PC and the FPGA and that the calculations provided by each one will have small (even non-representative) differences.

For synthetizing and implementing these heterogeneous systems using actors it is necessary to use of a framework. There is a good number of actor frameworks implemented for different programming languages. These frameworks are inspired on early actor programming languages such as ABCL ("Armed Bear Common Lisp (ABCL)," 2014), Ambienttalk (Dedecker et al., 2006), Humus, SALSA (Varela & Agha, 2001), Scala (Haller & Odersky, 2006), Rebeca Modeling Language, among others. The Actor Model has inspired a good number of industrial applications. For example, Twitter has used actors for scalability. Also, Microsoft uses the Actor Model for developing its Asynchronous Agents Library. Some of the most representative Actor libraries and Frameworks as shown in TABLE 4-1TABLE 4-1; however, everyday there are new actor frameworks and libraries available on the internet for addressing issues related to parallelism and concurrency in large-scale computing systems.

TABLE 4-1
ACTOR FRAMEWORKS AND LIBRARIES

Name	Last release	Programming language
Actor (https://github.com/edescourtis/actor)	05-30-2013	Java
JActor (http://jactorconsulting.com/product/jactor/)	01-22-2013	Java
Quasar (https://github.com/puniverse/quasar)	07-19-2013	Java
Pulsar (http://quantmind.github.io/pulsar/)	06-30-2013	Python
Theron (http://www.theronlibrary.com/)	08-20-2012	C++
Actor-CPP (http://code.google.com/p/actor-cpp/)	03-10-2012	C++
Libprocess (https://github.com/libprocess/libprocess)	07-13-2012	C++
QP frameworks for real-time embedded systems (http://www.state-machine.com/qp/)	07-09-2012	C and C++
ActorKit (https://github.com/stevedekorte/ActorKit)	09-14-2011	Objective-C
LabVIEW Actor Framework (http://ni.com/actorframework)	06-07-2014	NI LabVIEW
NAct (http://code.google.com/p/n-act/)	02-18-2012	.NET
Actor Framework (http://actorfx.codeplex.com)	02-09-2013	.NET
Ikaria (https://github.com/frankshearar/ikaria)	07-25-2012	Delphi

Actors and Agents

In the latest decades for handling distributed systems the agent's model has been widely used (Akbari, Setayeshmehr, Borsi, Gockenbach, & Fofana, 2010; Bevrani, Daneshfar, & Hiyama, 2012; Hengxuan, Haishun, Jinyu, Shijie, & Haibo, 2012; Muller, Hager, & Rehtanz, 2014). However, agents can be built as a particular application case of actors, which ideas have found application in multiagent systems (Muller et al., 2014).

The features of agents are:

- They are continuously active, making observation on the computing environment.
- The communication with agents is by modifying something that the agent is observing, with this method of communication is not necessary to know that the agent exists.
- In certain conditions, high level agents can generate new agents or update local state
- The information needed by agents to perform their observations is provided by the runtime environment
- All the information is processed at once; agents support broadcasting and multicast models of communication (Teofili, Di Mascolo, Bianchi, Salsano, & Zugenmaier, 2008)
- It is necessary to establish a control protocol for avoiding feedback systems when working with agents

These features describe agents as Remote Terminal Units (RTUs) (Khotimah, Krisnandi, & Sugiarto, 2011; Wan Jusoh, Mat Hanafiah, Raman, & Ghani, 2013), which are commanded by a higher level application for making decisions while the process evolves in time. Moreover, the applications where agents need to make decisions locally have evolved to intelligent agents (Chun-Lien, Cong-Kai, Tso-Chu, & Ching-Jin, 2014; Hewitt & Inman, 1991; Nasri, Ginn, & Moallem, 2014), which are a particular application of actors for giving intelligence to agents and increase the data security.

Security is an important issue to evaluate because while actors require to know if the destination of the message exists, agents can be added or removed from the system because the information they provide does not require any local processing and/or authentication procedure. As a consequence, when additional intelligence for avoiding chaos within the information system of agents is incorporated, these agents become actors and begin including security features to the messages sent to other agents or actors.

In fact, nowadays multiagent systems refer to the hierarchical combination of intelligent agents and agents. As mentioned above, intelligent agents are actors emulating agents for making management decisions and to command distributed agents within the agents frame. These implementations, reveal how actors and agents are compatible, but making of actors a higher level management unit capable of making control decisions locally, called in the agent's universe as *Intelligent Agents*.

For explaining the actor construction, the actor framework available on NI LabVIEW (Instruments, 2013) is used. This platform was selected because of the following features:

- The programming language is entirely graphical, making easier the documentation and the explanation of the actor structure
- The actor framework included is mature, the first version was released in 2010, making of this framework a stable platform for creating actor environments.

- LabVIEW Object Oriented Programming (LVOOP) allows to model actors as objects, including hierarchical structures and the tools for allocating processes in specific hardware cores and threads are included.

Building an actor's system

For building an actor's system it is necessary to consider the following features:

- The actor structure
- The message structure
- The hierarchy model
- The hardware utilization

These features will determinate the necessary routines and resources for ensuring the deterministic behavior of a particular actor system; however, depending on the available computing resources the behavior of these systems could drastically change, as in the case of implementations using the standard PC OS.

The actor structure

An actor can be structured by declaring two parts:

- The actor core
- The actor messages

The actor core is the program that defines all the possible behaviors of the actor, these behaviors will be called states considering the definition of actor in 4.2 as QDSM. The actor core of the NI LabVIEW Actor Framework is a user defined class that inherits all its properties and methods from the ActorFramework LabVIEW Class. The regular definition of the actor core using this framework is shown in Figure 4-1.

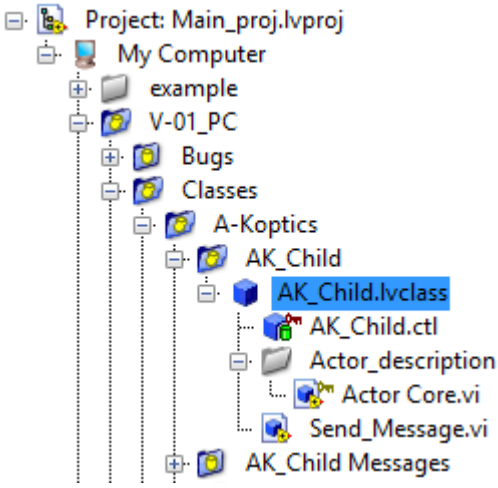


Figure 4-1. Project tree for creating an actor using NI LabVIEW

In this case, the actor's name is AK_Child.lvclass (the termination means that the program is a class). The control AK_Child.ctl is used for defining the variables that can be sent from an external actor to

this actor. Finally the *Actor Core* is a file .vi (VI means virtual instrument, which is the type of files used by LabVIEW to declare a program). The actor core is basically a program that includes the actor description as a state machine and a message handler. Both programs should run in parallel within the actor core, but at the same time, they should share information about the incoming and leaving messages. For building the state machine that will define the actor behaviors, these must be modeled as an action-reaction functional graph, which must consider how a several actions could drive the system from one state to another. An example of this model is shown in Figure 4-2.

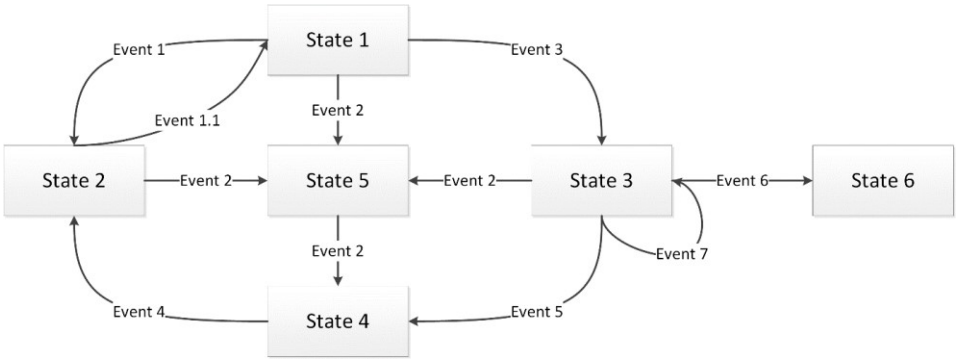


Figure 4-2. State Machine structure

As can be seen in Figure 4-2, for passing from one state to another it is necessary that certain event occurs while the process on the current state is executed. The passing from state A to B is unidirectional, which requires a different event for passing from state B to A, or a different logic route (passing through several states before arriving in the desired state).

For creating a state machine using NI LabVIEW the container structure will be a *while loop*; however, in many cases the *for loop* can be also used if the number of iterations of the process is known, which will force states to follow a desired sequential routine. Inside the iterative structure will be declared all the states using a *switch* structure (C, C++, Delphi, and LabVIEW, among others) for navigating within the different behaviors of the actor. These behaviors will be declared inside of each case of the *switch* structure and will define depending of the local results, the next state of the machine ("Tutorial: State Machines- National Instruments," 2015). An example of a functional actor core is shown in Figure 4-3.

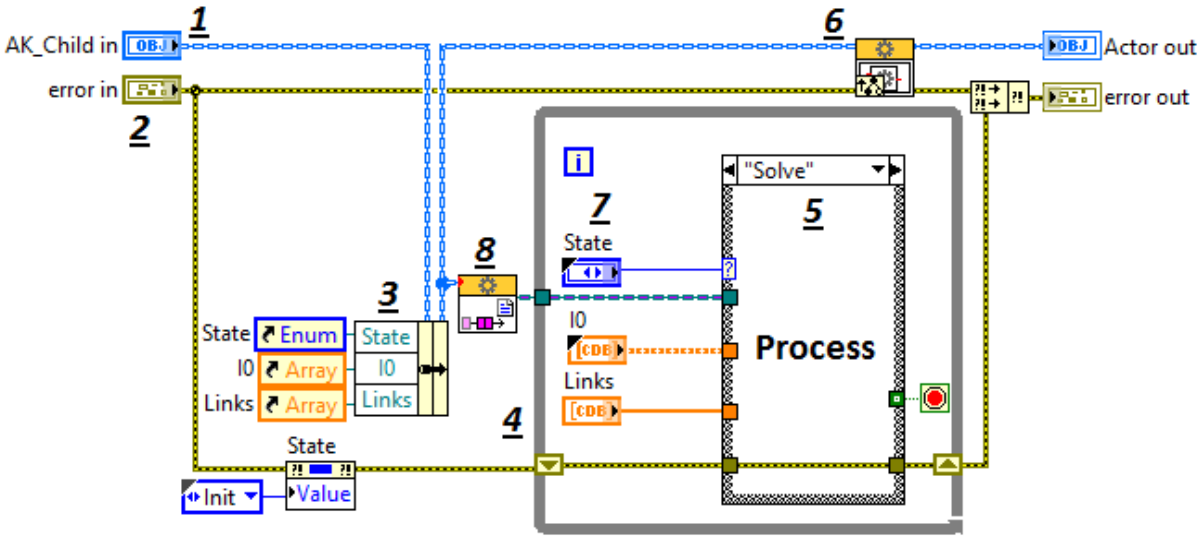


Figure 4-3. Functional actor core created using NI LabVIEW (running under regular Windows OS)

As can be seen in Figure 4-3, there are different parts of the actor core that must be identified. In this Figure, the different parts of the actor are signaled with an underlined number, these parts are:

1. The actor reference: This reference is used for localizing this actor from other external actors. This reference includes the address of the actor, the number of clones in case of being part of a set of clones and the location of the actor in memory.

2. The error in/out: These controls and indicators are used for directing errors when they occur for being handled locally or externally. These are highly important for controlling the behavior of an actor.

3. References to local controls and indicators: The local controls and indicators represent memory space designated by the actor to store data. The aim when including these references is that external actors can modify this content when sending messages to the local actor. Basically, when an external actor sends a message to the local actor, the content of the message is stored in these memory spaces.

4. The iterative structure: This structure is the container of the actor core; within this structure, the different states of the machine are declared. The only way of stopping the machine is to drive the machine to an end state, which will stop the execution of the *while loop*.

5. The current state of the actor.

6. The message handler: This structure is a subprogram (sub-VI) provided with the NI LabVIEW Actor Framework for handling the messages sent from other actors to the local actor.

7. The data memory space of the actor: This space is declared by placing controls and indicators, which are accessed by the local actor and modified locally or externally through the messages sent by other actors.

8. This Sub-VI provides the reference of the caller actor for sending messages to it when required.

There are also complementary parts that can be defined for operation purpose, such as Pre-launch Initialization routines, stopping routines, dynamic actor's addressing, among others. These additional parts can be used for declaring extra behaviors of the actor that cannot be handled from the actor core. Then, for sharing information with other actors, each actor should configure the message structure and how these messages will affect local variables when received from another actor.

The actor messages

The messages of the actor can be classified as follows:

- High coupling messages
- Low coupling messages
- Zero coupling messages

High coupling messages are messages whose structure can be used only for one purpose. This kind of structure is used when the variety of content for sending data from one actor to another is limited, which makes about this message structure very solid. However, the disadvantage of this type of message is that it can be used by the actor for sending the same type of data, forcing the programmer to create several types of messages for sending different type of information to another actor by reducing the re-use of the code and increasing the size of memory needed. Additionally, at every moment that the actor changes at a programming level, is possible that the messages of the actor should be adjusted to the new actor content.

Low coupling messages are more complex messages, where in addition to the content of the message there is a header for indicating the destination of the content. In this type of message the destination will modify the memory block specified in the header of the message; nevertheless, the content of the message can have only one format such as an array of integers, array of floating point numbers, array of complex, strings, among others. Although this structure is more flexible than the high coupling structure, several messages are needed for covering all the variables existing on the destination actor in case of being necessary.

On the other hand, zero coupling messages are messages composed of a header for indicating the type and destination of the content, and a non-format content. This non-format content can be capsulated using a variant variable (Eyal-Salman, Seriai, & Dony, 2013), which is a very common type of variable in our days programming software. This variant will be formatted once the message arrives to the destination according to the parameters sent in the header. This structure allows to send diverse contents using the same message structure. This is a very flexible structure because its content is independent of the actor's structure; however, it demands to identify and organize the information before send it to the destination actor.

The management of the hardware resources

Depending on the implementation type (Standard PC or RT), the hardware resources of the actor must be handled in a different way. In the case of PC the actor computing resources will be handled by the local OS due to the non-deterministic computing environment. A multitask OS such as Windows, must to distribute computing resources to all the software executing in the same time instant. As a result, the performance of the application could vary depending on the number of services, applications and nature of the applications running at certain moment.

On the other hand, RT applications runs over dedicated OS such as Windows embedded, Windows embedded compact, LabVIEW RT, Linux (not all implementations), among others. In these OS the number of applications and services running are the minimum and the computing environment must be handled from the executing application. This advantage allows to allocate each actor on specific cores and threads for guaranteeing the performance of the application.

The standard PC architecture

In the standard PC architecture the actor is generated as a clone of a certain object, or as the heir of a higher level actor (depending on the hierarchy model chosen). However, the hardware resources assigned to the actor are not specified.

As a result, the iterative structure used for containing the actor core is a normal *while loop* as shown in Figure 4-3. As shown in Figure 4-3, there is no place in the code where the execution core is specified and/or the thread were it is expected the execution of the actor. By placing the normal *while loop* the

actor will be executed in a random hardware location, this is, Windows (or the local multitask OS) will decide on which core and thread the routine will be executed. This control methodology will prevent unstable behavior when another application such as antivirus, media player, etc. is executed by the user simultaneously and could require more computing power for its execution. For example, for reproducing a video in HD the performance will be decided if the local computer has incorporated a set of Graphical Processing Units (GPU) embedded within a graphic acceleration card, or if the computer has integrated a video chip that uses memory from the local RAM memory.

The Real-Time Architecture

On the other hand, the RT architecture considers the assignment of hardware resources to each actor. This assignment is made by contemplating the number of available cores, which is the base for distributing actors within the cores according to the performance requirements.

For distributing hardware resources within the set of created actors, NI LabVIEW provides the *Timed Loop* structure, which allows to select the execution core for the routine and the priority for executing the routine (thread). An example of this structure is shown in Figure 4-4.

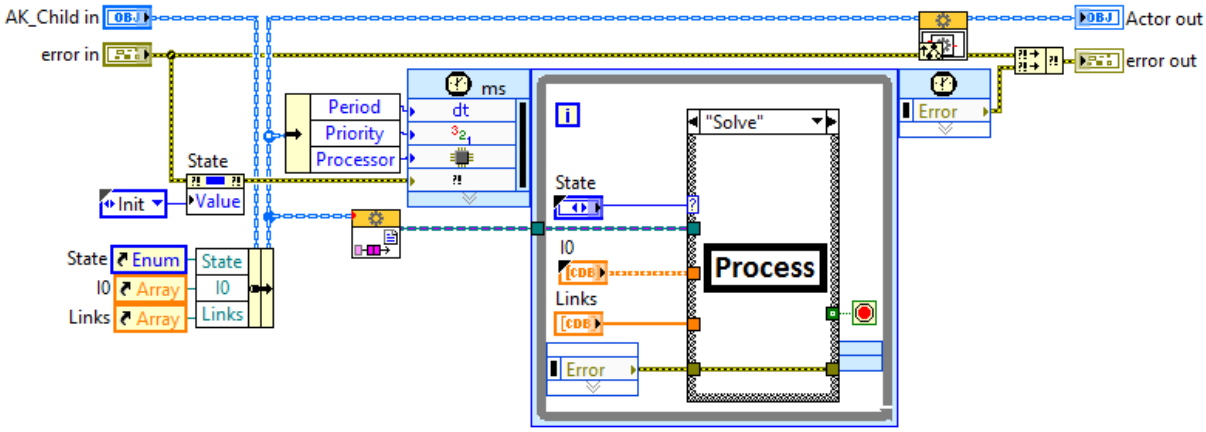


Figure 4-4. Functional actor implemented for RT execution (running under Windows embedded OS)

As can be seen in Figure 4-4, the actor is similar the actor shown in Figure 4-3; however, the containing structure is different and the actor variables as well. Within the set of variables of the actor there are 3 additional parameters called period, priority and processor as shown in Figure 4-5.

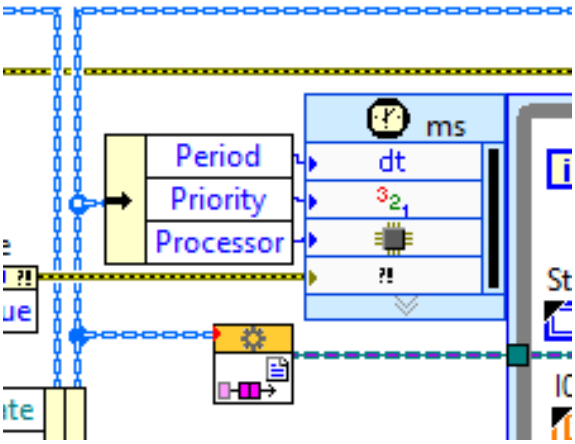


Figure 4-5. New parameters required for actor's RT execution (running under Windows embedded OS)

The period is a constant that works as a timeout for controlling the execution time of the actor's states. The priority is a number between 1 and 100 for specifying the percentage of use of the core by the actor. Finally, the processor defines the destination core on which the actor will be executed.

Depending on the number of the physical cores, the performance of the application will increase. This affirmation is based on the fact that if there are more cores available each actor would be assigned to a single core with 100% of priority; improving the performance due to the dedicated attention for processing the actor states.

Actor's environment proposed for implementing A-Diakoptics

For implementing Diakoptics based on actors the first step is to define the hierarchical relationship between actors according to the computational model proposed in chapter 3. In this model the hierarchy is composed by 3 levels as shown in Figure 4-6.

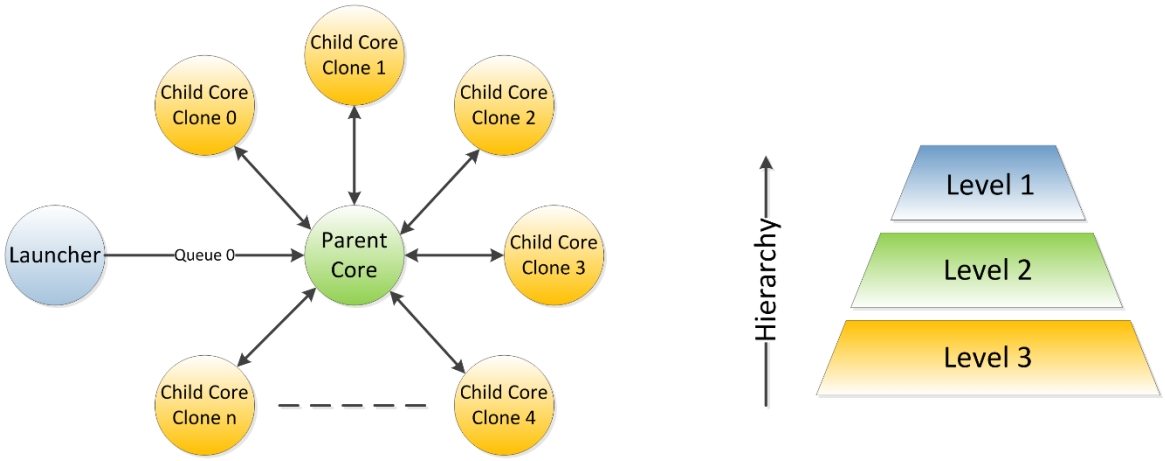


Figure 4-6. Actor framework proposed for implementing the A-Diakoptics methodology

These hierarchical levels are described as follows:

- The higher level is an application that launches the compilation actor and provides to it the information about the interconnected system. This actor coordinates the ending of the simulation process. This actor will be called *Launcher*.

- The second level is the compilation actor, which separates the interconnected system into its primitive and orthogonal equivalent networks. Additionally, calculates the number of solver actors to be generated according to the number of primitive networks created. This level Sends the solving information to each independent solver and handles the communication of partial results. This actor also coordinates the start and ending of the simulation. This actor will be called *Parent*.

- The third level is composed by the solver actors. These actors perform the solving routine of the primitive networks and calculate the complementary voltages according to the multilevel distribution. Its operation is governed by the second level. This actors will be called *Childs*.

Each level requires a description in terms of actors using the form presented in 4.2. Using this description the components of the proposed hierarchical framework will be represented as shown in Figure 4-7.

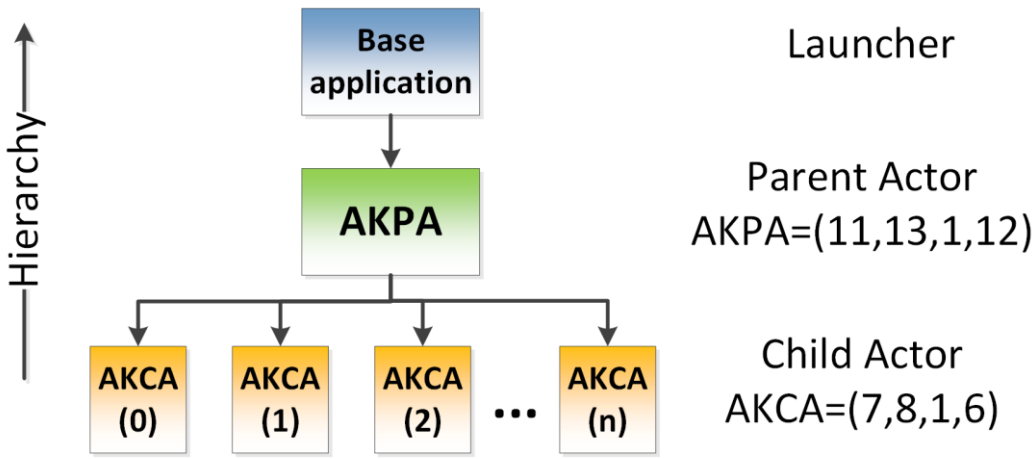


Figure 4-7. Hierarchical actor's model proposed for implementing A-Diakoptics

As can be seen in Figure 4-7, the launcher actor has no description in terms of actors; this is because function of the launcher application consist into provide the data of the interconnected network to the parent actor. This data is generated by another actor who compiles the system described graphically by the user, which is processed using OpenDSS (Dugan & McDermott, 2011). On the other hand, the number of interactions between the parent and its children is considerably high until the simulations finishes. For this reason the launcher is considered just as a starting point that provides data for being processed in further stages using actors. However, the launcher keeps being an actor with few behaviors defined. The expected interactions between actors are as shown in Figure 4-8.

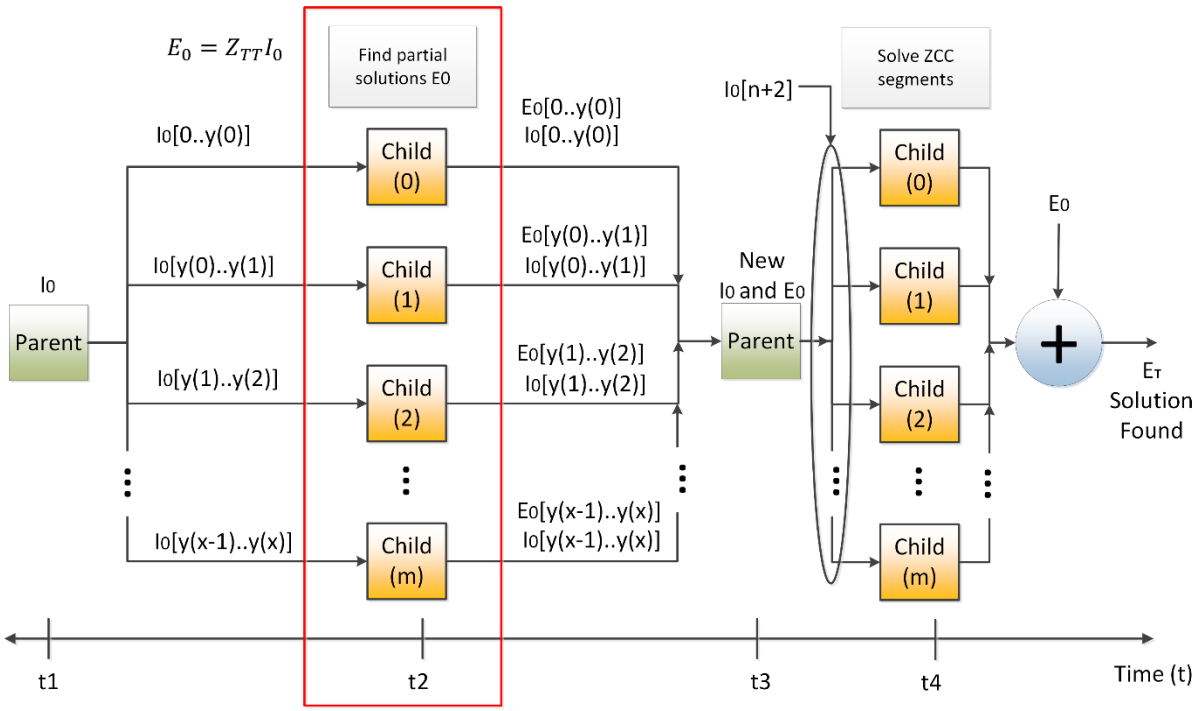


Figure 4-8. Expected interactions between actors

These interactions obey to the algorithm presented in chapter 3 for implementing multilevel Diakoptics using independent parallel solvers. As can be seen in Figure 4-8, the number of child actors is generated by the parent actor and the data for performing calculation is also provided for this actor.

The parent actor architecture

As mentioned above, the parent actor is the compilation routine for separating the interconnected network into primitive and orthogonal networks. The functions of this actor are:

- To separate the interconnected network
- To build the orthogonal networks and separate them using the multilevel approach of Diakoptics
- To determinate the number of child actors to be created according to the number of primitive networks
- To handle the messages sent between the child actors for receiving and providing partial results
- To distribute the available hardware resources between the existing child actors
- To control the simulation execution and to synchronize the concurrent computations
- To deliver information to the actors that needed, when and where they needed

As presented in chapter 3, the algorithm for separating the interconnected network into primitive networks is an iterative computing routine executed when the simulation is being initialized. This algorithm will provide the number of primitive networks formed; as a consequence, if the number of available computing cores is known the priority of each actor can be calculated as follows:

$$\boldsymbol{Priority}(\%) = \frac{\boldsymbol{Actors}}{\boldsymbol{Cores}} * \mathbf{100} \qquad \mathbf{4.3}$$

However, this expression is only valid if the size of each primitive network is the same for all cases, which is not the case due to the location of the switches within a DS do not correspond to a balanced distribution of nodes. Moreover, the computational time required for solving a sparse matrix can vary depending on the size of the matrix; this lack of balance could bring problems for ensuring an adequate equilibrium when distributing hardware resources into actors.

For solving these issues the following logic is proposed: Consider a single core machine, this machine will solve a power system that has 263 nodes distributed as follows:

$$\boldsymbol{PN_{S1}} = [\mathbf{3, 10, 50, 200}] \qquad \mathbf{4.4}$$

In equation 4.4 each number represents the size of each primitive network (these matrixes are square). If these networks will be solved using a single core, the priority of each network can be calculated as follows:

$$\boldsymbol{Priority}(\%) = \frac{\boldsymbol{PN_{S1}}}{\boldsymbol{INN}} * \mathbf{100} = \frac{[\mathbf{3,10,50,200}]}{\mathbf{263}} * \mathbf{100} = [\mathbf{1.3, 3.7, 19.0, 76.0}\,] \quad \mathbf{4.5}$$

Where *INN* is the number of nodes of the interconnected network. As can be seen in equation 4.5, the priority will depend on the size of the primitive network; nevertheless, this relationship could change if the number of physical cores increases. Let's assume that the number of cores is equal to 4, in this case, each actor will have assigned the 100% of priority for its execution no matter the size of their

primitive network, but in this case the system will assign important computing power to a task that can be performed using less resources.

The lack of balance between the problem and the hardware dedicated to attend it must be considered for guaranteeing an adequate distribution of resources; as a result, it is expected that the solution time on each iteration becomes deterministic no matter the size of the interconnected network, but considering the number of available computing hardware. This method will apply only to the RT architecture due to the lack of control present in the standard PC architecture (the control is performed by the multitask OS).

Experimentally, it has been determinate that to solve a square sparse matrix with a size of 2800^2 elements takes 2.5ms with a processor Intel® Core™ i7-4810MQ Processor 3.4 GHz (4 cores), 16 GB RAM, working with Windows Embedded OS. This will be the reference for distributing the hardware resources. The first step will be to group the primitive networks by size until filling the maximum capacity of the core. Then, within each core distribution the new priorities are assigned to each actor for its execution. In the example previously presented the total size of the matrix is 263^2, this means that the system can be solved using a single core within a multi-core computing architecture. Now consider a medium-scale system such as the EPRI circuit 7 ("Distribution Test Feeders," 2013). This system modeled in OpenDSS has 2452 nodes and 11 switches. After opening all the switches the interconnected network is separated into 13 primitive networks and the maximum matrix size will be set in 900^2 elements. The vector PN_S for the proposed system will be as follows:

$$\boldsymbol{PN_S = [9, 3, 256, 860, 20, 36, 46, 169, 236, 323, 133, 121, 240]} \qquad \mathbf{4.6}$$

This information is used for grouping the primitive networks until reach the maximum matrix size starting from the smaller primitive networks. As a result, the primitive networks are grouped in three groups, which reveals that for solving the system with the desired performance three cores are required. The groups formed are as follows:

- Group one (G1) = [9, 3, 20, 36, 46, 169, 133, 121, 240]
- Group two (G2) = [256, 236, 323]
- Group three (G3) = [860]

Because no group reach the maximum node capability proposed for the processor, the sum of the sizes of the primitive networks in the group will be used for calculating the priority of each actor. Then, for each group of actors the execution priority will be as follows:

- Priority G1(%) = [1.16, 0.39, 2.57, 4.63, 5.92, 21.75, 17.12, 15.57, 30.89]
- Priority G2(%) = [31.41, 28.96, 39.63]
- Priority G3(%) = [100]

This way, all the child actors are distributed considering the proposed hardware architecture for guaranteeing a balanced distribution of computing hardware resources. Some of the obtained actors will be executed with 100% of priority over a single core, while some others, will be executed with a different priority in another core for sharing the computing power of the core. The expression for distributing the hardware resources into a group of actors can be described as follows:

$$\boldsymbol{Priority_{Gn}}(\%) = \frac{\boldsymbol{PN_{SGn}}}{\sum \boldsymbol{PN_{SGn}}} * \mathbf{100} \qquad \mathbf{4.7}$$

Where $Priority_{Gn}(\%)$ is the vector of priorities of the group G in the core n, and PN_{SGn} are the sizes of the primitive networks of the group G for being executed in the core n.

This linear approach can be used for distributing the available hardware resources and to guarantee a balanced execution of the simulation; however, when the number of cores is not enough for guaranteeing the expected computation time, the option will be to overpass the maximum established until a restricted limit (120% or 150% of the maximum limit initially proposed), compromising the computing time such as in the standard PC architecture.

The structure of the state machine of the parent actor considering the definition given in Figure 4-7 is shown in Figure 4-9. The *Init* state of this actor is reached when launched by the *Launcher* actor.

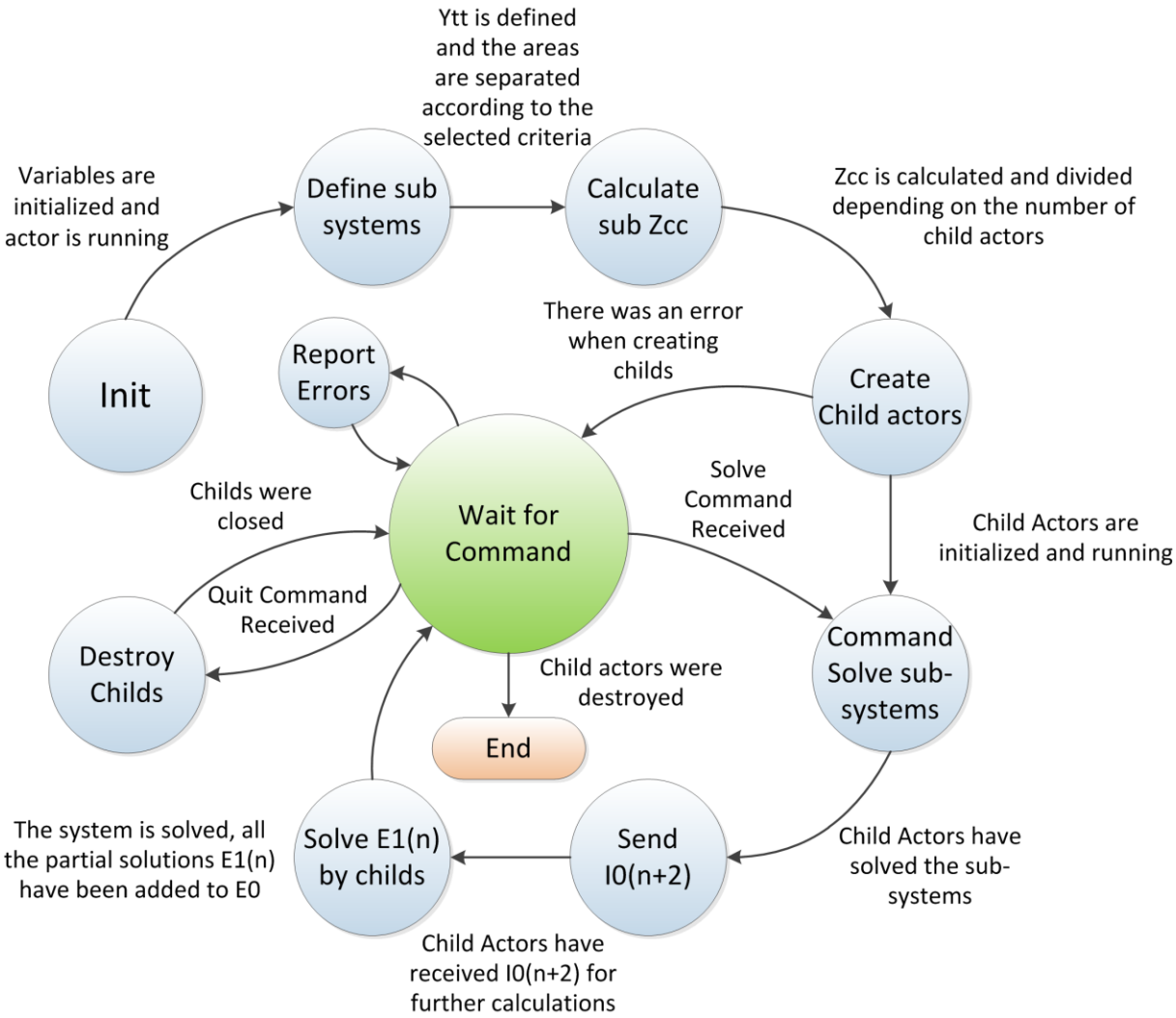


Figure 4-9. State machine proposed for the parent actor

The child actor architecture

On the other hand, the structure of the child actor is much simpler than the parent actor. The child actor is the PUN that perform the calculations needed for finding all the partial results of the Diakoptics representation of the interconnected system, provides these results to the destination actors for build the total solution of the equivalent interconnected system.

These activities demand the creation of local memory spaces for storing the data of the local Y_{BUS} matrix, the partial Z_{CC} matrix and the content of $I_{0(n+m)}$. These memory spaces are declared by creating controls for each content in the child actor structure.

The structure of the state machine of the parent actor considering the definition given in Figure 4-7 is shown in Figure 4-10.

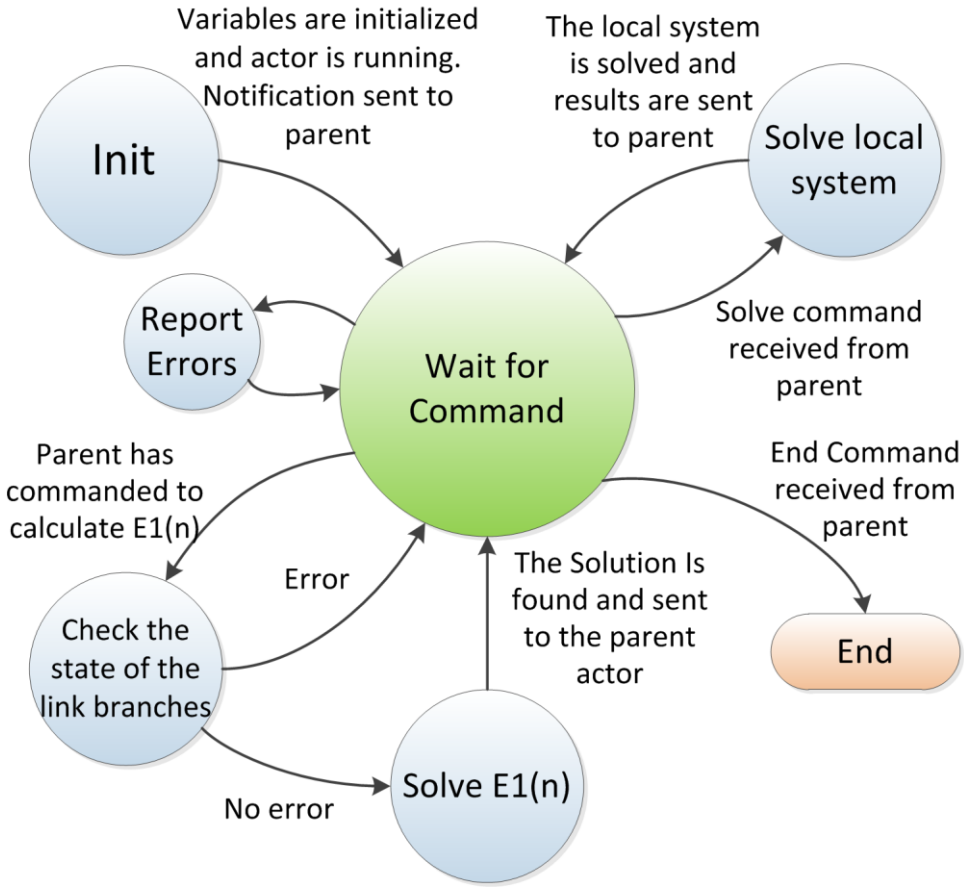


Figure 4-10. State machine proposed for the child actor

This actor is the direct heir of the actorframework LabVIEW class for preserving its independency; however, its operation can be commanded by remote using messages sent by the parent actor. Moreover, the creation of clones of this actor is governed by the parent actor, which starts and end all the clones (children) created.

The solver implemented in this actor to solve the local system (state solve local system) corresponds to the same routine proposed in OpenDSS. This routine uses the nodal formulation of the local system combining direct solution methods with iterative correction procedures to find the most accurate solution. This algorithm is explained in (Dugan & McDermott, 2011).

The direct solution method used by OpenDSS is the KLU factorization solver (Davis & Natarajan, 2010a, 2010b), this method is incorporated by using the DLL provided for OpenDSS. This DLL is accessed using direct connection through C++ prototypes assembly in NI LabVIEW ("How Do I Call a Dynamic Link Library (DLL) from LabVIEW?," 2007). With this connection method, the KLUSolver DLL will be accessed using an equivalent to early binding, which will guarantee the performance of the solver without adding an important overhead to the execution of the DLL.

The messages architecture

The messages for this framework are sent using queues, which allow to send information at 100Mbps for reducing the latency between computations. The model chosen for data exchange is the zero coupling model; the message is composed of an integer as a header that indicates the type of data and destination, and the message is a Variant type variable for containing any type of content.

To communicate with external actors (actors outside the same computer), the communication bus used is Ethernet and the protocol used is TCP/IP. In this case, the header specifies the type of data (1 Byte) and the dimensions of the message (in case of being a matrix, vector, scalar, etc.); the message is sent as a byte stream that will be formatted in the destination.

With this model the messages sent between actors must obey to the logic proposed in Figure 4-8. The set of messages sent by actors listed by sender and receiver within A-Diakoptics is presented in TABLE 4-2; these messages are sent asynchronously (the order depends on who has the result first) but always using queues, which guarantees that no-message will be lost and all will be attended by the destination.

TABLE 4-2
LIST OF MESSAGES SENT WITHIN THE A-DIAKOPTICS FRAMEWORK

Message	From	To
Launch parent actor	Launcher	Parent
Compile and reorganize system	Parent	Local
Calculate number of child actors	Parent	Local
Launch child actors	Parent	Childs
Initialize Child actors	Parent	Childs
Solve sub-systems	Childs	Parent
Send I0(n+2)	Parent	Childs
Calculate E1(n)	Childs	Parent
Deliver partial calculation of I0	Childs	Parent
Add partial calculation of E1 to E0	Childs	Parent
Close Childs	Parent	Childs
Close Parent	Launcher	Parent

Conclusions

The actor framework is an information model for implementing parallel and concurrent systems. This model considers the occurrence of asynchronous events within a parallel system, and proposes the

exchange of information between independent PUNs by sending messages; as a result, it is expected to provide inconsistency robustness to the parallel system and list all the possible behaviors of it.

With the actor model an implementation of the Diakoptics algorithm proposed in chapter 3; as a result, a computational implementation called Diakoptics based on actors (A-Diakoptics) is proposed. With this implementation the equivalent interconnected network created with Diakoptics can be handled using parallel, independent and concurrent solvers, coordinated using a hierarchical computing model of three levels.

Using the actor framework provided with NI LabVIEW two different architectures are proposed based on the expected performance of the application. In one of these architectures the hardware resources are managed by the local multitask OS (PC architecture); on the other hand, the available hardware resources are managed by the parent actor in order to guarantee a balanced processing environment, and an adequate distribution of the existing computing power (RT architecture).

Chapter 5. Applications of A-Diakoptics

CHAPTER 5. APPLICATIONS OF A-DIAKOPTICS **84**

CHAPTER 5 ..... **85**

APPLICATIONS OF A-DIAKOPTICS 85
THE SIMULATION OF DISTRIBUTION POWER SYSTEMS 85
Distribution System Simulation using Standard PC architectures *86*
The harmonics simulation mode *88*
The sequential-time TSA *90*
Performance of the PC application *91*
Distribution System Simulation using Real-Time architectures *100*
OTHER FUTURE APPLICATIONS 105
DS state estimation *105*
Advanced Distribution Automation *108*

GENERAL CONCLUSIONS **111**

REFERENCES **113**

Chapter 5

Applications of A-Diakoptics

As mentioned above, the scope of A-Diakoptics is to make traditional Transient Stability Analysis (TSA) methods suitable for multithread processing, which represents the main contribution of this work; however, this method can handle several issues within the smart grid design, monitoring and control.

The main application of A-Diakoptics is the simulation of small-scale, medium-scale and large–scale power systems looking for improving the performance of existing simulation platforms for offline and RT-HIL. This application considers the integration of several technologies within a heterogeneous computing system for calculating the power system values, to export and import digital and analog signals (PHIL), to communicate with external controllers and monitoring devices (CHIL), guaranteeing deterministic computing times in Real-Time (RT). However, this feature can be taken to other applications with the same characteristics, but at a different scale and outside the laboratory.

These other applications are management activities called Advanced Distribution Automation (ADA) (Northcote-Green & Wilson, 2006; Zavoda, 2008, 2010). These activities are dedicated to monitor and control distribution systems, including the different threats proposed in smart grid studies. In this chapter, A-Diakoptics is proposed for performing the distributed state estimation of distribution systems (M. Baran & McDermott, 2009; M. E. Baran, Jinxiang, & Kelley, 1996) and also, to simplify the performance of ADA activities such as feeder reconfiguration, fault detection and isolation, among others (Abdulhadi, Coffele, Dysko, Booth, & Burt, 2011; Arritt & Dugan, 2011; Augustine, Mishra, & Lakshminarasamma, 2015).

This chapter is dedicated to showing the different applications of A-Diakoptics starting with the simulation of small, medium and large-scale distribution systems using a standard PC and RT architectures. Then, the application within the ADA context will be presented making emphasis on distributed state estimation and mentioning the approach for covering extra applications.

The simulation of Distribution power systems

As mentioned along this document, the aim of A-Diakoptics is the evolution of traditional TSA methods to make them suitable for multi thread processing, which will improve the performance of these methods by taking advantage of nowadays computing architectures for standard PC and RT processing.

To show how A-Diakoptics can improve the performance of the existing TSA methods and tools, an open source simulator is chosen; this simulator is EPRI's OpenDSS and as mentioned in chapter 2, several features makes of OpenDSS an adequate candidate for testing A-Diakoptics:

1. The program is well coded. It was conceived in term of classes and objects, making easier the code reading and modification.
2. The COM interface is a good alternative for interfacing external software without compromise the simulator performance, which is possible by modifying the number of internal iterations provided by this software.
3. It includes a good number of models and control devices and there are several study cases documented.

4. It is not a power flow program but a harmonics program with capabilities for solving power flows, including simulation modes such as harmonics that cannot be found in traditional power flow solvers.
5. The solver of OpenDSS uses direct methods for performing power flow analysis.

The first approach for evaluating the performance improvement is the standard PC, which will be the closest approach to the standard operation of OpenDSS. This new software is called the Distribution System Simulator for standard PC architectures (DSSim-PC).

Distribution System Simulation using Standard PC architectures

DSSim-PC (Montenegro, 2013a) is the implementation of A-Diakoptics oriented to standard PC architectures; this application is the first and closer implementation of A-Diakoptics for testing the performance improvement of OpenDSS. Both applications will be tested using the same PC architecture, which includes the concurrent execution of other applications such as software services, antivirus, firewalls, among others.

However, DSSim-PC includes additional features to OpenDSS, these features are:

- Graphical User Interface (GUI).
- The creation of additional elements.
- The creation of additional sequential time simulation modes.
- Additional interfaces for data exchanging with external software.
- GUIs for extended configuration modes such as the edition of XY graphs, load profiles, TC Curves for protections, among others.

A considerable number of these features were implemented using NI LabVIEW, which is the software that supports this work; however, some other improvements were implemented using Delphi XE 7 to modify directly the source code of OpenDSS to not compromise the application performance, and to contribute to the development of this simulation tool, which is the aim of open source code. The proposed structure in terms of actors of DSSim-PC is presented in Figure 5-1.

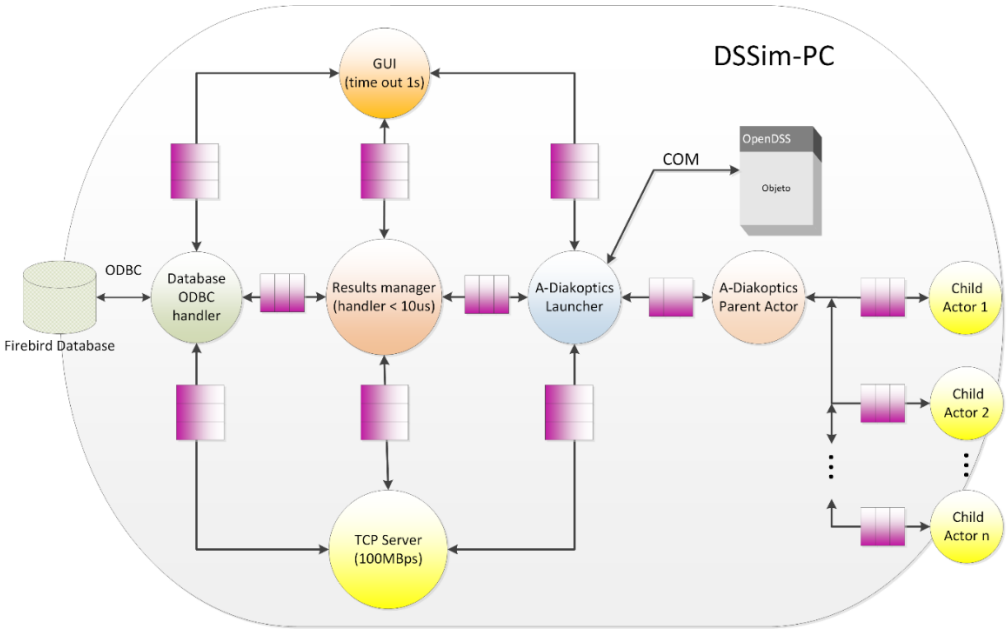


Figure 5-1. Actor framework proposed for DSSim-PC

In the DSSim-PC software there are 5 actors in the upper layer: The graphical interface (GUI), the A-Diakoptics launcher (AKL), the database handler (DBH), the TCP server (TCPS and finally, the results manager (RSM). The messages between actors are sent using queues and each message specifies the request and the complementary data.

Because other resident programs (like antivirus) are running at the same time in normal PCs, the hardware resources are handled by the local OS (low determinism). The GUI actor receives instructions to publish graphical data from other actors. This data can be an error message, the command to refresh or create a drawing, general information for presenting to the user and to publish information from a simulated meter. It interacts with the user using events and counts with its own communication manager to receive and send messages. If any actor requires delivering graphical information, it must to communicate with the GUI actor.

The AKL actor updates the values and states of the declared elements within the customized OpenDSSEngine (Dugan & McDermott, 2011). This actor is the highest level actor of the hierarchical structure proposed in chapter 3 for implementing the A-Diakoptics algorithm. As can be seen in Figure 5-1, this actor creates the parent actor and communicates directly with the OpenDSSEngine, which is the DLL implementation of OpenDSS provided by EPRI. The communication between AKL and the OpenDSSEngine is performed using a COM interface, while the communication with the other actors is performed through queues; however, due to the nature of the OpenDSSEngine it is considered as an actor communicated by using a different communication method.

Once the AKL actor creates the parent actor, it compiles the interconnected power Distribution System (DS) using the OpenDSSEngine. Then the OpenDSSEngine sends the Y_{BUS} matrix to the AKL actor, which distributes this information to the parent actor. Additionally, the OpenDSSEngine sends the information about the loads, distributed generators, reactors, capacitors, storage elements, controllers and all the elements connected to the network. This data is retransmitted to the parent actor for being distributed between the child actors created later by the parent actor. Then the simulation is taken between the parent actor and the child actors, the OpenDSSEngine is used for compiling the code generated by DSSim-PC and to provide back the description of the DS to DSSim-PC.

The DBH actor receives queries from other actors to be translated into SQL commands to exchange data with the database, which contains information about the graphical elements created in the GUI, the meters created, control devices, among others. This actor allows also to store large data and performs queries for detecting graphical elements within the GUI for moving, create, etc. It can handle up to 255 concurrent connections.

The TCPS actor receives connections from external applications and through an ASCII protocol; this actor exchanges information with other actors to solve the remote queries/commands. Through the TCPS external actors can connect to the simulation and even control it by using the defined set of commands, which is available with the documentation of DSSim-PC (Montenegro, 2013b). Finally, the RSM is the actor that coordinates the simulation progress according to the data exchanged with all the other actors. This actor handles the error generated by the other actors and determinates which will be the next action after one unexpected event occurs.

The structure of DSSim-PC is a multirate computing model because each actor works at different sample rates; however, because it is the local OS the manager that distributes the hardware resources, the performance of this application will depend on the number of applications running at the moment of its execution. All the actors proposed in this architecture are executed within the same machine

except for the external actors, which are external applications built in other programming languages (C++, Delphi, MATLAB, LabVIEW, etc.) connected to the TCPS actor to interact with the simulation.

The harmonics simulation mode

Depending on the simulation mode, the distribution of data within the child actors can change. For example, the harmonics simulation mode is a non-conventional simulation mode included with OpenDSS; however, this mode is not part of the sequential-time simulation modes and can be associated with the snap simulation modes. In this mode, the user can define the harmonic spectrum for each load and measure the resulting spectrum in different point of the network. The solution of the system is found by sequentially solving the system at each frequency defined for the system.

Since March 2013 the load model for harmonics studies can be modified by the user in OpenDSS. Additionally, in February 2014 the sequential-time simulation for harmonics mode was introduced to OpenDSS through DSSim-PC. Both items are described as follows:

The Load model in OpenDSS

A one-line diagram of the OpenDSS Load model in harmonics mode is shown in Figure 5-2. It is conceptually a multiphase Norton equivalent with the shunt admittance in the model consisting of a parallel R-L part and a series R-L part. The values for the variables G, B, R, and X are nominally determined from the specified load kW and kVAr values at 100% rated voltage. The current source value is determined from the fundamental frequency power flow solution of the distribution system. The current in the load computed from the power flow, I_{fund}, is modified by the multiplier in the Load object's assigned Spectrum object at each frequency. The phase angle of the I_{fund} is rotated appropriately for each frequency in the harmonic solution (Dugan, Arrit, Henry, McDermott, & Sunderm, 2014). OpenDSS automatically populates the values in this model when it switches from a power flow mode to harmonics mode.

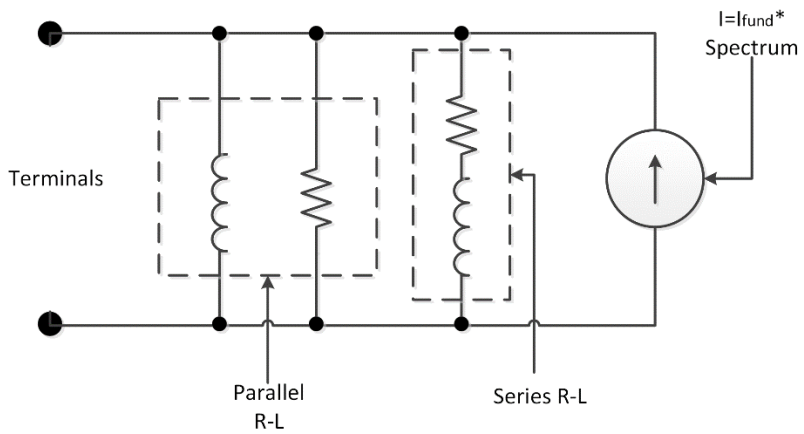


Figure 5-2. Load model in harmonics mode (Dugan et al., 2014)

By default, 50% of the load is assumed to be represented by the parallel R-L model and 50% by the series R-L model. One generally does not know exactly how a particular load should be modeled and this 50/50 mix has proven to be a good compromise. The mix can be changed by setting the *%SeriesRL* property. Setting this to 100% (all series R-L) will tend to predict conservatively high values of harmonic distortion. Setting it to 0% (all parallel R-L) yields lower distortion by providing more damping of system resonances.

Rotating machine load is best modeled by a series R-L model for harmonic analysis. However, when determined from specified power (kW, kvar) values used in the power flow, the series R-L branch is generally too highly resistive. The machine model should be more reactive with the reactance determined from the blocked rotor impedance for asynchronous motors. There is an option to specify the reactance of the series R-L branch in per unit of the kVA of the load to accommodate this. If there are many such loads on the distribution system the impact of employing this modeling approach is to shift system resonances to slightly higher frequencies.

There is also an option in the program for neglecting the shunt admittance in the load model entirely. The harmonic current from the source is directly injected into the system. This is generally acceptable unless the system is in sharp resonance. Then the model will predict impossibly high voltage distortion due to attempting to drive a current into the very high impedance of a system that is in parallel resonance. The main reason for using a Norton equivalent for the load model is to avoid this problem.

When the distribution system is not in resonance at a given frequency the mix assumed for the shunt admittances is of less importance. The system impedance is much less than the shunt impedance of the model and nearly all of the current source output is injected into the distribution system model.

The sequential-time harmonics simulation mode

The incorporation of the sequential-time simulation mode results from the question: What happens to the power system when harmonic loads get connected/disconnected at a given instant in time? This is because nowadays loads like EVs and systems based on power electronics are more common, and their energy exchange with the power system is dynamic in time. This behavior corresponds to a dynamic Total Harmonic Distortion (THD) present in the system, which requires a dynamic response of control devices for ensuring the reliable operation of the power system.

The sequential-time simulation mode for harmonics is inspired by the existing sequential-time modes of OpenDSS. The software solves the power system for all frequencies separately at each time step. This is repeated sequentially and can demand quite significant amounts of computation time as the system's size and number of harmonics or interest grow. In DSSim-PC the system's solution is performed by connecting the OpenDSS in-process COM server, OpenDSSEngine, as actor.

In harmonics simulation mode the processing times for each time step increases because the solver can only solve one frequency at the time. For this simulation mode the number of child actors will be equal to the number of frequencies in the defined spectrum; the aim is to assign the Y_{BUS} matrix of the system at each frequency to its corresponding child actor, thus reducing the processing time to solve the system at multiple frequencies as shown in **Figure 5-3**.

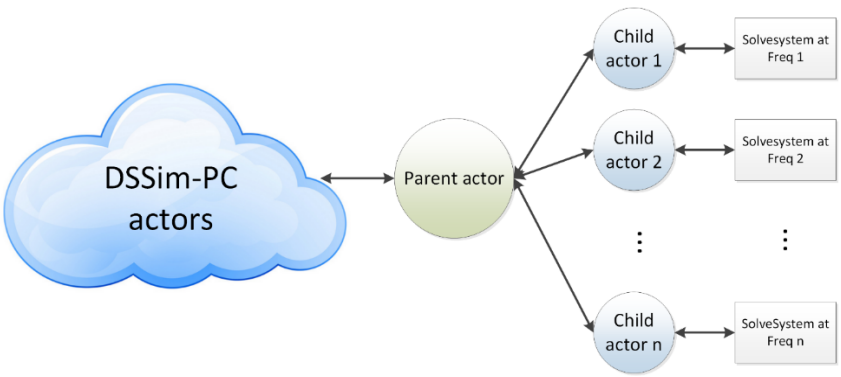


Figure 5-3. Content of the child actors when working in harmonics mode

In this simulation mode the performance will be not the best when the size of the system is large-scale; however, the performance will be much better than the sequential routine implemented in OpenDSS. When a load changes in time its representation as a current source is updated before the next time step. This way, the effects on the power system's harmonics after the connection/disconnection of loads can be simulated in time. This simulation mode takes advantage of the actor framework developed for A-Diakoptics to improve the performance of the sequential routine proposed originally in OpenDSS; nevertheless, this mode do not uses A-Diakoptics for solving the DS.

Additionally, harmonic meters and THD meters can be added by the user to the Graphical User Interface (GUI) for monitoring. These meters can be placed anywhere on the system and are updated at each time step. The maximized frontal panel of the harmonics meter is shown in Figure 5-4.

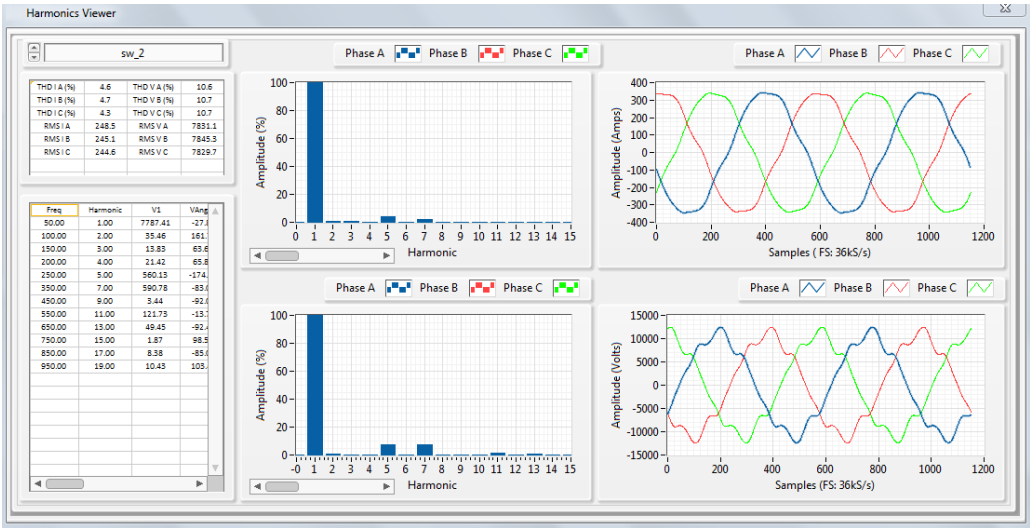


Figure 5-4.Harmonics meter graphical panel

The sequential-time TSA

Sequential-time TSA is the simulation mode equivalent to Time and Dynamic modes of OpenDSS. This simulation mode uses A-Diakoptics to solve the modeled DS including the dynamic behavior of the distributed generators, and replaces the original solver proposed by OpenDSS using the methodology presented in chapter 3. However, the child actors of A-Diakoptics utilizes the same algorithm to solve the primitive networks used in OpenDSS, which is the KLUSolver (Davis & Natarajan, 2010). Additionally, several models have been included to the simulation core inspired by the models programmed in OpenDSS; these models are integrated to the simulation by modeling them in Delphi (Barrow, 2005), generating the Thevenin equivalent of them and updating their state on each time step as shown in Figure 5-5.

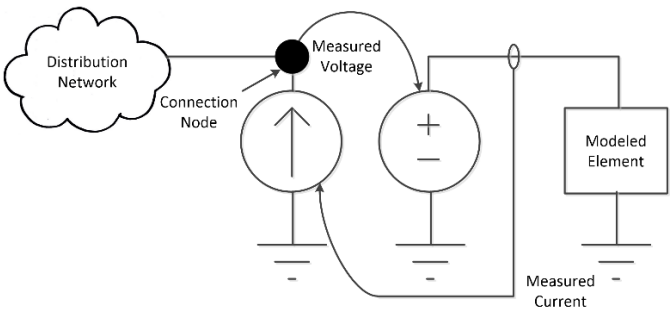


Figure 5-5. Exchange methodology to include new models in the simulation

These new models are represented as an impedance and a current source in the primitive network. However, for updating its state the voltage in the terminals of the element, which is the same at the connection bus, is feedback to calculate the value of the element's current source for the next time step. The list of the simulation modes implemented and the list of available elements in DSSim-PC is shown in TABLE 5-1.

TABLE 5-1
SIMULATION MODES AND AVAILABLE ELEMENTS IN DSSIM-PC

Type	Name	A-ODSS	Type	Name	A-ODSS
S-Mode	Time	Yes	Element	Wind gen	New
S-Mode	Harmonics	Modified	Element	Generator	Yes
S-Mode	Frequency Sweep	New	Element	House	Modified
S-Mode	Remote controlled	New	Element	Building	Modified
S-Mode	Dynamics	Yes	Element	Isource	Yes
S-Mode	Snap	Yes	Element	Vsource	Yes
Element	Load	Yes	Meter	Harmonics	New
Element	Capacitor	Yes	Meter	THD	New
Element	Reactor	Yes	Meter	Energy	New
Element	Storage	Yes	Meter	Stored energy	New
Element	VSController	Yes	Meter	Polar	New
Element	Electric Vehicle	Modified	Meter	Table	New
Element	PhotoV Panel	Yes			

A-ODSS = Available in the original version of OpenDSS
S-Mode = Simulation mode

Performance of the PC application

For testing the performance of the PC application of DSSim-PC, several power systems were modeled in this platform and in OpenDSS. The systems modeled were imported from the cases of study already modeled in OpenDSS using the .DSS to .DSP translator program included in DSSim-PC. With this program, the compatibility between the two platforms is ensured and the mismatch between the results is minimized due to the elimination of possible differences between the two models.

The systems selected to evaluate the performance are small-scale, medium-scale and large-scale DS, these systems and its features are listed in TABLE 5-2 (Fuller, Kersting, Dugan, & Jr., 2013). For the tests performed on standard PC the hardware architecture consists in an Intel Core^{TM} i5 processor 2.4 GHz (2 cores), 8 GB RAM and OS Windows 7 professional.

TABLE 5-2
CHARACTERISTICS OF THE SIMULATED SYSTEMS

System	# of Nodes	# of Switches	# of Buses	Scale
IEEE 8500	8561	44	4876	Large
EPRI's Circuit 5	3437	73	2998	Medium
EPRI's Circuit 7	2455	12	1256	Medium
IEEE 123	281	9	133	Low

After passing through the translator from .DSS to .DSP, the DS modeled in OpenDSS is represented graphically in DSSim-PC; this way, the user is able to modify the model using graphical tools, to incorporate graphical meters for storing simulated data into comma separated value file format (.CSV), to identify the network topology and visualize changes performed in the middle of the simulation.

DSSim-PC uses a model of layers for representing large-scale DS. This model consists in to create reference buses, which will be used for containing sub-networks of the system composed by lines, transformers, power conversion devices and power delivery devices (except for the slack bus of the DS). As a result, the complexity when drawing a large-scale DS is reduced and scaled to different

levels for accelerating the GUI refreshing process. As an example, two DS are shown in Figure 5-6. The first network (EPRI's Circuit 7) is modeled using only one layer, while the second network (IEEE 8500 nodes) is modeled using 2 layers.

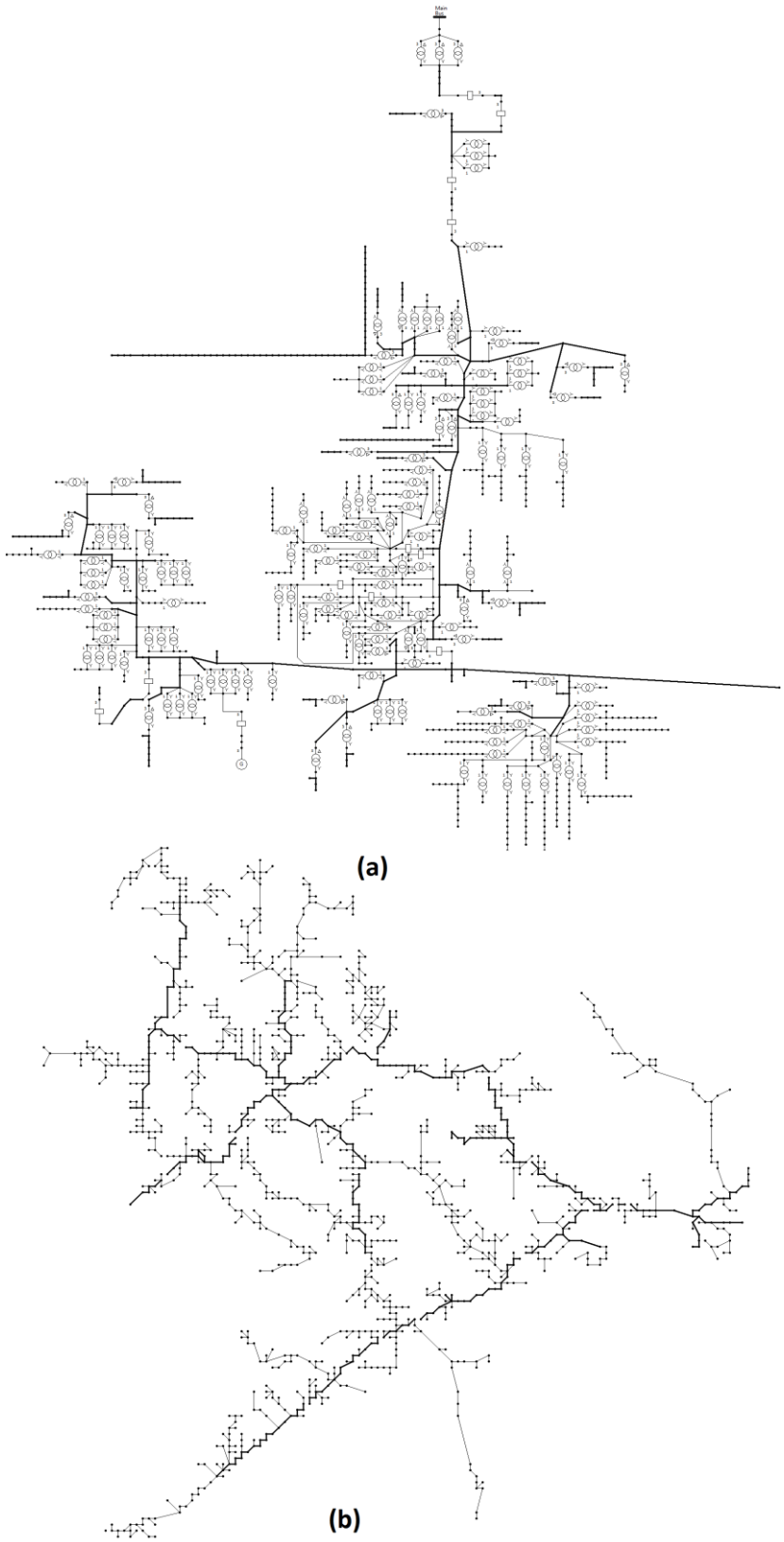


Figure 5-6. Medium-scale (a) and large-scale (b) DS modeled in DSSim-PC using the translator from .DSS to .DSP. Network (*a)* has one layer and network (*b)* has two layers.

The concept of layers is illustrated in Figure 5-7; in this Figure the modeled system has 2 layers with 2 connection points. For going from one layer to another, the user only has to make double click over the interface bus, which is the bus that connects the different layers of the graphical model.

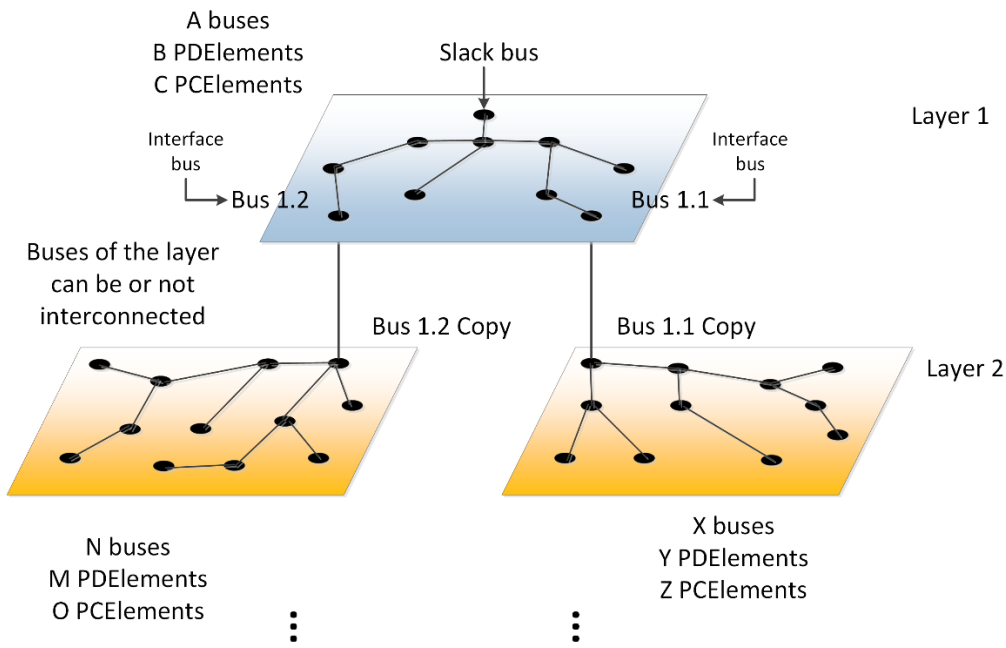


Figure 5-7. Layers model for representing small-scale, medium-scale and large-scale power systems

In the model of layers the number of layers can grow indefinitely. A new layer is created when an interface bus is defined within the layer; then, the destination layer is defined by the user, which normally is a lower level layer. The higher layer contains the slack bus while the other layers can include DER and other devices.

When importing a model from OpenDSS to DSSim-PC, the system uses a predefined bus coordinate file provided by EPRI; this file describes the location (x, y coordinates in pixels) of some elements included in the model, the elements non-included in this file are assigned to lower level layers (normally layer 2) for complementing the model graphically. The number of buses, lines and elements contained on each layer can be different.

The systems modeled in DSSim-PC and OpenDSS are simulated for evaluating the average simulation time required using 10000 iterations, this analysis do not considers the time required to compile the DS. The results of this test stage are shown in TABLE 5-3 and in Figure 5-8.

TABLE 5-3
RESULTS ACHIEVED FOR 1 ITERATION WITH STANDARD PC

System	Architecture	# Cores used	Solution time	Solver	Memory used (MB)	# of areas
IEEE 8500	PC	1	46.9 ms	OpenDSS	93.45	1
IEEE 8500	PC	2	15.6 ms	DSSim-PC	57.56	15
EPRI ckt 5	PC	1	11.3 ms	OpenDSS	71.98	1
EPRI ckt 5	PC	2	1,6 ms	DSSim-PC	31.28	13
EPRI ckt 7	PC	1	10.3 ms	OpenDSS	64.49	1
EPRI ckt 7	PC	2	1.2 ms	DSSim-PC	21.25	10
IEEE 123	PC	1	325 us	OpenDSS	44.55	1
IEEE 123	PC	1	288 us	DSSim-PC	17.94	4

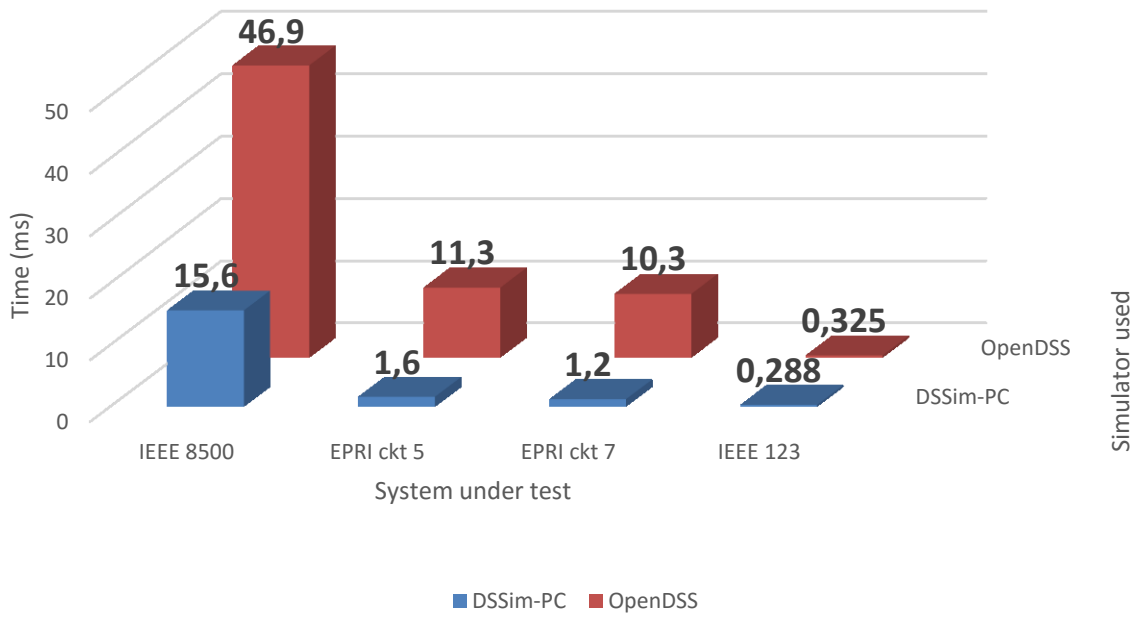


Figure 5-8. Simulation time for executing 1 iteration on DSSim-PC and OpenDSS

AS can be seen in Figure 5-8, the performance of the application is drastically improved when the size of the DS is large-scale and medium-scale; however, when the size of the DS is small-scale the improvement on the performance is almost negligible, in fact, the difference in time shown for the case IEEE 123 nodes is the reflect of windows services running at the time of the test (adding 50us). The improvements in the bigger networks are supported by the use of several cores when performing the simulation, this is the reason for which the improvement on the small-scale network is neglected.

The improvement in the medium and large-scale cases is also consequences of the optimization in the memory use. The A-Diakoptics model uses less RAM memory due to the reduction of global variables for sharing information between actors; moreover, because of the selected messaging model (zero coupling) the memory needed for sending data between actors depends on the size of the message content, but this memory is released once the message is received by the destination (Instruments, 2013).

The second part of the test consists into verify the time required for performing a simulation of 365 days in sequential time mode, the time step the simulation is 1 hour and each load within the DS under tests has associated a load profile, so the load changes in time. Additionally, a control program developed in MATLAB is connected to each simulation using the interfaces available on each case.

The connection with DSSim-PC is performed using the TCPS actor and the control program is executed in another computer to maintain the actor conception, which consist in to dedicate hardware resources to each actor, as in this case, to an external actor. Nevertheless, if desired both applications can be executed on the same computer. On the other hand, the connection with OpenDSS is performed using the COM interface; however, due to the nature of the interface the control program must be executed in the same computer. These test cases are presented in Figure 5-9 and Figure 5-10 respectively.

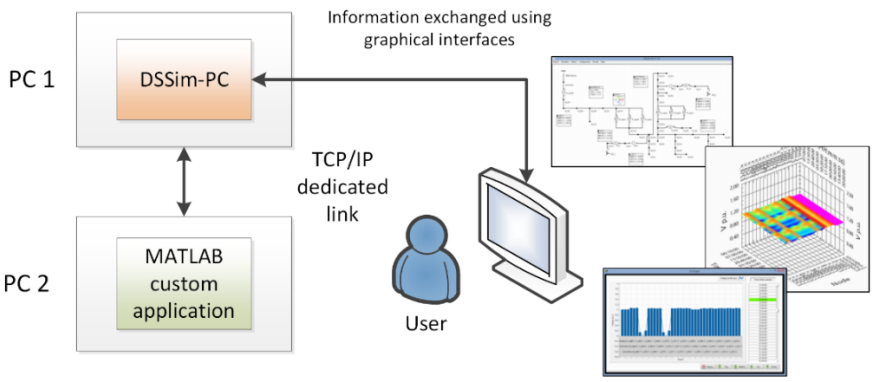


Figure 5-9. Testing scenario using DSSim-PC

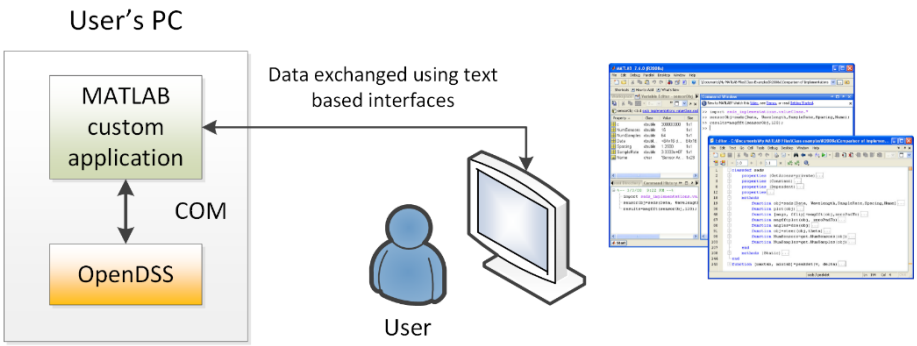


Figure 5-10. Testing scenario using OpenDSS

The results obtained after performing the simulations for the test scenario 2 are shown in Figure 5-11.

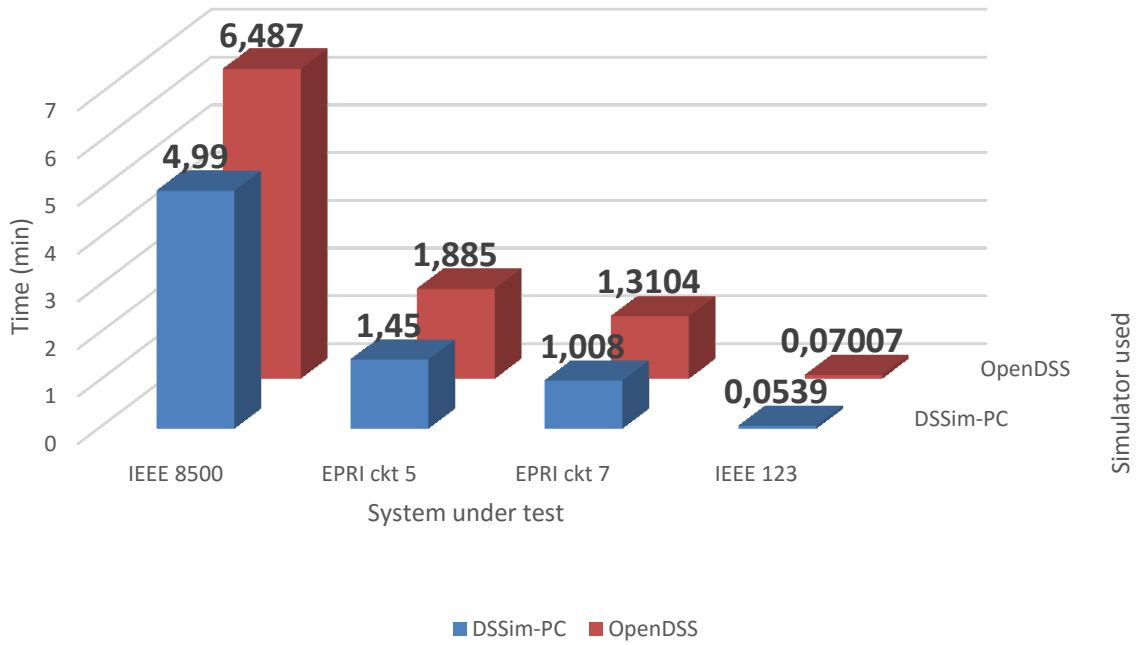


Figure 5-11. Results obtained in test scenario 2

The control routine programmed in MATLAB is the same in both cases, the difference is located in the interface used for exchanging data. This routine basically reconfigure feeders (if possible) and adds or removes capacitors when the voltage level is lower than 0.96 p.u. As can be seen in Figure 5-11, apparently the most significant improvement when simulating the proposed systems for a year is given with the large-scale network.

However, when calculating the improvement level in percentage in all cases is equal to 23%. Of course, when the simulation times are bigger the amount of time gained becomes significant. These results reveal the effect of including external actors to the simulation process, and how A-Diakoptics improves the performance of the entire simulation process when this situation happens. Additionally, several external actors can be incorporated to the simulation using the TCPS actor included in the proposed framework for DSSim-PC.

This experiment reveals the effect of the interfaces when performing co-simulations; MATLAB gets access to the OpenDSSEngine COM interface by using late bindings (Rogerson, 1997), which adds an important overhead to the co-simulation. On the other hand, DSSim-PC gets access to the KLUSolver DLL by using direct connection methods, which will be the equivalent to early bindings, thus reducing the time required to find a solution and improves the performance of the code developed in MATLAB when interacting within a co-simulation environment. Additionally, it reveals the effect when using high speed communication interfaces such as the Ethernet-TCP connections, which do not add an important overhead to the co-simulation environment.

The third test consist in to determine the improvement in terms of computational burden when using A-Diakoptics in standard PC architectures, but to perform this test, it is necessary to know the relationship in terms of performance between NI LabVIEW and Embarcadero Delphi ("Delphi-Embarcadero Technologies Inc.," 2015), which is the programming language on that OpenDSS is programmed.

To perform this comparison a program to access the OpenDSSEngine using early bindings is developed using NI LabVIEW and Delphi. In both cases, the ActiveX connection method used is early bindings. In this test, 3 different functions included in the OpenDSSEngine are used by the proposed programming languages; each function is used iteratively 1177000 times from each software and the total time required to complete the task is measured. The results are shown in TABLE 5-4.

TABLE 5-4
Time required when accessing the same OpenDSSEngine functions using Standard PC and the COM interface (early bindings) (Dugan & McDermott, 2011)

Function used	# of iterations	Time required by Delhpi (ms)	Time required by NI LabVIEW (ms)
Fix Load active power using class ActiveCircuit.Load.kW	1177000	613	638
Fix Load active power using the text interface Text.Command	1177000	4360	4527
Fix Load active and reactive powers using the text interface Text.Command	1177000	8477	8825

As can be seen in TABLE 5-4, the difference in time between the two languages is 4% in average; this means that there is a small overhead added by LabVIEW when accessing the COM interface of the OpenDSSEngine, which reveals the difference in terms of performance between the software platforms used for developing the original OpenDSS (Delphi) and it's a-Diakoptics version (NI LabVIEW). Then, considering these results the systems proposed in TABLE 5-2 are simulated 10000 times using OpenDSS and DSSim-PC. In both simulations the loads follow a load profile and voltage

regulators located in transformers perform control actions. The results are shown in TABLE 5-5 and in Figure 5-12.

TABLE 5-5
Results when simulating 10000 iterations with OpenDSS and DSSim-PC

Simulated circuit	# of Iterations	Time required by OpenDSS (ms)	Time required by DSSim-PC (ms)
IEEE 8500 Node	10000	43653	18771
EPRI's Circuit 5	10000	32754	22273
EPRI's Circuit 7	10000	23698	175537
IEEE 123 Node	10000	1466	2851

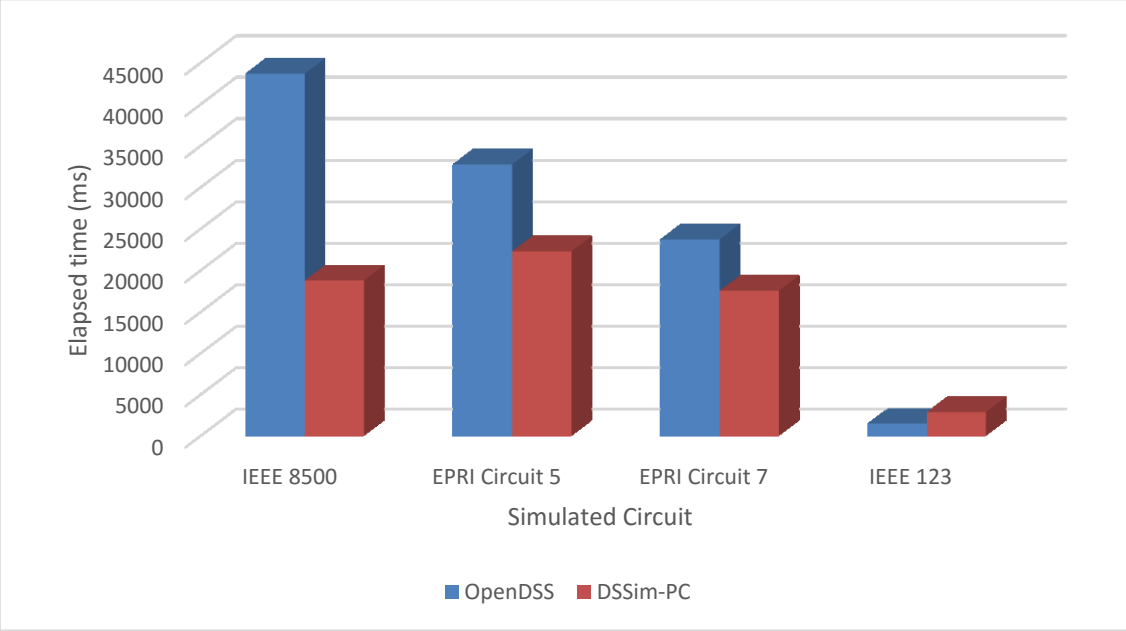


Figure 5-12. Results when simulating 10000 iterations with OpenDSS and DSSim-PC (Graphical)

The results shown in Figure 5-12 reveal the real improvement reached when working with standard PC operating systems. As can be seen, the improvement in terms of processing time become representative only when the size of the circuit corresponds to medium or large-scale. Additionally, it is evident the effect of letting the operative system to assign automatically the threads for the execution of the child actors; as a result, even if the execution time in OpenDSS gets improved the execution time of the actor framework can be affected.

These results also reveal the relationship between the performance of the A-Diakoptics method and the size of the simulated power system. As can be seen in Figure 5-12, when the size of the circuit is not medium or large-scale the results in terms of performance can be not as good as expected, but on the contrary, instead of having an improvement an extra overhead will be added to the simulation. This result suggests that when the systems size does not correspond to a medium or large-scale circuit (Khaitan & Gupta, 2012) it will result more efficient to solve the system without splitting it. The conclusion mentioned above is supported by the difference between the times presented in Figure 5-12, these differences are shown in Figure 5-13. As can be seen in Figure 5-13, when the power system size is not big enough (IEEE 123 node), the best alternative to improve the computation time will be to analyze the system as a single block.

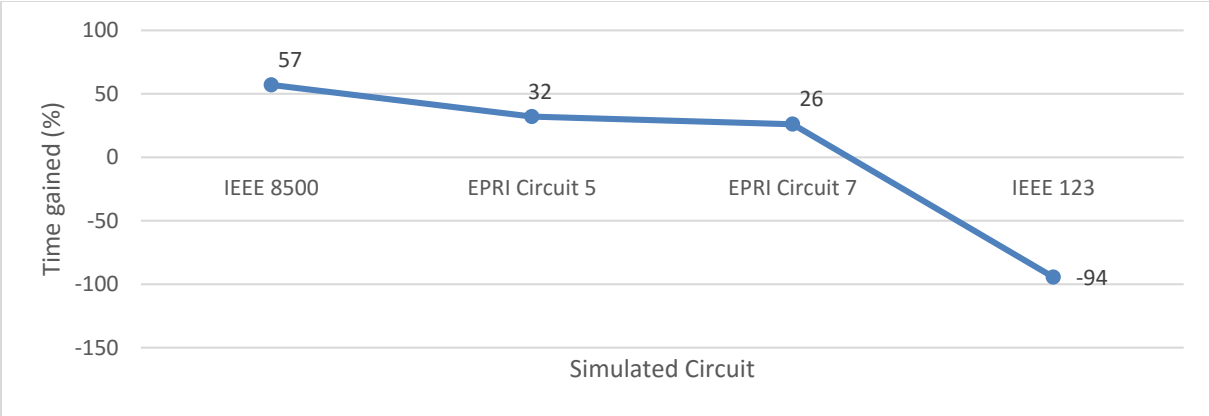


Figure 5-13. Computing time improvements (percentage) reached when using A-Diakoptics within PC architectures

The fourth and last test consist in to evaluate the simulation fidelity. For this test stage two systems are simulated in DSSim-PC and OpenDSS in dynamic mode; the aim is to verify the error when oscillatory modes get involved in the simulation. For doing this, the test system selected is the EPRI's circuit 7 and a distributed generator is connected through a switch to the bus x_1001805; this generator has a power of 1MW. The simulation is performed using a time step of 400us and the simulation window is 2 seconds.

In these simulations the switch that interconnects the generator and the DS trips after 200ms of simulation time, then, the generator is separated from the DS and starts to operate in island mode. After 600ms the generator is reconnected to the DS (800ms of simulation time) and the oscillations product of this reconnection are measured in different point of the network. 4 monitors are placed at nodes x_1001805 (generator), x_1001685, x_1001746 and 243937, the last 3 monitors are far from the generator. These monitors will measure the values p.u. of the voltage in the mentioned buses. The measurements taken in the simulation of the test number 3 are shown in Figure 5-14. Additionally, a comparison between the voltages p.u. provided by DSSim-PC and OpenDSS is shown in Figure 5-15, which is the magnitude of the voltage in the phase A at the bus x_1001805.

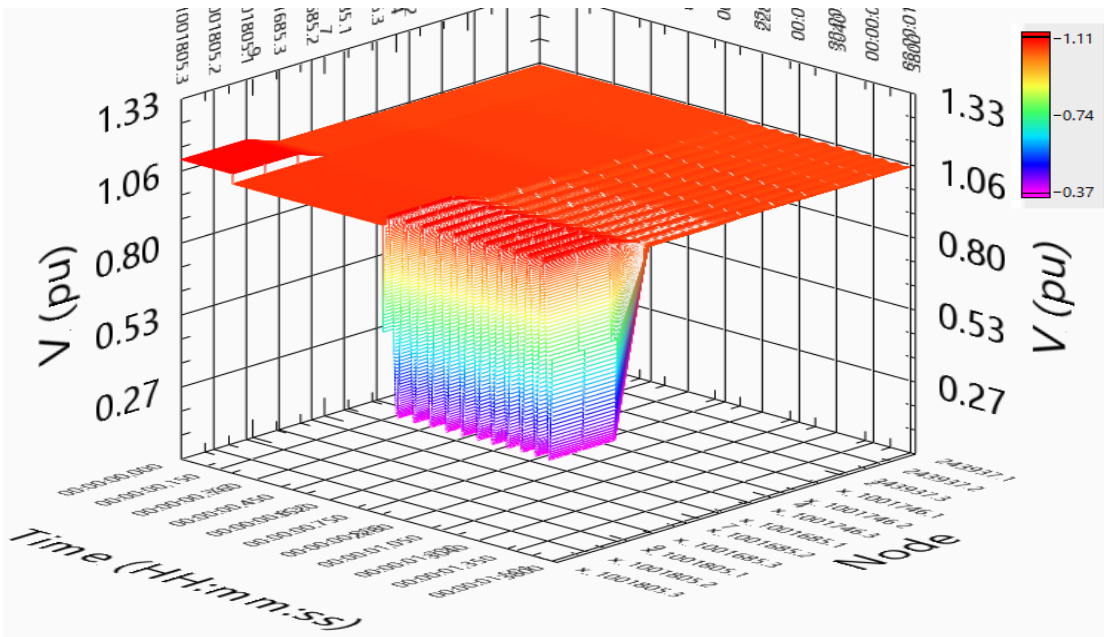


Figure 5-14. Measurements taken at different points of EPRI's circuit 7 in dynamic simulation, graphic generated with DSSim-PC (Torsional mode oscillations)- 1.5 sec

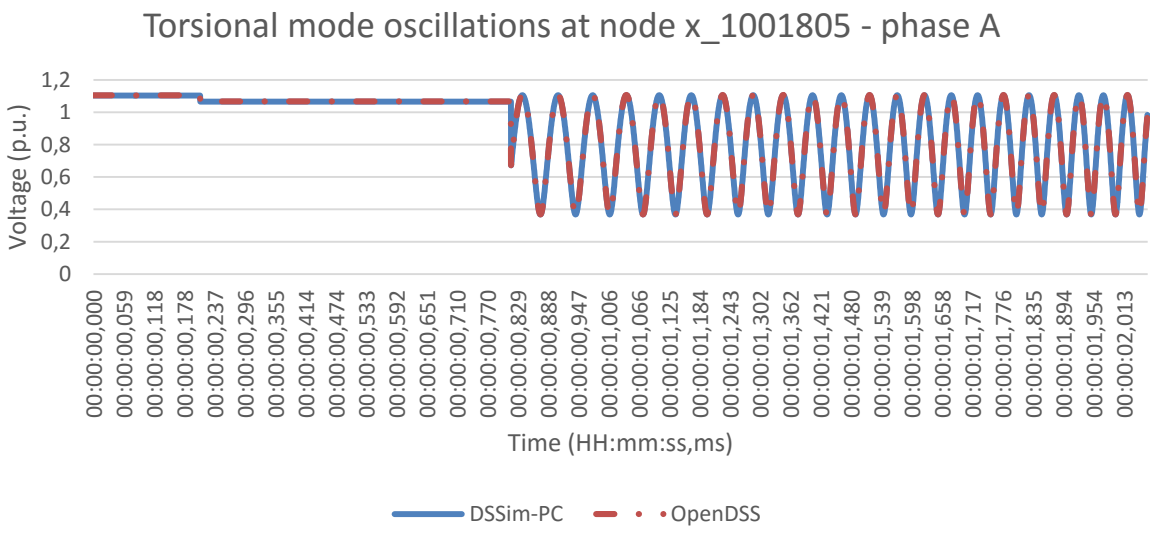


Figure 5-15. Torsional mode oscillations in Voltages (p.u.) provided by DSSim-PC and OpenDSS at node x_1001805 - phase A, 2 sec

As can be seen in Figure 5-15, the error between both voltages is very low. The maximum value of this error is 1E-4, which is evidence about the fidelity of the simulation performed by DSSim-PC referenced to the reference simulation platform OpenDSS. The same comparison is performed for the other phases of the bus and the other buses where the measurements have been taken; the obtained error was lower than 1E-4. Additionally, in the Figure 5-14 it can be seen how the oscillations are propagated through the rest of the power system and depending on the distance to the point where the disturbance is generated, the magnitude of the oscillation will be reproduced. In fact, by using multi-rate theory it is possible to reproduce a good approach of the AC voltage/current signal based on these results (Laddomada et al., 2011; Tan, 2007).

Consider the data presented in Figure 5-14 and in Figure 5-15, which was generated at a sample rate of 2500 samples per second (S/s), and a sinusoidal waveform generated at 36000 S/s. Both waveforms can be mixed to reproduce the behavior of the voltage/current accurately as follows: The waveform sampled at low frequency will be the data waveform in terms of amplitude and phase angle, while the waveform sampled at the higher frequency will be the carrier signal. As a result, the data waveform will update each 400us the value in terms of amplitude and phase angle of the carrier signal. For filling the space between one sample and the next of the data signal, the carrier signal will conserve its dynamic until the next data arrives. This concept is illustrated in Figure 5-16.

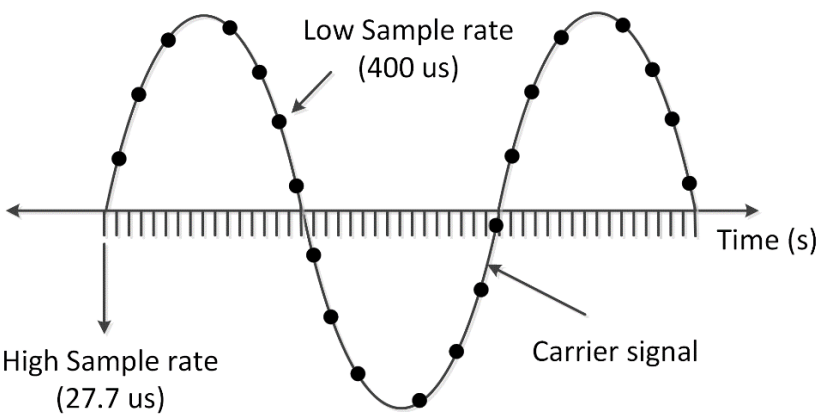


Figure 5-16. Multirate waveform generated by mixing the data and carrier signals

After applying the concept of the data and carrier signals to the data generated with the simulator, and approach to the real voltage waveform (p.u.) is obtained. This waveform is shown in Figure 5-17. In Figure 5-17 the resulting waveform has a sample rate of 36000 S/s and its magnitude and phase angle are updated each 400 us.

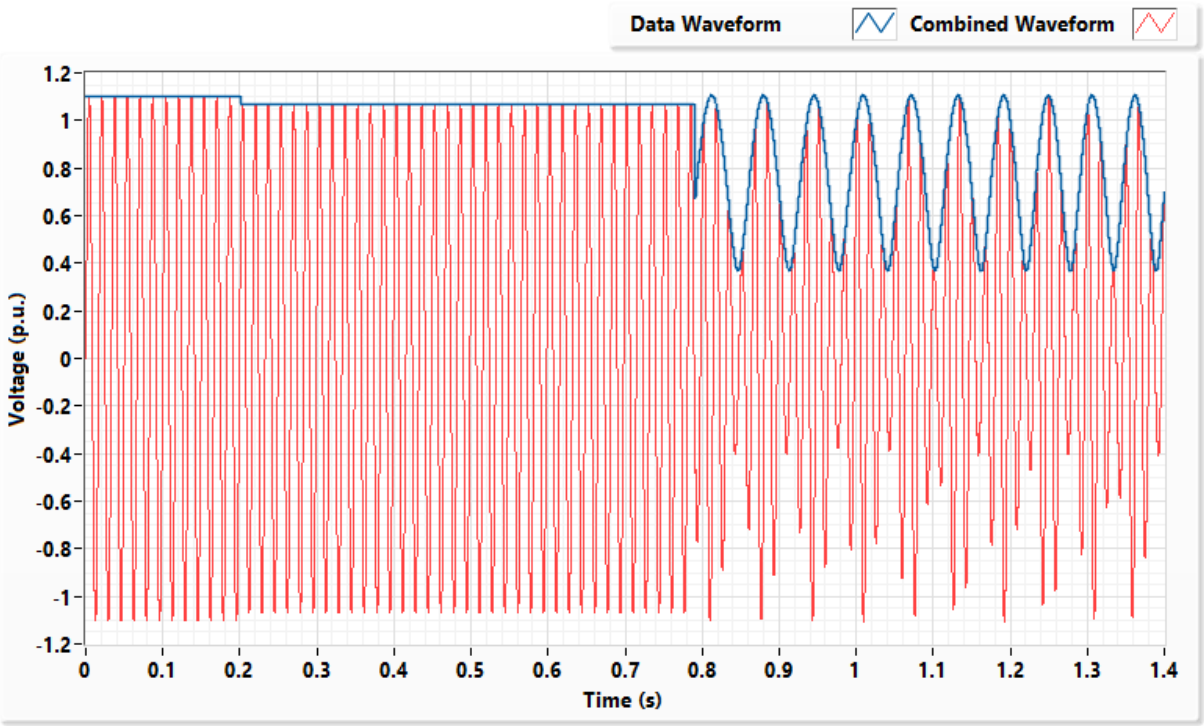


Figure 5-17. Data waveform and combined waveform (data + carrier) for reproducing the voltage signal according to the data generated.

Distribution System Simulation using Real-Time architectures

The RT architecture shares the same features than the PC version, but the structure of the actor framework and the way actors are built change. In this architecture, the parent actor controls the available hardware resources for being used by the child actors; this distribution of resources is made by using the procedure described in chapter 4. Additionally, the set of actors that composes A-Diakoptics are executed on a dedicated computer using a RT OS called LabVIEW RT ("Converting a Desktop PC to a LabVIEW Real-Time Target," 2012). This computer is a separate hardware that receives the data of the interconnected network from the launcher actor, which is executed with the other actors over a standard PC with a standard OS such as Windows 7. The RT computer counts with an Intel® Core™ i7-4810MQ Processor 3.4 GHz (4 cores), 16 GB RAM. The actor framework designed for this architecture is shown in Figure 5-18.

As can be seen in Figure 5-18, the connection between the two environments is made using a dedicated TCP connection through the Ethernet adapter of both machines. Moreover, the dedicated computation environment designed for A-Diakoptics includes a new actor, which is a dedicated TCP server to provide data to other actors connected to generate voltage/current waveforms, to simulate communication devices or to perform specialized tasks oriented to CHIL or PHIL. This means that the RT computer requires at least two Network cards: one dedicated to communicate with the nondeterministic GUI pre-compiler computer, and the other dedicated to attend the incoming requests from external actors to provide them the required simulation data.

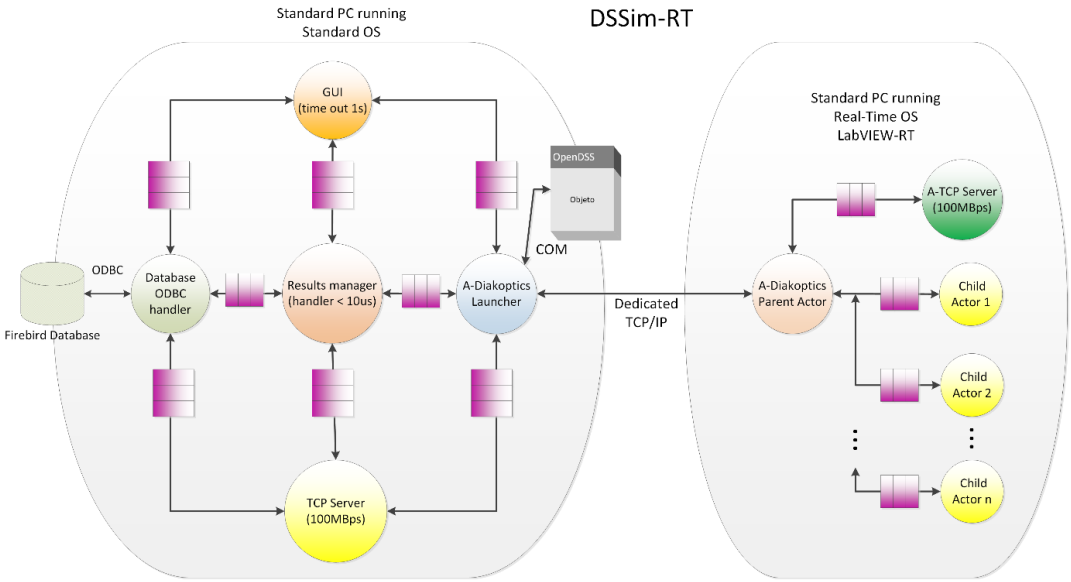


Figure 5-18. Actor framework proposed for DSSim-RT

This type of architecture is very common in existing RT simulation platforms (Dufour & Belanger, 2006; Jalili-Marandi, Ayres, Ghahremani, Belanger, & Lapointe, 2013; Mitra, Lidong, & Harnefors, 2014); however, the hardware proposed when implementing A-Diakoptics focuses on the utilization of low cost standard PC platforms for exploiting its multicore features.

For generating voltage/current waveforms, to simulate communication devices and to perform specialized tasks the hardware used is the NI cRIO 9082 (Instruments, 2012). As mentioned above, these external actors communicate with the parent RT actor using TCP-Ethernet connections. This heterogeneous computing environment is shown in Figure 5-19.

The heterogeneous computing environment proposed for DSSim-RT can be modified according to the local needs; for example, if the number of signals is not very high, the specialized remote actors (cRIO) can be replaced by other devices such as NI single-board RIO ("NI Single-Board RIO Out Of the Box," 2012) or less expensive controllers. Since the sample rates and protocols are respected, the modularity and scalability of the system can be customized as desired.

Because the hardware resources are dedicated to solve the system at each time step, the expected computing times required to solve each one of the test systems early proposed should be lower. To verify this hypothesis the first test performed consist in to simulate one iteration and measure the computing time required. The results of this test are shown in TABLE 5-6.

TABLE 5-6
RESULTS ACHIEVED FOR 1 ITERATION WITH DEDICATED OPERATING SYSTEM AND HARDWARE

System	Architecture	# Cores used	Solution time	Solver	Memory used (MB)	# of areas
IEEE 8500	RT	4	4.4 ms	DSSim-RT	57.56	45
EPRI ckt 5	RT	4	631 µs	DSSim-RT	31.28	73
EPRI ckt 7	RT	4	623 µs	DSSim-RT	21.25	13
IEEE 123	RT	1	150 µs	DSSim-RT	17.94	8

These results are compared with the results obtained previously for the standard PC architecture, this values are presented in Figure 5-20.

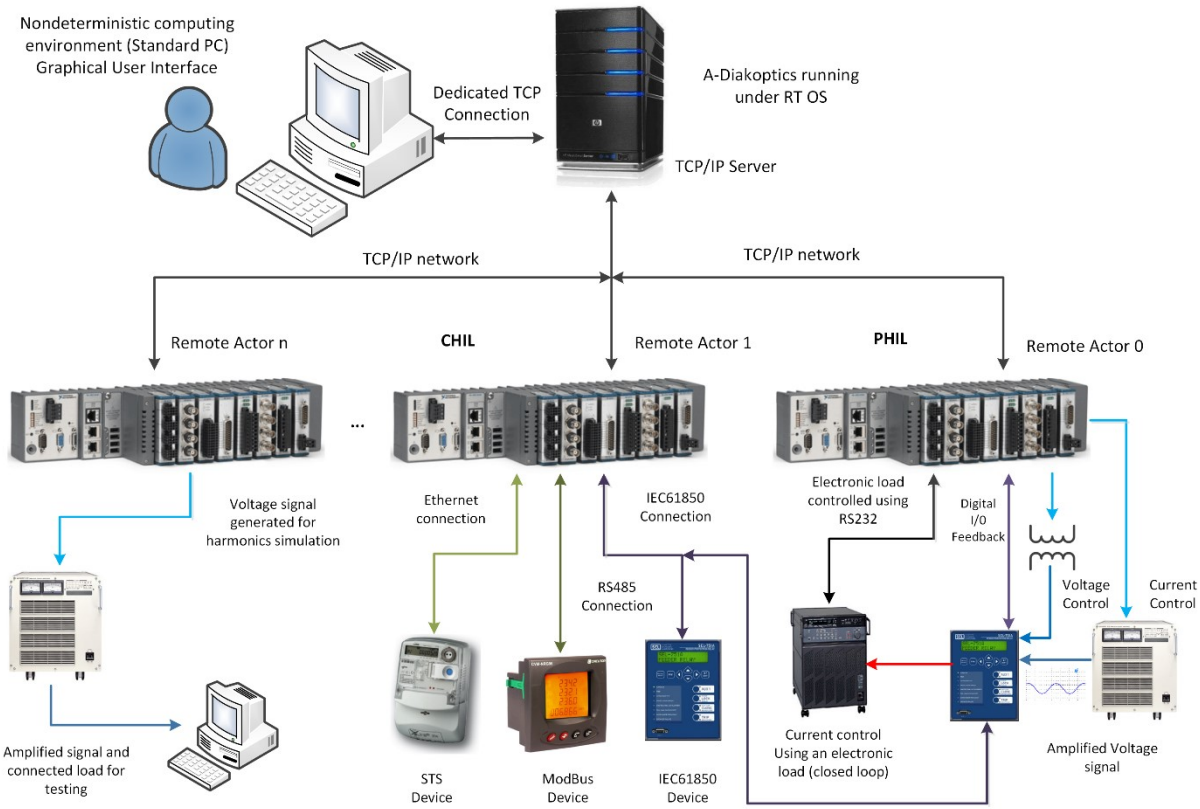


Figure 5-19. Heterogeneous computing environment proposed for implementing DSSim-RT

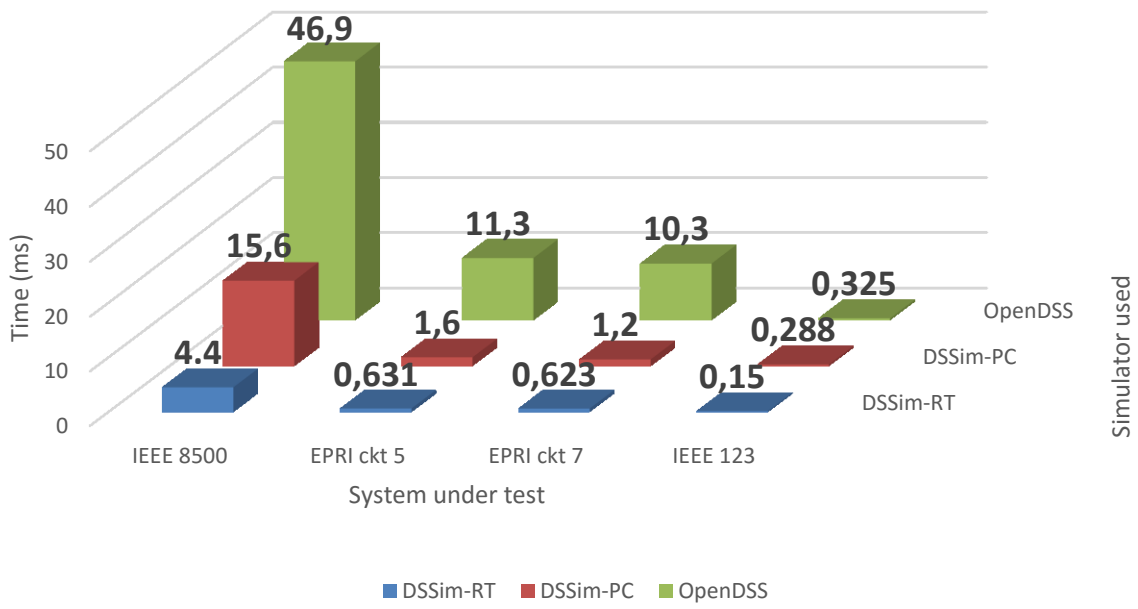


Figure 5-20. Performance comparison between PC and RT simulators using A-Diakoptics

As can be seen in Figure 5-20, the improvement in the computing time required when using the RT architecture is significant. The results achieved reveal the importance of the management of hardware resources, and how when they are dedicated to a single purpose the performance of an application can be improved representatively. The solution time required for one simulation step in the case of large-

scale systems results competitive with the times reported by existing RT HIL simulators ("ePHASORsim, the future of real-time simulation begins here," 2014).

Due to the results obtained in the first test with the RT architecture, the step time for the application is set to 4.5ms, which means that the simulation will refresh all the simulation values in sequential-time simulation every 4.5ms. Considering these parameters the waveform generation routine will update the magnitude and angle of the waveforms synchronously each 4.5ms. To guarantee high determinism while generating the voltage and current waveforms, a local parallel computing system is required. The NI cRIO is a multicore processing system with an FPGA integrated; allowing the separation of specialized tasks to guarantee high determinism in terms of computing time.

The generation system consists of two parts: the first is the communication and processing actor, which requests the data about the magnitude and phase angle of the element observed in the power system. This data is delivered with the type of waveform to generate to the waveform generation routine, this routine is implemented on the FPGA of the NI cRIO (hardware actor), which updates periodically the values of magnitude and phase angle every 4.5ms as follows.

$$\boldsymbol{y(n) = x(n) * \cos\left(\frac{2\pi n}{F_s} + \varphi\right)} \qquad \textbf{5.1}$$

Where n is an integer representing the time step in the discrete time domain, $x(n)$ is the magnitude of the waveform, F_S is the sample frequency used in sample per second (36kS/s) and φ is the phase angle of the waveform. Equation 5.1 is used for generating all waveforms.

The waveforms shown in Figure 5-21 are generated with DSSim-RT when simulating the separation and reconnection of a 1MW generator on EPRI's circuit 7. The localization of the generator corresponds to the one proposed for the test 4 of the standard PC architecture.

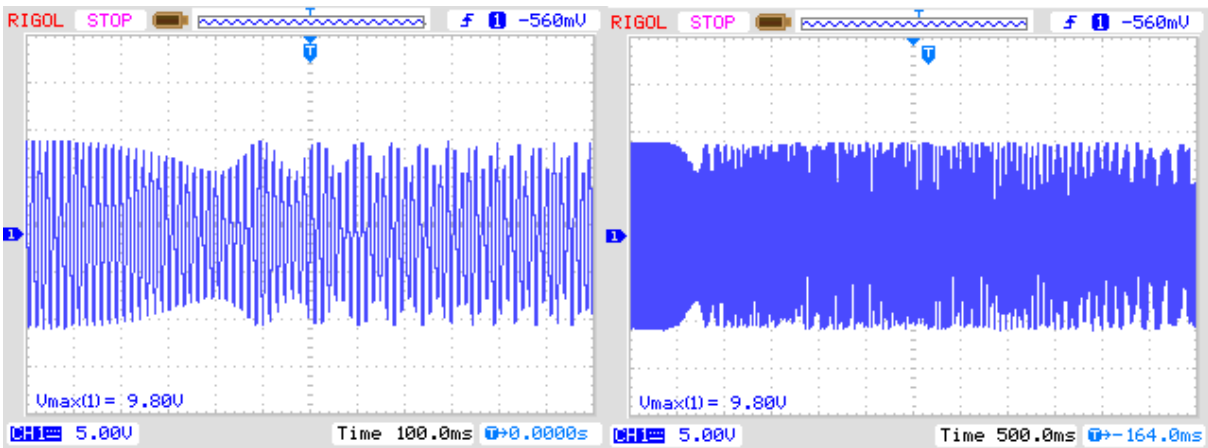


Figure 5-21. Waveform generated with NI cRIO after the separation of a 1MW generator in dynamic mode, the waveform changes in magnitude and angle due to the torsional mode oscillations. Phase A

As can be seen in Figure 5-21, allows to reproduce the oscillation modes generated in the TSA simulation. In fact, it is expected that improving the hardware architecture the sample time used for refreshing the magnitude and phase angle of the generated waveforms. To verify this hypothesis, the large-scale system proposed for the tests (IEEE 8500 nodes) is simulated using different number of cores to see the trend when this resource is incremented. The result of the tests is shown in Figure 5-22.

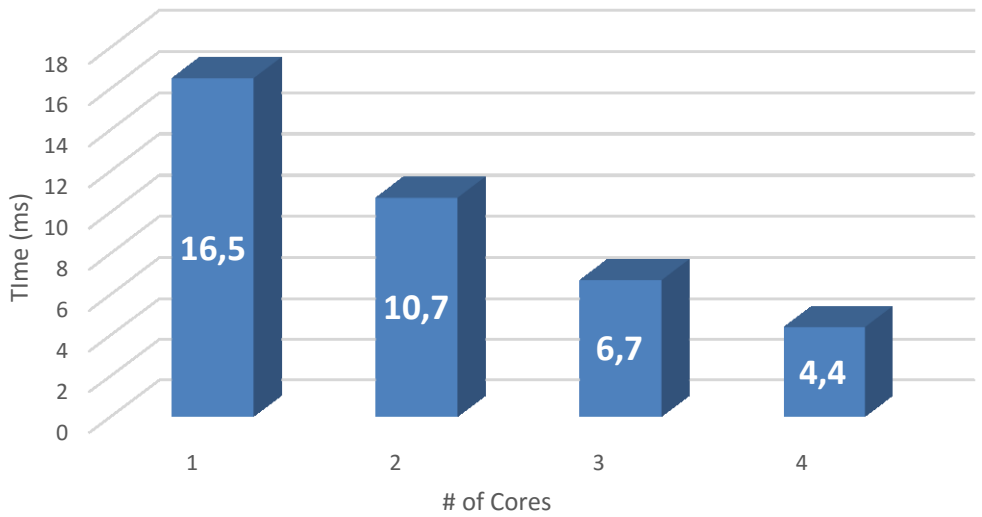


Figure 5-22. Simulation time (1 iteration) of the IEEE 8500 node test system simulated using several cores

These results are used to find the fitting curve for trying to predict the trend of the simulation time when increasing the number of available cores. This curve is proposed using a third order polynomial equation as follows:

$$y(c) = 24.8221 - 9.6859c + 1.4564c^2 - 0.078c^3 \qquad 5.2$$

The comparison between both curves, the estimated time and the measured time, is shown in Figure 5-23. Then, using equation 5.2 a projection is made for inferring the behavior of the simulation time when increasing the number of cores used for simulating the proposed power system. This projection is shown in Figure 5-24, and provides a general idea about what to expect when incorporating a large number of cores for performing the simulation in RT; however, this data must be confirmed experimentally as part of future work using more powerful computing hardware.

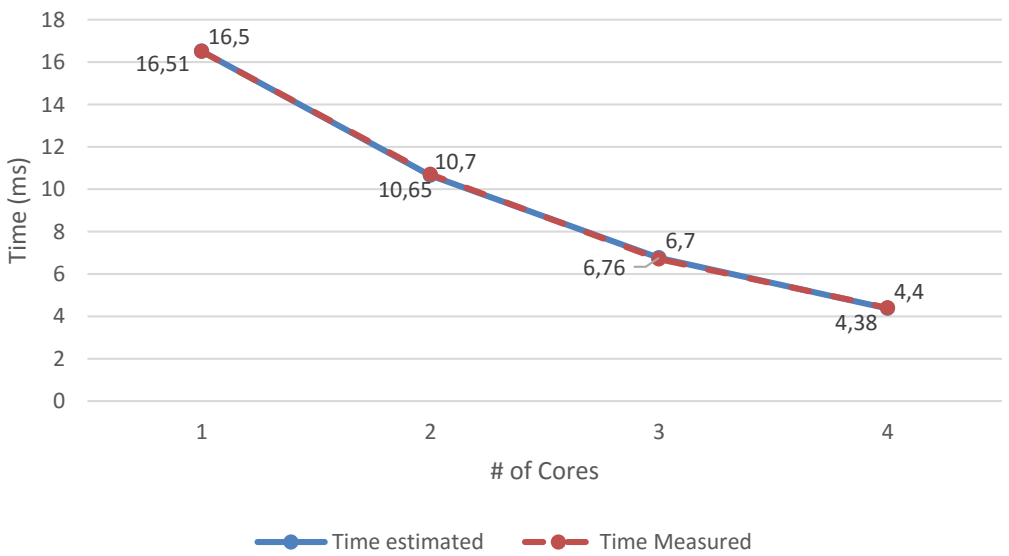


Figure 5-23. Comparison between the measured times and estimated times using equation 5.2

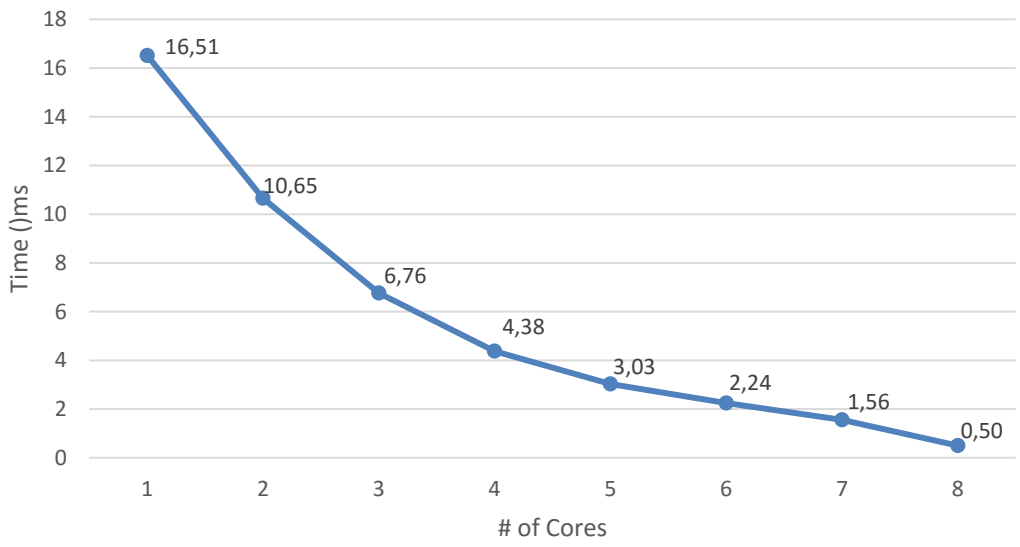


Figure 5-24. Projection for estimating the behavior of the system processing time when the number of cores on the simulation increases

Other future applications

DS state estimation

Real Time (RT) SCADA monitoring of electrical power systems has become a major topic in the last years due the need to manage the increasing number of Smart Grid applications (Arritt & Dugan, 2011; Dugan, Arritt, McDermott, Brahma, & Schneider, 2010; Giri, Parashar, Trehern, & Madani, 2012; Kezunovic, Vittal, Meliopoulos, & Mount, 2012; Zahedi, 2011). In the case of the DS applications such as the inclusion of DER, feeder reconfiguration, fault location and isolation and electric vehicle integration (Abdulhadi et al., 2011; M. E. Baran, Hooshyar, Shen, Gajda, & Huq, 2011; Zavoda, 2010), demands the use of RT energy management system (EMS), which must be compatible with many other services at commercial level using the same communication infrastructure ("Personal Energy Management," 2013; Xuan & Bin, 2008; Zhong et al., 2013; Zhongwei & Jing, 2010).

For these reasons, the observability of the DS has been increased by installing Intelligent Electronic Devices (IED) for monitoring and control purposes (Jerome, 2001; Powalko, Rudion, Komarnicki, & Blumschein, 2009); the utilities has started to install Smart Meters (SM) at the customer side, at points of common coupling and transformers to collect information about the system state and offer a new services portfolio (Zavoda, 2008).

Nevertheless, handling and processing this amount of data represents a heavy challenge in terms of computational performance for RT applications. Because all the measurements are collected using the same communication infrastructure, characteristics such as speed and bandwidth represent a bottleneck as an outcome of the time involved for recording data from a single IED. As a result, the performance of the monitoring system will not result as it was expected (Shahidehpour & Wang, 2004).

To handle these issues, State Estimation algorithms (SE) propose advanced methodologies for monitoring the power system using an optimal number of measurements looking for:

- Increasing the robustness of the monitoring system due the existing redundant measurements
- Reducing the number of transactions needed to estimate the current state of the power system
- Making the EMS compatible with the current and future measurement infrastructure (Abur & Expósito, 2004; Monticelli, 1999).

However, in DS the number of nodes and the network topology increases the complexity of the SE algorithm, and if it is built following conventional serial vertical programming strategies (M. Baran & McDermott, 2009) could degrade the EMS RT performance when Smart Grid management activities are performed (Kersting & Dugan, 2006; Uluski, 2010).

SE algorithms are based on the weight least square (WLS) algorithm proposed for transmission systems; in fact, Branch Current SE (BCSE) is the most used algorithm for performing SE on DS (M. Baran, 2012; M. E. Baran, Jaesung, & McDermott, 2009; Haibin & Schulz, 2004), because of its efficient in handling line-flow and power-injection measurements for radial networks and is compatible with voltage measurements taken from large scale Advanced Metering Infrastructure (AMI) (M. Baran & McDermott, 2009). However, when adding more measurements for reducing the estimation error of the BCSE algorithm, there are latencies that can increase the estimation time.

These latencies are product of several processes associated to the communication process, which includes the preparation of the message to be send, the field bus features to send the data, the way the message is handled on the destination and the time required to respond to the received command. These processes can be summarized in 4 stages:

- Generating the command to send
- The time required for transmitting data to the destination (includes interfacing elements such as routers, switches, signal amplifiers, among others)
- The arriving of the data to the destination and its processing for being handled
- The action performed by the destination as product of the incoming messages

These stages are illustrated in Figure 5-25. As can be seen in this Figure, the largest time is normally associated with the communication channel depending on if it is wireless, cooper pair, optic fiber or any other physical medium used for transmitting data. However, this Figure considers only the case when the communications channel works as expected; the reliability of the channel, which is associated with the probability of failure of this, will add more time due to the timeout delay when a failure occurs (Vasquez et al., 2010).

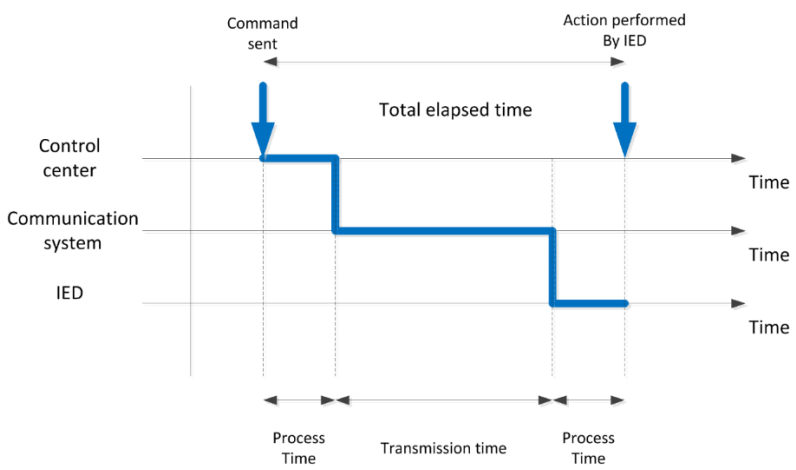


Figure 5-25. Latencies by using a communication network to communicate with IEDs

Classically, all measurements are centralized at the substation of the DS for estimating its state, but due to the features exposed above, this data acquisition methodology could not be enough to reach the expected step time to perform ADA activities to handle the DS (Giri et al., 2012). This methodology is presented in Figure 5-26. As can be seen in this Figure, each time a new measurement is included to improve the observability of the DS, then more demanding will be the expected reliability of the communication links and more delays will be added to the state estimation system (Northcote-Green & Wilson, 2006; Shahidehpour & Wang, 2004).

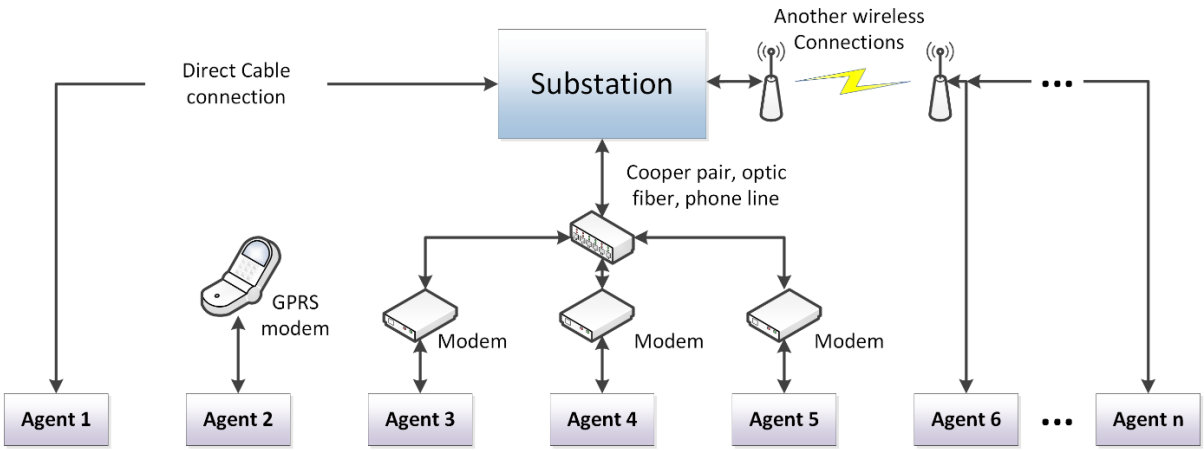


Figure 5-26. Classical methodology for gathering data from distributed meters (Agents)

By applying A-Diakoptics, the system can be separated into several zones of observation, which will feed the substation (Parent actor) using the information processed by the satellite management systems (child actors). Each child will manage all the agents of the zone by determining the necessary measurements for estimating the state of the local zone (Akbari, Setayeshmehr, Borsi, Gockenbach, & Fofana, 2010); then, these actors will provide the information about I_0 and E_0 to the parent actor for being updated in all the child actors to calculate E_1 and complement the partial solutions on E_0.

This architecture proposes to concentrate the readings of the agents in the child actors, which will reduce the number of centralized reading to just a few number gathered by the local EMS characterized by the child actors. These actors sill be commanded by the parent actor, reducing the amount of data and connections required to estimate the state of the DS (Bevrani, Daneshfar, & Hiyama, 2012).

These child actors are intelligent agents (Kouluri & Pandey, 2011; Zhong et al., 2013) that will decide which meters to read and will manage the switching devices that connects the zone to the interconnected network; monitoring locally the state of the link branch between the local primitive network and the other primitives, and updating the state of this branch on the portion of the Z_{CC} for complementing the partial solutions provided by the child actors. This architecture is shown in Figure 5-27.

This architecture will allow to handle a large number of measurements by adding a modular topology for sharing information, where the data is read using parallel local EMS that will feedback the substation to complete the partial calculations performed by these intelligent agents.

Additionally, it is expected that the time required for gathering data from the distributed agents will be lower than in the traditional case presented above. Because each child actor handles the communication with the local set of agents, several reading will be performed in parallel and their processing will be performed locally before complementing the total solution.

This application of A-Diakoptics is part of the future work proposed by this thesis, which can be assessed by using RT-CHIL and/or RT-PHIL.

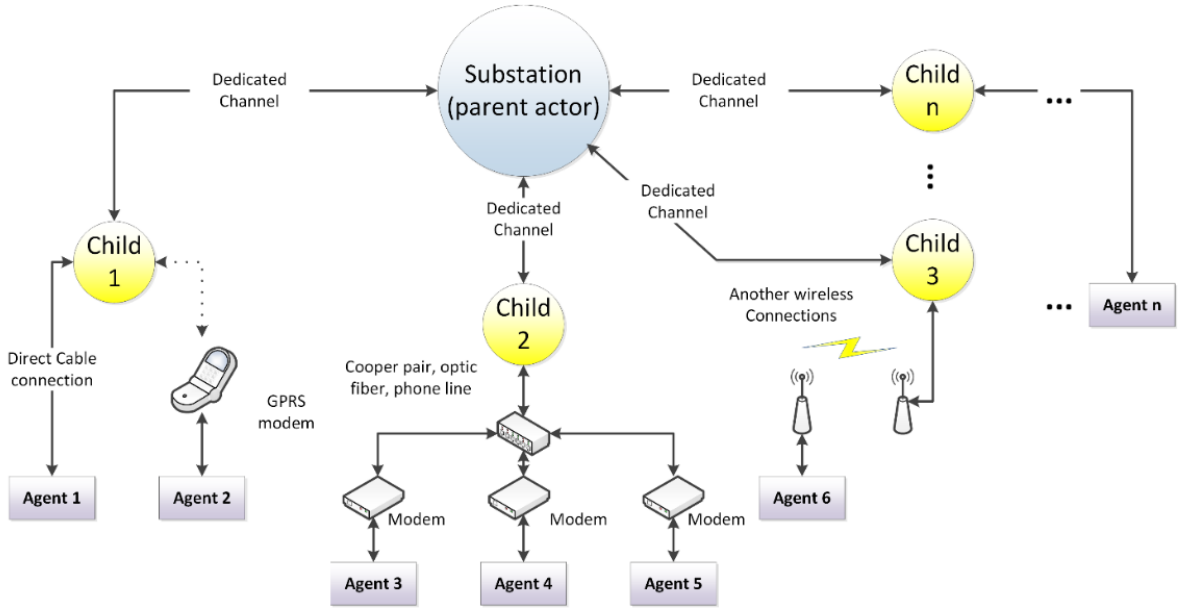


Figure 5-27. State estimator architecture using A-Diakoptics

Advanced Distribution Automation

As shown above, A-Diakoptics is a methodology for parallelizing processes and management activities that traditionally are performed sequentially in power systems and in other fields; this parallelization covers a wide spectrum of applications related with ADA. The aim of this section is to present how A-Diakoptics can improve the performance of several ADA activities considering the way they are performed classically, these applications are part of the future work proposed on this thesis.

Concepts such as wide-area distribution automation (DA) are proposed in literature for increasing the reliability of DS; however, several challenges are proposed at the same time (Greer, Allen, Schnegg, & Dulmage, 2011). These management activities have been developed an applied by utility engineers for decades, leading an evolution process that has deployed sophisticated monitoring and control systems in the last years (Zahedi, 2011), which has been driven by economical communication technologies and new embedded and powerful computation technologies.

Nevertheless, as the complexity of DS increases also the opportunities for increasing reliability by using automated management systems, which demands that the developers and users of these systems must be well trained and to keep them always in alert to avoid pitfalls in their applications (Nunoo & Ofei, 2010; Uluski, 2010). This means that depending on the way of use and the design of the EMS the effectiveness of traditional protection practices could be degraded; after the enter in operation of the automated system the DS will no longer stay in the same configuration/topology, the protections will no longer be coordinated due to these changes, among other features.

Many of these new features of the DS involve interaction with the dispatcher due to the effects that these could have on the power system; this requires a full understanding of the effects on the DS to not compromise the security, quality, reliability and availability (SQRA) of the DS (Fan, Chen, Kalogridis, Tan, & Kaleshi, 2012), which is one of the main concerns when designing EMS nowadays (Kezunovic et al., 2012). As can be seen in this picture, the features of modern EMS have taken the design and operation of these systems to a new and sophisticated level.

This sophistication has led to involve innovative methods to perform automated control activities. As an example, the feeder reconfiguration activity has been proposed to be implemented using artificial intelligence techniques such as particle swarm (Wu-Chang, Men-Shen, & Fu-Yuan, 2007), grey correlation analysis (Men-Shen & Fu-Yuan, 2010), black box optimization (Lei, Fang, & Xianyong, 2013), clonal selection algorithm (Narimani, Vahed, Azizipanah-Abarghooee, & Javidsharifi, 2014), enhanced gravitational search (Kavousi-Fard & Niknam, 2014), among others. The aim in all cases is to reduce the technical losses, to improve the load balance and voltage deviation and to supply energy to many customers as possible when an unexpected event occurs in the DS.

These methods are based on optimization functions focused into minimize or maximize the features mentioned above, but always considering the system topology which is described using graphs. These graphs could be built to consider single or multiple phases according to the network's characteristics. The solution of the optimization function is found using iterative algorithms for calculating the minimum or maximum values for the specified variables, which can take important periods of time when the size of the network increases.

As discussed in the previous section, by using A-Diakoptics the interconnected network can be decomposed in a set of smaller primitive networks; as a consequence, the system can be processed using parallel PUNs modeled as child actors, which are coordinated by a parent actor using communication systems. Then the DS can be scaled for reducing the time required for calculating the optimal values for each primitive, and then, to estimate the reconfiguration according to the coordinated values delivered by the parent actor once the child actors provide to it their partial optimal values.

The same analysis can be applied to fault location and isolation (Celeita, Zambrano, & Ramos, 2014; Dobakhshari & Ranjbar, 2015; Gong et al., 2010; Yanfeng & Guzman, 2013). This activity considers also the graph that describes the network topology to perform the fault detection and as a consequence, its possible isolation.

The most used algorithms for finding the most probable fault route are the undirected Dijkstra algorithm and the statistical hypothesis testing trying to cover DS and Transmission System issues. The time performance of these techniques referenced on graphs is highly dependent on the size of the graph such as in feeder reconfiguration, which suggests that this activity is also suitable for being improved using A-Diakoptics for achieving the reduction in the time required for its execution using a parallel and concurrent heterogeneous computing environment.

Other activities such as Volt/VAr control are also suitable for the application of A-Diakoptics and improving its performance. By separating the interconnected network using its electrical equivalent representation, the problem can be parallelized using a concurrent computing environment, which can be modeled using the actor model for exchanging information between actors and to coordinate the interactions between them. As a consequence, the EMS becomes highly scalable, modular and flexible to handle small, medium and large-scale DS, which can be also used for attending issues in transmission systems.

Although these ADA activities have not been implemented using A-Diakoptics, the simulator developed using this methodology has been used in several works where the goal is to test and validate ADA systems offline and RT. These works include the following:

- Connection Schemes for Electric Vehicle Supply Equipment (Ramos, Rios, & Rincon, 2014)
- Fault Location Algorithm based on shortest path optimization problem for Distribution Networks with DG (Contreras & Ramos, 2014)
- Fault Location Framework for Distribution Systems with DG using DSSim-PC (Celeita et al., 2014)
- Application of Reactive Energy to UK Low Voltage Networks and Embedded Generation Effects on Networks Performance (Mosaad, 2014)
- Monitoring of electrical power systems (Montenegro, Ramos, & Bacha, 2014)

General Conclusions

A-Diakoptics is an advanced methodology for simulating large power systems; covering the basic needs for the implementation of smart grid applications. This methodology can be used for making existing solution algorithms suitable for multithread processing. The challenges proposed in chapter 2 were addressed through A-Diakoptics as follows:

To guarantee deterministic computing times for RT-HIL simulation, this method separates and distributes the DS within a homogeneous multicore processing environment using standard computing architectures. However, the performance of the simulation will depend on the solution algorithm implemented on the child actors and the number of physical cores available. This method depends on the available hardware resources for guaranteeing performance; so, it is expected that the presented results can be improved using more computing hardware. For the cases presented in this paper.

Because the behavior of power devices is handled locally by the solving (child) actor, the simulation of sophisticated control devices interacting within the DS does not affect the performance of the simulation.

Due to the number of cells required for representing a link branch and the structure of the solution, topological changes can be addressed by modifying a lower number of positions in memory. With A-Diakoptics the DS can be separated in many parts as needed; this way, the solution will consider topological changes within the primitive networks without requiring additional calculations.

A-Diakoptics can be used with any solution method for solving the power flow analysis of DS. This way, experienced users of these methods can improve them for exploiting the existing computer architectures and leave behind the paradigm of sequential programing.

With A–Diakoptics several frequencies can be solved in the same simulation. This feature opens the door for simulating hybrid AC-DC systems and related cases.

However, the results achieved have demonstrated that A-Diakoptics is an effective method for improving the performance of the simulation when the size of the power system is medium/large-scale; when the size of the system enters in the small-scale, ,it will be better to using traditional methods or to solve the power system using only one child actor.

Another important factor when performing co-simulations and when working with heterogeneous computing systems is how the interfaces are used. Particularly, the access to ActiveX interfaces such as COM interfaces can be handled by using early and late bindings; this last one is the default use by many programming languages adding an important overhead; on the other hand, when using early bindings or equivalent connection methods the performance of the co-simulation can be as good as expected.

Additionally, this method has demonstrated to be suitable for addressing several managing activities within the ADA concept. The aim of A-Diakoptics is to improve existing methods for making suitable for heterogeneous parallel computing systems, where concurrent and asynchronous events that can be used for reducing the time required for estimating/calculating the behavior of the system in Real-Time. This is possible thanks to the technological evolution, which demands for the actual and future power engineers to complement the paradigm of sequential processing to start developing parallel processes in the many-cores computing era.

However, the tools generated in this thesis are used in different scenarios for developing and testing ADA solutions as presented above, delivering a flexible tool to perform smart grid studies using co-simulation within a parallel simulation architecture.

Only some features of the A-Diakoptics method are implemented in the DSSim-PC version available on the internet. The full features will be implemented in the source code of OpenDSS for making it available for the users of OpenDSS and DSSim-PC.

References

Abbas, A., & Ahmad, A. (2002, 16-17 Aug. 2002). *Object oriented parallel programming.* Paper presented at the Students Conference, 2002. ISCON '02. Proceedings. IEEE.

Abdel-Akher, M., Nor, K. M., & Rashid, A. H. A. (2005). Improved Three-Phase Power-Flow Methods Using Sequence Components. *IEEE Transactions on Power Systems, 20*(3), 1389-1397. doi: 10.1109/TPWRS.2005.851933

Abdulhadi, I., Coffele, F., Dysko, A., Booth, C., & Burt, G. (2011, 5-7 Dec. 2011). *Adaptive protection architecture for the smart grid.* Paper presented at the 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe), 2011

Apostolov, A. (2005, 6-9 June 2005). *Verification of models of distribution systems with distributed generation.* Paper presented at the 18th International Conference and Exhibition on Electricity Distribution, 2005. CIRED 2005. .

Areerak, P. S. K.-L. A. K.-N. (2011). Mathematical Model and Control Strategy on DQ Frame for Shunt Active Power Filters. *World Academy of Science, Engineering and Technology, 5*, 243-252.

Arritt, R. F., & Dugan, R. C. (2011). Distribution System Analysis and the Future Smart Grid. *IEEE Transactions on Industry Applications, 47*(6), 2343-2350. doi: 10.1109/TIA.2011.2168932

Agha, G., & Panwar, R. (1992, 23 Mar 1992). *An Actor-Based Framework for Heterogeneous Computing Systems.* Paper presented at the Workshop on Heterogeneous Processing, 1992. Proceedings.

Amestoy, P., Davis, T., & Duff, I. (1996). An Approximate Minimum Degree Ordering Algorithm. *SIAM Journal on Matrix Analysis and Applications, 17*(4), 886-905. doi: 10.1137/S0895479894278952

Ashcraft, C. (1995). Compressed Graphs and the Minimum Degree Algorithm. *SIAM Journal on Scientific Computing, 16*(6), 1404-1411. doi: doi:10.1137/0916081

Akbari, A., Setayeshmehr, A., Borsi, H., Gockenbach, E., & Fofana, I. (2010). Intelligent agent-based system using dissolved gas analysis to detect incipient faults in power transformers. *IEEE Electrical Insulation Magazine, 26*(6), 27-40. doi: 10.1109/MEI.2010.5599977

Armed Bear Common Lisp (ABCL). (2014, 04-19-2015). Retrieved 05-15, 2015, from https://common-lisp.net/project/armedbear/

Abur, A., & Expósito, A. G. (2004). *Power System State Estimation: Theory and Implementation*: Taylor & Francis.

Augustine, S., Mishra, M. K., & Lakshminarasamma, N. (2015). Adaptive Droop Control Strategy for Load Sharing and Circulating Current Minimization in Low-Voltage Standalone DC Microgrid. *IEEE Transactions on Sustainable Energy, 6*(1), 132-141. doi: 10.1109/TSTE.2014.2360628

Bacha Seddik, M. I., Bratcu Antoneta Iuliana. (2014). *Power Electronic Converters Modeling and Control* (1 ed.). http://www.springer.com/us/book/9781447154778#aboutBook: Springer-Verlag London.

Barney, B. (2014, 11/12/2014). Introduction to Parallel Computing. *Introduction to Parallel Computing.* Retrieved 04-30, 2015, from https://computing.llnl.gov/tutorials/parallel_comp/

Basu, K., Debusschere, V., Bacha, S., Maulik, U., & Bondyopadhyay, S. (2015). Nonintrusive Load Monitoring: A Temporal Multilabel Classification Approach. *IEEE Transactions on Industrial Informatics, 11*(1), 262-270. doi: 10.1109/TII.2014.2361288

Basu, K., Debusschere, V., Douzal-Chouakria, A., & Bacha, S. (2015). Time series distance-based methods for non-intrusive load monitoring in residential buildings. *Energy and Buildings, 96*(0), 109-117. doi: http://dx.doi.org/10.1016/j.enbuild.2015.03.021

Braham, A., Schneider, H., & Metz, M. (1997, 9-14 Nov 1997). *Recent developments in real-time analogue simulation of high power electrotechnical systems.* Paper presented at the 23rd International Conference on Industrial Electronics, Control and Instrumentation, 1997. IECON 97. .

Brant, A., & Lemieux, G. G. F. (2012, April 29 2012-May 1 2012). *ZUMA: An Open FPGA Overlay Architecture.* Paper presented at the 2012 IEEE 20th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).

Balamurugan, K., & Srinivasan, D. (2011, 5-8 Dec. 2011). *Review of power flow studies on distribution network with distributed generation.* Paper presented at the IEEE Ninth International Conference on Power Electronics and Drive Systems (PEDS), 2011.

Boemer, J. C., Gibescu, M., & Kling, W. L. (2009, June 28 2009-July 2 2009). *Dynamic models for transient stability analysis of transmission and distribution systems with distributed generation: An overview.* Paper presented at the PowerTech, 2009 IEEE Bucharest.

Brown, H. E., Carter, G. K., Happ, H. H., & Person, C. E. (1963). Power Flow Solution by Impedance Matrix Iterative Method. *IEEE Transactions on Power Apparatus and Systems, 82*(65), 1-10. doi: 10.1109/TPAS.1963.291392

Balakrishnan, R., & Ranganathan, K. (2000). *A Textbook of Graph Theory*: Springer.

Björck, A. (1996). *Numerical Methods for Least Squares Problems*: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104).

Baker, H. G. (1978). *Actor-Systems for Real-Time computation.* (PhD Doctoral Thesis), Massachusetts Institute of Technology M.I.T. (MIT/LCS/TR-197)

Bevrani, H., Daneshfar, F., & Hiyama, T. (2012). A New Intelligent Agent-Based AGC Design With Real-Time Application. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 42*(6), 994-1002. doi: 10.1109/TSMCC.2011.2175916

Bhattacharyya, S. S., Brooks, C., Cheong, E., John Davis, I., Goel, M., Kienhuis, B., . . . Zheng, H. (2004). Heterogeneous Concurrent Modeling and Design in Java *Ptolemy project, heterogeneous modeling design* (2 ed., Vol. 2, pp. 194). http://ptolemy.eecs.berkeley.edu/papers/04/ptIIDesignSoftware/ptIIdesign2-software.pdf: The Regents of the University of California.

Baran, M. (2012, 22-26 July 2012). *Branch current based state estimation for distribution system monitoring.* Paper presented at the Power and Energy Society General Meeting, 2012 IEEE.

Baran, M., & McDermott, T. E. (2009). *Distribution System State Estimation Using AMI Data.* Paper presented at the IEEE/PES Power Systems Conference and Exposition, Seattle, WA.

Baran, M. E., Hooshyar, H., Shen, Z., Gajda, J., & Huq, K. M. M. (2011, 24-29 July 2011). *Impact of high penetration residential PV systems on distribution systems.* Paper presented at the Power and Energy Society General Meeting, 2011 IEEE.

Baran, M. E., Jaesung, J., & McDermott, T. E. (2009, 26-30 Oct. 2009). *Topology error identification using branch current state estimation for distribution systems.* Paper presented at the Transmission & Distribution Conference & Exposition: Asia and Pacific, 2009.

Baran, M. E., Jinxiang, Z., & Kelley, A. W. (1996). Meter placement for real-time monitoring of distribution feeders. *IEEE Transactions on Power Systems, 11*(1), 332-337. doi: 10.1109/59.486114

Barrow, J. (2005). *Introducing Delphi Programming: Theory Through Practice*: Oxford University Press.

Caixue, L., Kaldewey, T., Povzner, A., & Brandt, S. A. (2006, Dec. 2006). *Diverse Soft Real-Time Processing in an Integrated System.* Paper presented at the 27th IEEE International Real-Time Systems Symposium, 2006. RTSS '06. .

Cogswell, J. (2015). Multicore vs. Vectorized: Programming Techniques Compared. *Go parallel, Transalating Multicore Power into Application Performance, 1*, 1. Retrieved from Go parallel website: http://goparallel.sourceforge.net/multicore-vs-vectorized-programming-techniques-compared/

Crăciun, O., Florescu, A., Munteanu, I., Bratcu, A. I., Bacha, S., & Radu, D. (2014). Hardware-in-the-loop simulation applied to protection devices testing. *International Journal of Electrical Power & Energy Systems, 54*(0), 55-64. doi: http://dx.doi.org/10.1016/j.ijepes.2013.06.031

Chang, G. W., Liu, Y. J., Dinavahi, V., & Ke, M. J. (2008). *Applications of Real-Time Simulation Techniques for Harmonics Study of An Industrial Power System.* Paper presented at the Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE, New York.

Cheng, C. S., & Shirmohammadi, D. (1995). A three-phase power flow method for real-time distribution system analysis. *IEEE Transactions on Power Systems, 10*(2), 671-679. doi: 10.1109/59.387902

Chang, G. W., Chu, S. Y., & Wang, H. L. (2007). An Improved Backward/Forward Sweep Load Flow Algorithm for Radial Distribution Systems. *IEEE Transactions on Power Systems, 22*(2), 882-884. doi: 10.1109/TPWRS.2007.894848

Chen, T.-H., & Yang, N.-C. (2010). Loop frame of reference based three-phase power flow for unbalanced radial distribution systems. *Electric Power Systems Research, 80*(7), 799-806. doi: http://dx.doi.org/10.1016/j.epsr.2009.12.006

Chen, T. H., Mo-Shing, C., Hwang, K. J., Kotas, P., & Chebli, E. A. (1991). Distribution system power flow analysis-a rigid approach. *IEEE Transactions on Power Delivery, 6*(3), 1146-1152. doi: 10.1109/61.85860

Cheng, C. S., & Shirmohammadi, D. (1995). A three-phase power flow method for real-time distribution system analysis. *IEEE Transactions on Power Systems, 10*(2), 671-679. doi: 10.1109/59.387902

Copeland, B. J. (2008). The Church-Turing Thesis. In E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy. In E. N. Zalta (Series Ed.) (Fall 2008 ed.). http://plato.stanford.edu/archives/fall2008/entries/church-turing. Retrieved from http://plato.stanford.edu/archives/fall2008/entries/church-turing.

Chun-Lien, S., Cong-Kai, L., Tso-Chu, C., & Ching-Jin, C. (2014, 20-23 May 2014). *Design of a multi-agent system for shipboard power systems restoration.* Paper presented at the 2014 IEEE/IAS 50th Industrial & Commercial Power Systems Tehcnical Conference (I&CPS).

Celeita, D., Zambrano, S., & Ramos, G. (2014, 10-13 Sept. 2014). *Fault location framework for distribution systems with DG using DSSim-PC.* Paper presented at the 2014 IEEE PES Transmission & Distribution Conference and Exposition - Latin America (PES T&D-LA).

Contreras, A. F., & Ramos, G. A. (2014, 14-17 April 2014). *Fault location algorithm based on shortest path optimization problem for Distribution networks with DG.* Paper presented at the 2014 IEEE PES T&D Conference and Exposition.

Converting a Desktop PC to a LabVIEW Real-Time Target. (2012, 21/05/2015). *LabVIEW Real-Time.* Retrieved 05-10, 2015, from http://www.ni.com/tutorial/2733/en/

Dargahi, M., Ghosh, A., Ledwich, G., & Zare, F. (2012, 16-19 Dec. 2012). *Studies in power hardware in the loop (PHIL) simulation using real-time digital simulator (RTDS).* Paper presented at the 2012 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES).

de Jong, P. M. A., de Gelder, N., Bussink, F. J. L., Verhoeven, R. P. M., & Mulder, M. (2013, 5-10 Oct. 2013). *Time and energy management during descent: Human vs automated response.* Paper presented at the IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC), 2013.

Deokar, S. A., Waghmare, L. M., & Takale, M. D. (2009, 4-6 June 2009). *Power system switching transients analysis using multiresolution signal decomposition.* Paper presented at the International Conference on Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009. 2009.

Diaz, J., Munoz-Caro, C., & Nino, A. (2012a). A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era. *IEEE Transactions on Parallel and Distributed Systems, 23*(8), 1369-1386. doi: 10.1109/TPDS.2011.308

Diaz, J., Munoz-Caro, C., & Nino, A. (2012b). A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era. *IEEE Transactions on Parallel and Distributed Systems, 23*(8), 1369-1386. doi: 10.1109/TPDS.2011.308

Djoudi, A., Chekireb, H., Berkouk, E., & Bacha, S. (2014, 24-25 Sept. 2014). *Real time estimation of DFIG inductances and rotor currents.* Paper presented at the 3rd Renewable Power Generation Conference (RPG 2014).

Djoudi, A., Chekireb, H., Berkouk, E. M., & Bacha, S. (2014, 25-27 March 2014). *Stability analysis of DFIG stator powers control based on sliding mode approach.* Paper presented at the 2014 Ninth International Conference on Ecological Vehicles and Renewable Energies (EVER).

Dufour, C., & Belanger, J. (2006, 23-26 May 2006). *Real-Time Simulation of Fuel Cell Hybrid Electric Vehicles.* Paper presented at the SPEEDAM 2006. International Symposium on Power Electronics, Electrical Drives, Automation and Motion, 2006. .

Dufour, C., Belanger, J., & Lapointe, V. (2008, 12-15 Oct. 2008). *FPGA-based Ultra-Low Latency HIL Fault Testing of a Permanent Magnet Motor Drive using RT-LAB-XSG.* Paper presented at the POWERCON 2008. Joint International Conference on Power System Technology and IEEE Power India Conference, 2008. .

Dufour, C., Hoang, L.-H., Soumagne, J. C., & El Hakimi, A. (1996). Real-time simulation of power transmission lines using Marti model with optimal fitting on dual-DSP card. *IEEE Transactions on Power Delivery, 11*(1), 412-419. doi: 10.1109/61.484041

Dugan, R. C., Arrit, R. F., Henry, R., McDermott, T. E., & Sunderm, W. (2014). OpenDSS EPRI Distribution System Simulator - Harmonic Load Modeling Documentation. Retrieved December, 2014, from http://sourceforge.net/projects/electricdss/files/

Dandachi, N. H., & Cory, B. J. (1991, 5-8 Nov 1991). *Network flow methods applied to power system problems-a survey paper.* Paper presented at the 1991 International Conference on Advances in Power System Control, Operation and Management, 1991. APSCOM-91.

Davis, T. A., & Natarajan, E. P. (2010). Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. *ACM Transactions on Mathematical Software, 37*(3), 36:31-36:17. doi: DOI: 10.1145/1824801.1824814

Dugan, R. C., Arritt, R. F., McDermott, T. E., Brahma, S. M., & Schneider, K. (2010, 25-29 July 2010). *Distribution System Analysis to support the Smart Grid.* Paper presented at the Power and Energy Society General Meeting, 2010 IEEE.

Dugan, R. C., & McDermott, T. E. (2011, 24-29 July 2011). *An open source platform for collaborating on smart grid research.* Paper presented at the 2011 IEEE Power and Energy Society General Meeting, .

Dasgupta, S., & Agnihotri, G. (2009, 28-29 Dec. 2009). *A Control Strategy for a VSC HVDC System in Steady State Response.* Paper presented at the ACT '09. International Conference on Advances in Computing, Control, & Telecommunication Technologies, 2009. .

Davis, T. A., & Natarajan, E. P. (2010). Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. *ACM Transactions on Mathematical Software, 37*(3), 36:31-36:17. doi: DOI: 10.1145/1824801.1824814

de Hon, B. P., & Arnold, J. M. (2010, 20-24 Sept. 2010). *Discrete Green's function diakoptics — A toy problem.* Paper presented at the 2010 International Conference on Electromagnetics in Advanced Applications (ICEAA).

Diestel, R. (2006). *Graph Theory*: Springer.

Davis, T. A., & Natarajan, E. P. (2010b). *Sparse Matrix Methods for Circuit Simulation Problems* (B. Michielsen & J.-R. Poirier Eds. 1 ed. Vol. 16). Berlin Heidelberg: Springer-Verlag.

Dedecker, J., Cutsem, T. V., Mostinckx, S., D, T., #39, Hondt, & Meuter, W. D. (2006). *Ambient-Oriented programming in ambienttalk*. Paper presented at the Proceedings of the 20th European conference on Object-Oriented Programming, Nantes, France.

Distribution Test Feeders. (2013). *Distribution Test Feeders.* Retrieved 05-19, 2015, from http://ewh.ieee.org/soc/pes/dsacom/testfeeders/

Delphi-Embarcadero Technologies Inc. (2015). Retrieved 03-04, 2015, from http://www.embarcadero.com/products/delphi

Dobakhshari, A. S., & Ranjbar, A. M. (2015). A Wide-Area Scheme for Power System Fault Location Incorporating Bad Data Detection. *IEEE Transactions on Power Delivery, 30*(2), 800-808. doi: 10.1109/TPWRD.2014.2352336

Electric Power Research Institute. (2013a). OpenDSS Dynamics Mode.pdf (file), supplied as part of the OpenDSS documentation. Retrieved from Sourceforqe website: http://sourceforge.net/projects/electricdss/

Electric Power Research Institute. (2013b). OpenDSS PVSystem Model.pdf (file), supplied as part of the OpenDSS documentation. Retrieved from Sourceforqe website: http://sourceforge.net/projects/electricdss/

Engblom, J. (2008). Heterogeneous vs homogeneous systems, revisited. *Observations from Uppsala. Computer Simulation, Virtual Platforms, Embedded Programming, Multicore and More, 1*(1), 3. Retrieved from Heterogeneous vs homogeneous systems, revisited website: http://jakob.engbloms.se/archives/90

Eyal-Salman, H., Seriai, A. D., & Dony, C. (2013, 14-16 Aug. 2013). *Feature-to-code traceability in a collection of software variants: Combining formal concept analysis and information retrieval.* Paper presented at the 2013 IEEE 14th International Conference on Information Reuse and Integration (IRI).

ePHASORsim, the future of real-time simulation begins here. (2014). *ePHASORsim.* Retrieved 02-03, 2015, from http://www.opal-rt.com/press-release/ephasorsim-future-real-time-simulation-begins-here

Fernandes, A. M., Pereira, R. C., Neto, A., Valcarcel, D. F., Alves, D., Sousa, J., . . . Varandas, C. A. F. (2012, 9-15 June 2012). *Real-time processing system for the JET hard X-ray and gamma-ray profile monitor enhancement.* Paper presented at the Real Time Conference (RT), 2012 18th IEEE-NPSS.

Fan, Z., Chen, Q., Kalogridis, G., Tan, S., & Kaleshi, D. (2012, 14-17 Oct. 2012). *The power of data: Data analytics for M2M and smart grid.* Paper presented at the 3rd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe), 2012.

Fuller, J., Kersting, W., Dugan, R., & Jr., S. C. (2013). Distribution Test Feeders. *IEEE PES Distribution System Analysis Subcommittee's.* Retrieved 10-23, 2013, from http://ewh.ieee.org/soc/pes/dsacom/testfeeders/

Gick, B., Gallenkamp, T., & Wess, T. (1996, 29 Apr-3 May 1996). *Introducing Digital Simulation Into An Analogue Real-time Power System Simulator.* Paper presented at the Sixth International Conference on (Conf. Publ. No. 423) AC and DC Power Transmission.

Gombert, C. (2005). *Simulation temps-réel des dispositifs d'Electronique de Puissance dédiés aux réseaux d'énergie électrique.* nstitut National Polytechnique de Grenoble- INPG, https://tel.archives-ouvertes.fr/tel-00171082. Retrieved from https://tel.archives-ouvertes.fr/tel-00171082 (HAL Id: tel-00171082)

Garcia, P. A. N., Pereira, J. L. R., Carneiro, S., Jr., da Costa, V. M., & Martins, N. (2000). Three-phase power flow calculations using the current injection method. *IEEE Transactions on Power Systems, 15*(2), 508-514. doi: 10.1109/59.867133

Ghosh, S., & Das, D. (1999). Method for load-flow solution of radial distribution networks. *IEE Proceedings- Generation, Transmission and Distribution, 146*(6), 641-648. doi: 10.1049/ip-gtd:19990464

Gupta, A., & George, T. (2010). Adaptive Techniques for Improving the Performance of Incomplete Factorization Preconditioning. *SIAM Journal on Scientific Computing, 32*(1), 84-110. doi: 10.1137/080727695

Giri, J., Parashar, M., Trehern, J., & Madani, V. (2012). The Situation Room: Control Center Analytics for Enhanced Situational Awareness. *Power and Energy Magazine, IEEE, 10*(5), 24-39. doi: 10.1109/MPE.2012.2205316

Gong, C., Peng, M., Gong, C., Zhao, Q., Tian, Z., Zhu, H., & Liu, Z. (2010, 28-31 March 2010). *Research on Digital Control System Simulation for Nuclear Power Plants.* Paper presented at the 2010 Asia-Pacific Power and Energy Engineering Conference (APPEEC).

Greer, R., Allen, W., Schnegg, J., & Dulmage, A. (2011, 10-13 April 2011). *Distribution automation systems with advanced features.* Paper presented at the Rural Electric Power Conference (REPC), 2011 IEEE.

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development. *Management Science, 46*(4), 451-466. doi: doi:10.1287/mnsc.46.4.451.12056

Hewitt, C. (1976). Viewing Control Structures as Patterns of Passing Messages (C. S. a. A. I. L. (CSAIL), Trans.) (1 ed., pp. 61). http://hdl.handle.net/1721.1/6272: Massachusetts Institute of Techonology

Hewitt, C. (2012, August 16-18, 2011). *Actor Model of Computation: Scalable Robust Information Systems.* Paper presented at the Inconsistency Robustness 2011, Stanford University.

Hewitt, C., & Barker, H. (1977). Actors and continuous functionals (M. I. T. A. I. Laboratory, Trans.). In M. I. T. A. I. Laboratory (Ed.), *M.I.T. Artificial Intelligence Laboratory* (1 ed., Vol. 1, pp. 36): Massachusestts Institute of Technology. Laboratory of computer science.

Happ, H. H. (1966). Orthogonal Networks. *IEEE Transactions on Power Apparatus and Systems, PAS-85*(3), 281-294. doi: 10.1109/TPAS.1966.291669

Happ, H. H. (1967). Z Diakoptics - Torn Subdivisions Radially Attached. *IEEE Transactions on Power Apparatus and Systems, PAS-86*(6), 751-769. doi: 10.1109/TPAS.1967.291887

Happ, H. H. (1971). *Diakoptics and networks* (A. Press Ed. 1 ed. Vol. 1): Elsevier Science.

Happ, H. H. (1980). *Piecewise Methods and Applications to Power Systems*: Wiley.

Instruments, N. (2013). Actor Framework Template documentation. Retrieved 24, 05, 2013, from http://www.ni.com/white-paper/14115/en

Hae-woo, P., Hanwoong, J., Hyunok, O., & Soonhoi, H. (2011). Library Support in an Actor-Based Parallel Programming Platform. *IEEE Transactions on Industrial Informatics, 7*(2), 340-353. doi: 10.1109/TII.2011.2123905

Haller, P., & Odersky, M. (2006). Event-Based Programming Without Inversion of Control. In D. Lightfoot & C. Szyperski (Eds.), *Modular Programming Languages* (Vol. 4228, pp. 4-22): Springer Berlin Heidelberg.

Hengxuan, L., Haishun, S., Jinyu, W., Shijie, C., & Haibo, H. (2012). A Fully Decentralized Multi-Agent System for Intelligent Restoration of Power Distribution Network Incorporating Distributed Generations [Application Notes]. *IEEE Computational Intelligence Magazine, 7*(4), 66-76. doi: 10.1109/MCI.2012.2215152

Hewitt, C., & Inman, J. (1991). DAI betwixt and between: from `intelligent agents' to open systems science. *IEEE Transactions on Systems, Man and Cybernetics, 21*(6), 1409-1419. doi: 10.1109/21.135685

Hewitt, C., Meijer, E., & Szyperski, C. (2012, 04-09-2012). The Actor Model (everything you wanted to know, but were afraid to ask). Retrieved 05-15, 2015, from http://channel9.msdn.com/Shows/Going+Deep/Hewitt-Meijer-and-Szyperski-The-Actor-Model-everything-you-wanted-to-know-but-were-afraid-to-ask

How Do I Call a Dynamic Link Library (DLL) from LabVIEW? (2007, 11/09/2014). 2. Retrieved 01/04, 2015, from How Do I Call a Dynamic Link Library (DLL) from LabVIEW?

Haibin, W., & Schulz, N. N. (2004). A revised branch current-based distribution system state estimation algorithm and meter placement impact. *IEEE Transactions on Power Systems, 19*(1), 207-213. doi: 10.1109/TPWRS.2003.821426

Imai, T., Iizuka, T., Makino, Y., & Horikoshi, T. (2002, 6-10 Oct. 2002). *Development and testing of multifunctional operation system for various power sources (simulation result using analogue power system simulator).* Paper presented at the Asia Pacific. IEEE/PES Transmission and Distribution Conference and Exhibition 2002.

Inoue, H., & Nakatani, T. (2010, 2-4 Dec. 2010). *Performance of multi-process and multi-thread processing on multi-core SMT processors.* Paper presented at the 2010 IEEE International Symposium on Workload Characterization (IISWC).

Ishchenko, A., Myrzik, J. M. A., & Kling, W. L. (2006, 6-8 Sept. 2006). *Transient Stability Analysis of Distribution Network with Dispersed Generation.* Paper presented at the Proceedings of the 41st International Universities Power Engineering Conference, 2006. UPEC '06.

Instruments, N. (2013). Actor Framework Template documentation. Retrieved 24, 05, 2013, from http://www.ni.com/white-paper/14115/en

Instruments, N. (2012). NI cRIO-9082 RT - Dual-Core Controller, LX150 FPGA with OS in Real Time. *NI cRIO-9082 RT.* Retrieved 20, 3, 2013, from http://sine.ni.com/nips/cds/view/p/lang/es/nid/210001

Jainschigg, J. (2012). Parallel Programing: Goals, Skills, Platforms, Markets, Languages. *Geeknet, 1*, 0. http://goparallel.sourceforge.net/wp-content/uploads/2013/05/Parallel-Programming-Goals-Skills-Platforms-Markets-Languages-10.16.12.pdf doi:10.16.12

Jalili-Marandi, V., Ayres, F. J., Ghahremani, E., Belanger, J., & Lapointe, V. (2013, 21-25 July 2013). *A real-time dynamic simulation tool for transmission and distribution power systems.* Paper presented at the 2013 IEEE Power and Energy Society General Meeting (PES).

Jang-Ping, S., & Chih-Yung, C. (1994, 19-22 Dec 1994). *Extracting multi-thread with data localities for vector computers.* Paper presented at the International Conference on Parallel and Distributed Systems, 1994.

Jen-Hao, T., & Chuo-Yean, C. (2002). A novel and fast three-phase load flow for unbalanced radial distribution systems. *IEEE Transactions on Power Systems, 17*(4), 1238-1244. doi: 10.1109/TPWRS.2002.805012

Jie, L., Eker, J., Janneck, J. W., Xiaojun, L., & Lee, E. A. (2004). Actor-oriented control system design: a responsible framework perspective. *IEEE Transactions on Control Systems Technology, 12*(2), 250-262. doi: 10.1109/TCST.2004.824310

Jerome, J. (2001, 2001). *Network observability and bad data processing algorithm for distribution networks.* Paper presented at the Power Engineering Society Summer Meeting, 2001.

Kopetz, H. (2011). *Real-Time Systems* (Second ed.). New York, NY 10013, USA: Springer Science+Business Media.

Kron, G. (1955). Detailed Example of Interconnecting Piece-Wise Solutions. *Journal of the Franklin Institute, 1*(1), 26. doi: 10.1016/0016-0032(55)90641-4

Kron, G. (1963). *Diakoptics: the piecewise solution of large-scale systems*: Macdonald.

Kamh, M., & Iravani, R. (2011, 24-29 July 2011). *A unified three-phase power-flow analysis model for electronically-coupled distributed energy resources.* Paper presented at the 2011 IEEE Power and Energy Society General Meeting.

Kamh, M. Z., & Iravani, R. (2010). Unbalanced Model and Power-Flow Analysis of Microgrids and Active Distribution Systems. *IEEE Transactions on Power Delivery, 25*(4), 2851-2858. doi: 10.1109/TPWRD.2010.2042825

Kersting, W. H. (2001). *Distribution System Modeling and Analysis*: CRC Press.

Kersting, W. H., & Dugan, R. C. (2006, Oct. 29 2006-Nov. 1 2006). *Recommended Practices for Distribution System Analysis.* Paper presented at the Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES.

Kuzmin, A., Luisier, M., & Schenk, O. (2013). Fast Methods for Computing Selected Elements of the Green's Function in Massively Parallel Nanoelectronic Device Simulations. In F. Wolf, B. Mohr & D. an Mey (Eds.), *Euro-Par 2013 Parallel Processing* (Vol. 8097, pp. 533-544): Springer Berlin Heidelberg.

Klos, A. (1982). What is diakoptics? *International Journal of Electrical Power & Energy Systems, 4*(3), 192-195. doi: http://dx.doi.org/10.1016/0142-0615(82)90049-7

Khotimah, P. H., Krisnandi, D., & Sugiarto, B. (2011, 20-21 Oct. 2011). *Design and implementation of Remote Terminal Unit on Mini Monitoring Weather Station Based on Microcontroller.* Paper presented at the 2011 6th International Conference on Telecommunication Systems, Services, and Applications (TSSA).

Kavousi-Fard, A., & Niknam, T. (2014). Optimal Distribution Feeder Reconfiguration for Reliability Improvement Considering Uncertainty. *IEEE Transactions on Power Delivery, 29*(3), 1344-1353. doi: 10.1109/TPWRD.2013.2292951

Kezunovic, M., Vittal, V., Meliopoulos, S., & Mount, T. (2012). The Big Picture: Smart Research for Large-Scale Integrated Smart Grid Solutions. *IEEE Power and Energy Magazine, 10*(4), 22-34. doi: 10.1109/MPE.2012.2196335

Khaitan, S. K., & Gupta, A. (2012). *High Performance Computing in Power and Energy Systems*: Springer Berlin Heidelberg.

Kouluri, M. K., & Pandey, R. K. (2011, 7-9 Sept. 2011). *Intelligent agent based micro grid control.* Paper presented at the 2011 2nd International Conference on Intelligent Agent and Multi-Agent Systems (IAMA).

Landry, S. J., & Pritchett, A. R. (2002, 27-31 Oct. 2002). *Displaying procedural vs. real-time information for paired approaches [ATC].* Paper presented at the The 21st Digital Avionics Systems Conference, 2002. Proceedings.

Laplante, P. A., & Ovaska, S. J. (2011). *Real-Time Systems Design and Analysis: Tools for the Practitioner*: Wiley.

Lucia, O., Urriza, I., Barraga, x, n, L. A., Navarro, D., . . . o, J. M. (2011). Real-Time FPGA-Based Hardware-in-the-Loop Simulation Test Bench Applied to Multiple-Output Power Converters. *IEEE Transactions on Industry Applications, 47*(2), 853-860. doi: 10.1109/TIA.2010.2102997

Lundstrom, B., Mather, B., Shirazi, M., & Coddington, M. (2013, 16-21 June 2013). *Implementation and validation of advanced unintentional islanding testing using power hardware-in-the-loop (PHIL) simulation.* Paper presented at the 2013 IEEE 39th Photovoltaic Specialists Conference (PVSC).

Lundstrom, B., Shirazi, M., Coddington, M., & Kroposki, B. (2013, 4-5 April 2013). *An Advanced Platform for Development and Evaluation of Grid Interconnection Systems Using Hardware-in-the-Loop: Part III -- Grid Interconnection System Evaluator.* Paper presented at the 2013 IEEE Green Technologies Conference.

Lo, K. L., & Zhang, C. (1993). Decomposed three-phase power flow solution using the sequence component frame. *IEE Proceedings Generation, Transmission and Distribution, 140*(3), 181-188.

Laddomada, M., Jovanovic Dolecek, G., Lim Yong, C., Luo, F.-L., Renfors, M., & Wanhammar, L. (2011). Advanced techniques on multirate signal processing for digital information processing [Editorial]. *Signal Processing, IET, 5*(3), 313-315. doi: 10.1049/iet-spr.2011.9058

Lei, T., Fang, Y., & Xianyong, F. (2013, 21-25 July 2013). *A novel method for distribution system feeder reconfiguration using black-box optimization.* Paper presented at the 2013 IEEE Power and Energy Society General Meeting (PES).

McDermott, T. E., & Dugan, R. C. (2003). PQ, reliability and DG. *IEEE Industry Applications Magazine, 9*(5), 17-23. doi: 10.1109/MIA.2003.1227867

Mekhtoub, S., Ibtiouen, R., Touhami, O., & Bacha, S. (2007, 10-12 Sept. 2007). *Analysis of transient currents of the connected self- excited induction generator following disturbance on the system supply.* Paper presented at the International Aegean Conference on Electrical Machines and Power Electronics, 2007. ACEMP '07.

Missaoui, R., Warkozek, G., Bacha, S., & Ploix, S. (2012, 14-17 Oct. 2012). *Real time validation of an optimization Building Energy Management strategy based on Power-Hardware-in-the-loop tool.* Paper presented at the 2012 3rd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies (ISGT Europe).

Montenegro, D., Hernandez, M. E., & Ramos, G. A. (2015). A realistic generator of power quality disturbances for practicing in courses of electrical engineering. *Computer Applications in Engineering Education, 23*(3), 391-402. doi: 10.1002/cae.21609

Montenegro, D., & Ramos, G. A. (2012, 3-5 Sept. 2012). *Smart Diagnosis of power quality disturbances using Bayesian networks.* Paper presented at the Transmission and Distribution: Latin America Conference and Exposition (T&D-LA), 2012 Sixth IEEE/PES.

Mo-Shing, C., & Tsai-Hsiang, C. (1991, 5-8 Nov 1991). *Application of three-phase load flow to power system distribution automation.* Paper presented at the APSCOM-91., 1991 International Conference on Advances in Power System Control, Operation and Management, 1991. .

Moghaddas-Tafreshi, S. M., & Mashhour, E. (2009). Distributed generation modeling for power flow studies and a three-phase unbalanced power flow solution for radial distribution systems considering distributed generation. *Electric Power Systems Research, 79*(4), 680-686. doi: http://dx.doi.org/10.1016/j.epsr.2008.10.003

Montenegro, D. (2013). DSSim-PC, Electrical Distribution Network simulator for PC. Retrieved 01, 07, 2013, from http://sourceforge.net/projects/dssimpc/?source=navbar

Maheshwarapu, S. (1998, 1998). *An effective root based algorithm for power system topological observability.* Paper presented at the TENCON '98. 1998 IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control.

Markowitz, H. M. (1957). The Elimination Form of the Inverse and Its Application to Linear Programming. *Management Science, 3*(3), 255-269. doi: 10.2307/2627454

Montenegro, D. (2013). TCP Communication Protocol For Data Exchange With The Dssim-XX System, provided as part fo the documentation of DSSim-RT/PC. Retrieved from DSSim-PC|SourceForge.net website: http://sourceforge.net/projects/dssimpc/

Montenegro, D., Ramos, G. A., & Bacha, S. (2014, 14-17 April 2014). *Distributed programming for high performance monitoring of electrical power systems.* Paper presented at the 2014 IEEE PES T&D Conference and Exposition.

Mrozowski, M., Niedzwiecki, M., & Suchomski, P. (1995). A fast recursive highly dispersive absorbing boundary condition using time domain diakoptics and Laguerre polynomials. *IEEE Microwave and Guided Wave Letters, 5*(6), 183-185. doi: 10.1109/75.386125

Muller, S. C., Hager, U., & Rehtanz, C. (2014). A Multiagent System for Adaptive Power Flow Control in Electrical Transmission Systems. *IEEE Transactions on Industrial Informatics, 10*(4), 2290-2299. doi: 10.1109/TII.2014.2315499

Men-Shen, T., & Fu-Yuan, H. (2010). Application of Grey Correlation Analysis in Evolutionary Programming for Distribution System Feeder Reconfiguration. *IEEE Transactions on Power Systems, 25*(2), 1126-1133. doi: 10.1109/TPWRS.2009.2032325

Mitra, P., Lidong, Z., & Harnefors, L. (2014). Offshore Wind Integration to a Weak Grid by VSC-HVDC Links Using Power-Synchronization Control: A Case Study. *IEEE Transactions on Power Delivery, 29*(1), 453-461. doi: 10.1109/TPWRD.2013.2273979

Monticelli, A. (1999). *State estimation in electric power systems: a generalized approach*: Kluwer Academic Publishers.

Mosaad, R. A. (2014). *Application of Reactive Energy to UK Low Voltage Networks and Embedded Generation Effects on Networks Performance.* (Laurea Magistrale in Ingegneria Elettrica), Università degli Studi di Padova, http://tesi.cab.unipd.it/47542/1/Master_Thesis_-_Remon_Atfy_Mosaad.pdf. Retrieved from http://tesi.cab.unipd.it/47542/1/Master_Thesis_-_Remon_Atfy_Mosaad.pdf (47542)

Nagel, I., Fabre, L., Cherkaoui, R., & Kayal, M. (2010, 24-26 June 2010). *High-speed power system stability simulation using analog computation: Systematic error analysis.* Paper presented at the 2010 Proceedings of the 17th International Conference Mixed Design of Integrated Circuits and Systems (MIXDES).

Nagarajan, A., & Ayyanar, R. (2014, 7-9 Sept. 2014). *Dynamic analysis of distribution systems with high penetration of PV generators using differential algebraic equations in OpenDSS.* Paper presented at the North American Power Symposium (NAPS), 2014.

Ning, Z., Chongqing, K., Kirschen, D. S., Qing, X., Weimin, X., Junhui, H., & Qian, Z. (2013). Planning Pumped Storage Capacity for Wind Power Integration. *IEEE Transactions on Sustainable Energy, 4*(2), 393-401. doi: 10.1109/TSTE.2012.2226067

Nasri, M., Ginn, H. L., & Moallem, M. (2014, 14-18 Sept. 2014). *Application of intelligent agent systems for real-time coordination of power converters (RCPC) in microgrids.* Paper presented at the 2014 IEEE Energy Conversion Congress and Exposition (ECCE).

Ni, B., & Xiao, D. (2010). Identification of non-uniformly sampled multirate systems with application to process data compression. *Control Theory & Applications, IET, 4*(6), 970-984. doi: 10.1049/iet-cta.2008.0570

Narimani, M. R., Vahed, A. A., Azizipanah-Abarghooee, R., & Javidsharifi, M. (2014). Enhanced gravitational search algorithm for multi-objective distribution feeder reconfiguration considering reliability, loss and operational cost. *IET Generation, Transmission & Distribution, 8*(1), 55-69. doi: 10.1049/iet-gtd.2013.0117

NI Single-Board RIO Out Of the Box. (2012). *NI Single-Board RIO.* Retrieved 04-02, 2015, from http://www.ni.com/video/726/en/

Northcote-Green, J., & Wilson, R. G. (2006). *Control and Automation of Electrical Power Distribution Systems*: CRC Press.

Nunoo, S., & Ofei, A. K. (2010, 27-29 Sept. 2010). *Distribution automation (DA) using supervisory control and data acquisition (SCADA) with advanced metering infrastructure (AMI).* Paper presented at the 2010 IEEE Conference on Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES).

Ney, M. M., & Le Maguer, S. (2002, 10-13 Sept. 2002). *The transmission-line matrix (TLM) method: an efficient tool for problem segmentation (diakoptics).* Paper presented at the 2002 International Conference on Mathematical Methods in Electromagnetic Theory.

Ocnasu, D. (2008). *Modélisation, Commande et Simulation Temps-Réel Hybride des Systémes de Génération Non Conventionnels.* Université Joseph-Fourier - Grenoble I, https://tel.archives-ouvertes.fr/tel-00339664. (HAL Id: tel-00339664)

Olobaniyi, F., Nouri, H., & Ghauri, S. (2012, 4-7 Sept. 2012). *Investigation of Diakoptics as a resourceful tool in power system analysis.* Paper presented at the 2012 47th International Universities Power Engineering Conference (UPEC).

Palensky, P., & Dietrich, D. (2011). Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads. *IEEE Transactions on Industrial Informatics, 7*(3), 381-388. doi: 10.1109/TII.2011.2158841

Proakis, J. G., & Manolakis, D. G. (1995). *Digital Signal Processing*. New Jersey, U.S.A.: Prentice-Hall, Inc.

Ptolemaeus, C. (2013). *System Design, Modeling, and Simulation Using Ptolemy II*: Ptolemy.org.

Personal Energy Management. (2013). *Landis+Gyr Ltd.* Retrieved 8 March, 2013, from http://www.landisgyr.com/za/en/pub/solutions/personal_energy_management.cfm

Powalko, M., Rudion, K., Komarnicki, P., & Blumschein, J. (2009, 8-11 June 2009). *Observability of the distribution system.* Paper presented at the 20th International Conference and Exhibition on Electricity Distribution - Part 1, 2009. CIRED 2009.

Radil, T., Ramos, P. M., Janeiro, F. M., & Serra, A. C. (2008). PQ Monitoring System for Real-Time Detection and Classification of Disturbances in a Single-Phase Power System. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, 57*(8), 1725 - 1733

Ramos, G. A., & Montenegro, D. (2012). Pattern Recognition of Power Quality Disturbances Based on Continuous Wavelet Transform. *International Review on Modelling and Simulations, 5*(1), 24-29.

Ren, W., Sloderbeck, M., Steurer, M., Dinavahi, V., Noda, T., Filizadeh, S., . . . Martinez, J. A. (2011). Interfacing Issues in Real-Time Digital Simulators. *IEEE Transactions on Power Delivery, 26*(2), 1221-1230. doi: 10.1109/TPWRD.2010.2072792

Rogersten, R., Vanfretti, L., Wei, L., Lidong, Z., & Mitra, P. (2014, 12-15 Oct. 2014). *A quantitative method for the assessment of VSC-HVdc controller simulations in EMT tools.* Paper presented at the 2014 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe).

Roitman, M., Watanabe, E. H., & Lyra, F. J. (1989). Power systems analog simulation enhancement using a new programmable electronic model. *IEEE Transactions on Power Systems, 4*(1), 286-292. doi: 10.1109/59.32490

Rajičić, D., & Taleski, R. (1998). Two novel methods for radial and weakly meshed network analysis. *Electric Power Systems Research, 48*(2), 79-87. doi: http://dx.doi.org/10.1016/S0378-7796(98)00067-4

Rogerson, D. (1997). *Inside COM*: Microsoft Press.

Ramos, G. A., Rios, M. A., & Rincon, C. A. (2014, 10-13 Sept. 2014). *Connection schemes for Electric Vehicle Supply Equipment.* Paper presented at the Latin America (PES T&D-LA), 2014 IEEE PES Transmission & Distribution Conference and Exposition.

Sankarakrishnan, A., & Billinton, R. (1995). Sequential Monte Carlo simulation for composite power system reliability analysis with time varying loads. *IEEE Transactions on Power Systems, 10*(3), 1540-1545. doi: 10.1109/59.466491

Schlesinger, R. (2010). *Developing Real World Software*: Jones & Bartlett Learning.

Shull, F., Rus, I., & Basili, V. (2000). How perspective-based reading can improve requirements inspections. *Computer, 33*(7), 73-79. doi: 10.1109/2.869376

Soni, M. (2015). Defect Prevention: Reducing Costs and Enhancing Quality. *Software/IT.* 3. Retrieved 10-04, 2015, from http://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality/

Steurer, M., Bogdan, F., Ren, W., Sloderbeck, M., & Woodruff, S. (2007, 24-28 June 2007). *Controller and Power Hardware-In-Loop Methods for Accelerating Renewable Energy Integration.* Paper presented at the IEEE Power Engineering Society General Meeting, 2007.

Stott, B. (1974). Review of load-flow calculation methods. *Proceedings of the IEEE, 62*(7), 916-929. doi: 10.1109/PROC.1974.9544

Stagg, G. W., & El-Abiad, A. H. (1968). *Computer methods in power system analysis*: McGraw-Hill.

Stewart, E. M., Aukai, T. P., MacPherson, S. D. J., Quach, B. P., Nakafuji, D., & Davis, R. (2012, 22-26 July 2012). *A realistic irradiance-based voltage flicker analysis of PV applied to Hawaii distribution feeders.* Paper presented at the 2012 IEEE Power and Energy Society General Meeting.

Shahidehpour, M., & Wang, Y. (2004). *Communication and Control in Electric Power Systems: Applications of Parallel and Distributed Processing*: Wiley.

Tan, L. (2007). *Digital Signal Processing: Fundamentals and Applications*: Elsevier Science.

Taraft, S., Rekioua, D., Aouzellag, D., & Bacha, S. (2015). A proposed strategy for power optimization of a wind energy conversion system connected to the grid. *Energy Conversion and Management, 101*(0), 489-502. doi: http://dx.doi.org/10.1016/j.enconman.2015.05.047

Thiaux, Y., Dang, T. T., Multon, B., Ben Ahmed, H., & Tran, Q. T. (2015, 17-19 March 2015). *Demand side management in PV/diesel stand-alone system with real-time Monte Carlo simulation of the consumer electrical behaviour.* Paper presented at the 2015 IEEE International Conference on Industrial Technology (ICIT).

Tianshu, B., Hongwei, Z., Peng, D., & Qixun, Y. (2008, 17-20 March 2008). *Transient Stability Analysis of Asynchronous Distribution Generation and its Impact on Protection.* Paper presented at the IET 9th International Conference on Developments in Power System Protection, 2008. DPSP 2008.

Teofili, S., Di Mascolo, M., Bianchi, G., Salsano, S., & Zugenmaier, A. (2008, 25-27 Aug. 2008). *User plane security alternatives in the 3G evolved Multimedia Broadcast Multicast Service (e-MBMS).* Paper presented at the Third International Conference on Communications and Networking in China, 2008. ChinaCom 2008. .

Tutorial: State Machines- National Instruments. (2015). *Tutorials NI LabVIEW.* Retrieved 05-19, 2015, from http://www.ni.com/tutorial/7595/en/

Tinney, W. F., & Walker, J. W. (1967). Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE, 55*(11), 1801-1809. doi: 10.1109/PROC.1967.6011

Ustun, T. S., Ozansoy, C., & Zayegh, A. (2011, 25-28 Sept. 2011). *Distributed Energy Resources (DER) object modeling with IEC 61850–7–420.* Paper presented at the 2011 21st Australasian Universities Power Engineering Conference (AUPEC).

Uriarte, F. M., & Butler-Purry, K. L. (2006, 17-19 Sept. 2006). *Diakoptics in Shipboard Power System Simulation.* Paper presented at the 38th North American Power Symposium, 2006. NAPS 2006.

Uluski, R. W. (2010, 25-29 July 2010). *The role of Advanced Distribution Automation in the Smart Grid.* Paper presented at the Power and Energy Society General Meeting, 2010 IEEE.

Van-Linh, N., Tuan, T.-Q., Bacha, S., & Be, N. (2014, Oct. 29 2014-Nov. 1 2014). *Charging strategies to minimize the peak load for an electric vehicle fleet.* Paper presented at the 40th Annual Conference of the IEEE Industrial Electronics Society, IECON 2014.

Varela, C., & Agha, G. (2001). Programming dynamically reconfigurable open systems with SALSA. *SIGPLAN Not., 36*(12), 20-34. doi: 10.1145/583960.583964

Vasquez, J. C., Guerrero, J. M., Miret, J., Castilla, M., de, V., x00F, & a, L. G. (2010). Hierarchical Control of Intelligent Microgrids. *Industrial Electronics Magazine, IEEE, 4*(4), 23-29. doi: 10.1109/MIE.2010.938720

Wang, X., Guo, C., Xiao, X., & Zhao, C. (2010, 20-22 Sept. 2010). *Research of analysis method of power system real-time digital simulation error.* Paper presented at the 2010 5th International Conference on Critical Infrastructure (CRIS).

Watson, N., Arrillaga, J., & Engineers, I. o. E. (2003). *Power Systems Electromagnetic Transients Simulation*: Institution of Engineering and Technology.

Wan Jusoh, W. N. S. E., Mat Hanafiah, M. A., Raman, S. H., & Ghani, M. R. A. (2013, 18-20 Nov. 2013). *Development of a new modeling circuit for the Remote Terminal Unit (RTU) with GSM communication.* Paper presented at the 2013 IEEE Conference on Clean Energy and Technology (CEAT).

Wu-Chang, W., Men-Shen, T., & Fu-Yuan, H. (2007, 5-8 Nov. 2007). *A New Binary Coding Particle Swarm Optimization for Feeder Reconfiguration.* Paper presented at the International Conference on Intelligent Systems Applications to Power Systems, 2007. ISAP 2007.

Xi, L., Gole, A. M., & Ming, Y. (2009). A Wide-Band Multi-Port System Equivalent for Real-Time Digital Power System Simulators. *IEEE Transactions on Power Systems, 24*(1), 237-249. doi: 10.1109/TPWRS.2008.2007000

Xuan, L., & Bin, S. (2008, 10-13 Dec. 2008). *Microgrids - an integration of renewable energy technologies.* Paper presented at the China International Conference on Electricity Distribution, 2008. CICED 2008. .

Yi, Z., Gole, A. M., Wenchuan, W., Boming, Z., & Hongbin, S. (2013). Development and Analysis of Applicability of a Hybrid Transient Simulation Platform Combining TSA and EMT Elements. *IEEE Transactions on Power Systems, 28*(1), 357-366. doi: 10.1109/TPWRS.2012.2196450

Yin, X., Haixiang, G., Ying, C., & Chen-Ching, L. (2014, 27-31 July 2014). *A fast EMT simulation method for control and protection studies of microgrids.* Paper presented at the 2014 IEEE PES General Meeting | Conference & Exposition.

Yizhong, H., Wenchuan, W., Boming, Z., & Qi, G. (2013, 6-9 Oct. 2013). *Development of an RTDS-TSA hybrid transient simulation platform with frequency dependent network equivalents.* Paper presented at the 2013 4th IEEE/PES Innovative Smart Grid Technologies Europe (ISGT EUROPE).

Yousefpoor, N., Parkhideh, B., Azidehak, A., & Bhattacharya, S. (2014, 14-18 Sept. 2014). *Convertible static transmission controller (CSTC) system model validation by controller hardware-in-the-loop-simulation.* Paper presented at the 2014 IEEE Energy Conversion Congress and Exposition (ECCE).

Yuhang, D., Hui, L., & Foo, S. (2009, 7-10 Sept. 2009). *Controller Hardware-In-the-Loop simulation for design of power management strategies for fuel cell vehicle with energy storage.* Paper presented at the IEEE Vehicle Power and Propulsion Conference, 2009. VPPC '09.

Yuhe, Z., Wenjia, W., Kobayashi, Y., & Shirai, K. (2012, 4-8 March 2012). *Remaining driving range estimation of electric vehicle.* Paper presented at the 2012 IEEE International Electric Vehicle Conference (IEVC).

Yildirim, E., Arslan, E., Kim, J., & Kosar, T. (2015). Application-Level Optimization of Big Data Transfers Through Pipelining, Parallelism and Concurrency. *IEEE Transactions on Cloud Computing, PP*(99), 1-1. doi: 10.1109/TCC.2015.2415804

Yildirim, E., JangYoung, K., & Kosar, T. (2012, 10-16 Nov. 2012). *How GridFTP Pipelining, Parallelism and Concurrency Work: A Guide for Optimizing Large Dataset Transfers.* Paper presented at the 2012 SC Companion: High Performance Computing, Networking, Storage and Analysis (SCC).

Yanfeng, G., & Guzman, A. (2013). Integrated Fault Location System for Power Distribution Feeders. *IEEE Transactions on Industry Applications, 49*(3), 1071-1078. doi: 10.1109/TIA.2013.2252596

Yannakakis, M. (1981). Computing the Minimum Fill-In is NP-Complete. *SIAM Journal on Algebraic Discrete Methods, 2*(1), 77-79. doi: 10.1137/0602010

Zavoda, F. (2008, 10-13 Dec. 2008). *The key role of intelligent electronic devices (IED) in advanced Distribution Automation (ADA).* Paper presented at the China International Conference on Electricity Distribution, 2008. CICED 2008. .

Zavoda, F. (2010). *Advanced Distribution Automation (ADA) Applications and Power Quality in Smart Grids.* Paper presented at the China International Conference on Electricity Distribution, Shanghai.

Zhang, Y., & Zhao, R. (2010, 17-19 Sept. 2010). *An open64-based cost analytical model in auto-vectorization.* Paper presented at the 2010 International Conference on Educational and Information Technology (ICEIT).

Zhong, F., Kulkarni, P., Gormus, S., Efthymiou, C., Kalogridis, G., Sooriyabandara, M., . . . Woon Hau, C. (2013). Smart Grid Communications: Overview of Research Challenges, Solutions, and Standardization Activities. *Communications Surveys & Tutorials, IEEE, 15*(1), 21-38. doi: 10.1109/SURV.2011.122211.00021

Zhu, Y.-y., Dong, P., Hu, T., & Xie, G.-p. (2013, 9-11 Sept. 2013). *The digital-analog hybrid simulation system of large scale wind and thermo electric power transmission with HVDC link.* Paper presented at the 2nd IET Renewable Power Generation Conference (RPG 2013).

Zhu, Y., & Tomsovic, K. (2002). Adaptive power flow method for distribution systems with dispersed generation. *IEEE Transactions on Power Delivery, 17*(3), 822-827. doi: 10.1109/TPWRD.2002.1022810

Zahedi, A. (2011, 25-28 March 2011). *Smart Grid Opportunities & Challenges for Power Industry to Manage the Grid More Efficiently.* Paper presented at the Power and Energy Engineering Conference (APPEEC), 2011 Asia-Pacific.

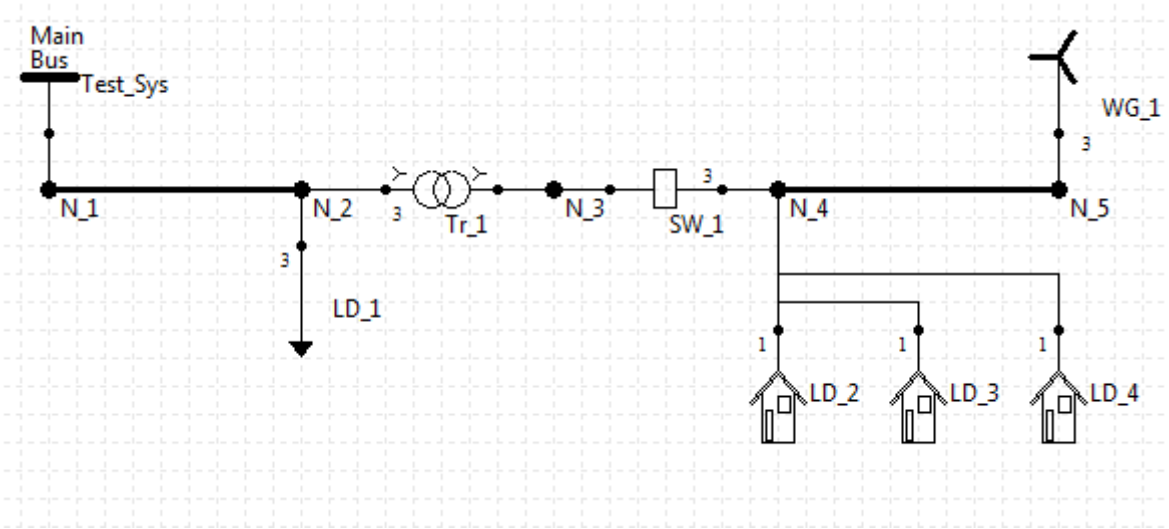
Zhongwei, S., & Jing, M. (2010, 24-26 Sept. 2010). *Efficient key management for advanced distribution automation system.* Paper presented at the 2nd IEEE International Conference on Network Infrastructure and Digital Content, 2010

Annexes and Complementary material

APPLICATION EXAMPLE OF A-DIAKOPTICS

SEPARATION AND SOLUTION ALGORITHM

Consider the following system:



This system includes a distributed generator (WG_1) en the final node. The Y_{BUS} matrix that describes this system is as shown in TABLE I.

TABLE I
Y Matrix of the example system

n1.1	n1.2	n1.3	n2.1	n2.2	n2.3	n4.1	n4.2	n4.3	n3.1	n3.2	n3.3	n5.1	n5.2	n5.3	
1.1e+2-3.7e+2i	0.4+31.1i	0.4+31.1i	-10+10i	0	0	0	0	0	0	0	0	0	0	0	n1.1
0.4+31.1i	1.1e+2-3.7e+2i	0.4+31.1i	0	-10+10i	0	0	0	0	0	0	0	0	0	0	n1.2
0.4+31.1i	0.4+31.1i	1.1e+2-3.7e+2i	0	0	-10+10i	0	0	0	0	0	0	0	0	0	n1.3
-10+10i	0	0	10.5-14.9i	0	0	0	0	0	-3.2+56.5i	0	0	0	0	0	n2.1
0	-10+10i	0	0	10.5-14.9i	0	0	0	0	0	-3.2+56.5i	0	0	0	0	n2.2
0	0	-10+10i	0	0	10.5-14.9i	0	0	0	0	0	-3.2+56.5i	0	0	0	n2.3
0	0	0	0	0	0	1e+7-5e+2i	0	0	-1e+7	0	0	-5e+2+5e+2	0	0	n4.1
0	0	0	0	0	0	0	1e+7-5e+2i	0	0	-1e+7	0	0	-5e+2+5e+2	0	n4.2
0	0	0	0	0	0	0	0	1e+7-5e+2i	0	0	-1e+7	0	0	-5e+2+5e+2	n4.3
0	0	0	-3.2+56.5i	0	0	-1e+7	0	0	1e+7-6.5e+2i	0	0	0	0	0	n3.1
0	0	0	0	-3.2+56.5i	0	0	-1e+7	0	0	1e+7-6.5e+2i	0	0	0	0	n3.2
0	0	0	0	0	-3.2+56.5i	0	0	-1e+7	0	0	1e+7-6.5e+2i	0	0	0	n3.3
0	0	0	0	0	0	-5e+2+5e+2i	0	0	0	0	0	5e+2-5e+2i	0	0	n5.1
0	0	0	0	0	0	0	-5e+2+5e+2i	0	0	0	0	0	5e+2-5e+2i	0	n5.2
0	0	0	0	0	0	0	0	-5e+2+5e+2i	0	0	0	0	0	5e+2-5e+2i	n5.3

This matrix can be separated in two independent sub-systems as shown in TABLE II. These sub-systems compose the matrix Z_{TT}. Neglect the link branches.

TABLE II

Z_{TT} for the example system (Fig. 1)

n1.1	n1.2	n1.3	n2.1	n2.2	n2.3	n3.1	n3.2	n3.3	n4.1	n4.2	n4.3	n5.1	n5.2	n5.3	
7.46e-4+ 2.64e-3i	1.38e+4+ 2.1e-4i	1.38e+4+ 2.1e-4i	7.62e-4+ 2.61e-3i	1.38e-4+ 2.07e-4i	1.38e-4+ 2.07e-4i	6.67e-5+ 2.28e-4i	1.21e- 51.81e-5i	1.21e- 51.81e-5i	0	0	0	0	0	0	n1.1
1.38e+4+ 2.1e-4i	7.46e-4+ 2.64e-3i	1.38e+4+ 2.1e-4i	1.38e-4+ 2.07e-4i	7.62e-4+ 2.61e-3i	1.38e-4+ 2.07e-4i	1.21e- 51.81e-5i	6.67e-5+ 2.28e-4i	1.21e- 51.81e-5i	0	0	0	0	0	0	n1.2
1.38e+4+ 2.1e-4i	1.38e+4+ 2.1e-4i	7.46e-4+ 2.64e-3i	1.38e-4+ 2.07e-4i	1.38e-4+ 2.07e-4i	7.62e-4+ 2.61e-3i	1.21e- 51.81e-5i	1.21e- 51.81e-5i	6.67e-5+ 2.28e-4i	0	0	0	0	0	0	n1.3
7.62e-4+ 2.61e-3i	1.38e-4+ 2.07e-4i	1.38e-4+ 2.07e-4i	5.08e-2+ 5.17e-2i	1.39e-4+ 2.04e-4i	1.39e-4+ 2.04e-4i	4.44e-3+ 4.53e-3i	1.22e-5+ 1.79e-5i	1.22e-5+ 1.79e-5i	0	0	0	0	0	0	n2.1
1.38e-4+ 2.07e-4i	7.62e-4+ 2.61e-3i	1.38e-4+ 2.07e-4i	1.39e-4+ 2.04e-4i	5.08e-2+ 5.17e-2i	1.39e-4+ 2.04e-4i	1.22e-5+ 1.79e-5i	4.44e-3+ 4.53e-3i	1.22e-5+ 1.79e-5i	0	0	0	0	0	0	n2.2
1.38e-4+ 2.07e-4i	1.38e-4+ 2.07e-4i	7.62e-4+ 2.61e-3i	1.39e-4+ 2.04e-4i	1.39e-4+ 2.04e-4i	5.08e-2+ 5.17e-2i	1.22e-5+ 1.79e-5i	1.22e-5+ 1.79e-5i	4.44e-3+ 4.53e-3i	0	0	0	0	0	0	n2.3
1.21e-5+ 1.81e-5i	1.21e-5+ 1.81e-5i	1.21e-5+ 1.81e-5i	4.44e-3+ 4.53e-3i	1.22e-5+ 1.79e-5i	1.22e-5+ 1.79e-5i	4.77e-4+ 1.94e-3i	1.06e-6+ 1.56e-6i	1.06e-6+ 1.56e-6i	0	0	0	0	0	0	n3.1
1.21e-5+ 1.81e-5i	1.21e-5+ 1.81e-5i	1.21e-5+ 1.81e-5i	1.22e-5+ 1.79e-5i	4.44e-3+ 4.53e-3i	1.22e-5+ 1.79e-5i	1.06e-6+ 1.56e-6i	4.77e-4+ 1.94e-3i	1.06e-6+ 1.56e-6i	0	0	0	0	0	0	n3.2
1.21e-5+ 1.81e-5i	1.21e-5+ 1.81e-5i	1.21e-5+ 1.81e-5i	1.22e-5+ 1.79e-5i	1.22e-5+ 1.79e-5i	4.44e-3+ 4.53e-3i	1.06e-6+ 1.56e-6i	1.06e-6+ 1.56e-6i	4.77e-4+ 1.94e-3i	0	0	0	0	0	0	n3.3
0	0	0	0	0	0	0	0	0	-1.57+ 2.89e-2i	0	0	-1.58+ 2.4e-2i	0	0	n4.1
0	0	0	0	0	0	0	0	0	0	-1.57+ 2.89e-2i	0	0	-1.58+ 2.4e-2i	0	n4.2
0	0	0	0	0	0	0	0	0	0	0	-1.57+ 2.89e-2i	0	0	-1.58+ 2.4e-2i	n4.3
0	0	0	0	0	0	0	0	0	-1.58+ 2.36e-2i	0	0	-1.58+ 1.93e-2i	0	0	n5.1
0	0	0	0	0	0	0	0	0	0	-1.58+ 2.36e-2i	0	0	-1.58+ 1.9e-2i	0	n5.2
0	0	0	0	0	0	0	0	0	0	0	-1.58+ 2.36e-2i	0	0	-1.58+ 1.9e-2i	n5.3

The diagonal link matrix is shown in TABLE III, to obtain this matrix an impedance of 1e-7 ohms have been assumed for SW_1. When the switch opens, all the values of this matrix become 1e+7.

TABLE III

Z_{LLD} FOR DESCRIBING SW_1

C1	C2	C3	
1e-7	0	0	C1
0	1e-7	0	C2
0	0	1e-7	C3

With Z_{TT} and Z_{LLD}, Z_{CC} can be calculated. Z_{CC} is shown in TABLE IV.

TABLE IV

Z_{CC}

C1	C2	C3	
-3.38e-1+ 5.51e-1i	-3.58e-1+ 2.49e-1i	-3.58e-1+ 2.49e-1i	C1
-3.58e-1+ 2.49e-1i	-3.38e-1+ 5.51e-1i	-3.58e-1+ 2.49e-1i	C2
-3.58e-1+ 2.49e-1i	-3.58e-1+ 2.49e-1i	-3.38e-1+ 5.51e-1i	C3

After solving each system independently they deliver the currents and voltages presented in TABLE V.

TABLE V
RESULTS DELIVERED FOR THE SUB-SYSTEMS

node	Voltage L	Current C_L	node	Voltage R	Current C_R
n1.1	1.2e+3+ 6.9e+2i	3.8e+5 -4e+5i	n4.1	9.5e+1+ 7.9e+1i	1.2e+1 +9.6i
n1.2	-5.9e-1- 1.4e+3i	-5.4e+5i -1.3e+5i	n4.2	2.1e+1- 1.2e+2i	2.5-1.5e+1i
n1.3	-1.2e+3+ 6.9e+2i	1.5e+5i +5.3e+5i	n4.3	-1.2e+2+ 4.3e+1i	-1.4e+1 +5.2i
n2.1	1.2e+3+ 6.8e+2i	3.7-2.1i	n5.1	9.5e+1+ 8e+1i	2.7e+2 -3.1e+2i
n2.2	-1.3e+1- 1.4e+3i	4e-2+ 4.3i	n5.2	2.1e+1- 1.2e+2i	-4e+2 -7.9e+1i
n2.3	-1.2e+3+ 7e+2i	3.7-2.2i	n5.3	-1.2e+2+ 4.3e+1i	1.3e+2 +3.9e+2i
n3.1	1e+2+ 5.9e+1i	0			
n3.2	-1.1-1.2e+2i	0			
n3.3	-1e+2+ 6.1e+1i	0			

The letter R or L that goes with the voltage and current titles refers to the left or right sub-system. The excitation current vector I_0 is the concatenation of the currents C_L and C_R.
The matrix Z_{CT} transposed and Z_{TC} are shown in TABLE VI.

TABLE VI
COMPLEMENTARY MATRIXES Z_{CT}* AND Z_T

	Z_{CT}*			Z_{TC}		
	c1	c2	c3	c1	c2	c3
n1.	-6.7e-5+	-1.2e-5+	-1.2e-5+	-6.7e-5-	-1.2e-5-	-1.2e-5-
1	2.3e-4i	1.8e-5i	1.8e-5i	2.3e-4i	1.8e-5i	1.8e-5i
n1.	-1.2e-5+	-6.75e-5+	-1.2e-5+	-1.2e-5-	-6.75e-5-	-1.2e-5-
2	1.8e-5i	2.3e-4i	1.8e-5i	1.8e-5i	2.3e-4i	1.8e-5i
n1.	-1.2e-5+	-1.2e-5+	-6.75e-5+	-1.2e-5-	-1.2e-5-	-6.75e-5-
3	1.8e-5i	1.8e-5i	2.3e-4i	1.8e-5i	1.8e-5i	2.3e-4i
n2.	-4.4e-3+	-1.2e-5+	-1.2e-5+	-4.4e-3-	-1.2e-5-	-1.2e-5-
1	4.5e-3i	1.8e-5i	1.8e-5i	4.5e-3i	1.8e-5i	1.8e-5i
n2.	-1.2e-5+	-4.4e-3+	-1.2e-5+	-1.2e-5-	-4.4e-3-	-1.2e-5-
2	1.8e-5i	4.5e-3i	1.8e-5i	1.8e-5i	4.5e-3i	1.8e-5i
n2.	-1.2e-5+	-1.2e-5+	-4.4e-3+	-1.2e-5-	-1.2e-5-	-4.4e-3-
3	1.8e-5i	1.8e-5i	4.5e-3i	1.8e-5i	1.8e-5i	4.5e-3i
n3.	-4.8e-4+	-1.1e-6+	-1.1e-6+	-4.8e-4-	-1.1e-6-	-1.1e-6-
1	1.9e-3i	1.6e-6i	1.6e-6i	1.9e-3i	1.6e-6i	1.6e-6i
n3.	-1.1e-6+	-4.8e-4+	-1.1e-6+	-1.1e-6-	-4.8e-4-	-1.1e-6-
2	1.6e-6i	1.9e-3i	1.6e-6i	1.6e-6i	1.9e-3i	1.6e-6i
n3.	-1.1e-6+	-1.1e-6+	-4.8e-4+	-1.1e-6-	-1.1e-6-	-4.8e-4-
3	1.6e-6i	1.6e-6i	1.9e-3i	1.6e-6i	1.6e-6i	1.9e-3i
n4.	-3.4e-1-	-3.6e-1-	-3.6e-1-	-3.4e-1+	-3.6e-1+	-3.6e-1+
1	5.5e-1i	2.5e-1i	2.5e-1i	5.5e-1i	2.5e-1i	2.5e-1i
n4.	-3.6e-1-	-3.4e-1-	-3.6e-1-	-3.6e-1+	-3.4e-1+	-3.6e-1+
2	2.5e-1i	5.5e-1i	2.5e-1i	2.5e-1i	5.5e-1i	2.5e-1i
n4.	-3.6e-1-	-3.6e-1-	-3.4e-1-	-3.6e-1+	-3.6e-1+	-3.4e-1+
3	2.5e-1i	2.5e-1i	5.5e-1i	2.5e-1i	2.5e-1i	5.5e-1i
n5.	-3.4e-1-	-3.6e-1-	-3.6e-1-	-3.4e-1+	-3.6e-1+	-3.6e-1+
1	5.5e-1i	2.5e-1i	2.5e-1i	5.5e-1i	2.5e-1i	2.5e-1i
n5.	-3.6e-1-	-3.4e-1-	-3.6e-1-	-3.6e-1+	-3.4e-1+	-3.6e-1+
2	2.5e-1i	5.5e-1i	2.5e-1i	2.5e-1i	5.5e-1i	2.5e-1i
n5.	-3.6e-1-	-3.4e-1-	-3.4e-1-	-3.6e-1+	-3.4e-1+	-3.4e-1+
3	2.5e-1i	5.5e-1i	5.5e-1i	2.5e-1i	5.5e-1i	5.5e-1i

Following the algorithm in TABLE I step 2 $e_c'=-Z_{CT}I_0$.

$$e_c' = [9.169 - 20.1138i \quad -22.0064 + 2.11549i \quad 12.833 + 18i]$$

Step 3 $i_c=(Z_{CC})^{-1}\, e_c'$.

$$i_c = [-64.126 - 34.7i \quad 2 + 72.887i \quad 62.119 - 3818i]$$

Steps 4 and 5, calculate E_1 and E_T.

TABLE VII
RESULTS AND ERROR WITH CLASSICAL SOLUTION METHOD

node	E_0	E_1	E_T	Error with OpenDSS
n1.1	1.2e+3+ 6.9e+2i	-3.8e-3+ 1.5e-2i	1200.16+ 692.25i	1e-12
n1.2	-5.9e-1- 1.4e+3i	1.5e-2- 4.4e-3i	-0.57- 1385.49i	1e-12
n1.3	-1.2e+3+ 6.9e+2i	-1.1e-2- 1.1e.2i	-1199.58+ 693.242i	1e-12
n2.1	1.2e+3+ 6.8e+2i	1.3e-1+ 4.4e-1	1195.7+ 676.14i	1e-12
n2.2	-1.3e+1- 1.4e+3i	3.2e-1- 3.3e-1i	-12.293- 1373.57i	1e-12
n2.3	-1.2e+3+ 7e+2i	-4.5e-1- 1.1e-1i	1183.4+ 697.4i	1e-12
n3.1	1e+2+ 5.9e+1i	-3.7e-2+ 1.4e-1i	104.575+ 59.264i	1e-12
n3.2	-1.1-1.2e+2i	1.4e-1- 3.9e-2i	-0.96-120.197i	1e-12
n3.3	-1e+2+ 6.1e+1i	-1e-1- 1e-1i	-103.612+ 60.93i	1e-12
n4.1	9.5e+1+ 7.9e+1i	9.1-2i	104.6+ 59.25i	1e-12
n4.2	2.1e+1- 1.2e+2i	-2.2e+1 +2.1i	-0.9737- 120.197i	1e-12
n4.3	-1.2e+2+ 4.3e+1i	1.3e+1+ 1.8e+1i	-103.6+ 60.943i	1e-12
n5.1	9.5e+1+ 8e+1i	9.2-2e+1i	104.7+ 59.79i	1e-12
n5.2	2.1e+1- 1.2e+2i	-2.2e+1+ 1.9i	-0.589-120.6i	1e-12
n5.3	-1.2e+2+ 4.3e+1i	1.3e+1+ 1.8e+1i	-104.145+ 60.81i	1e-12

EPRI'S CIRCUIT 7

GRAPHICAL REPRESENTATION, 1 LAYER USING DSSIM-PC/RT

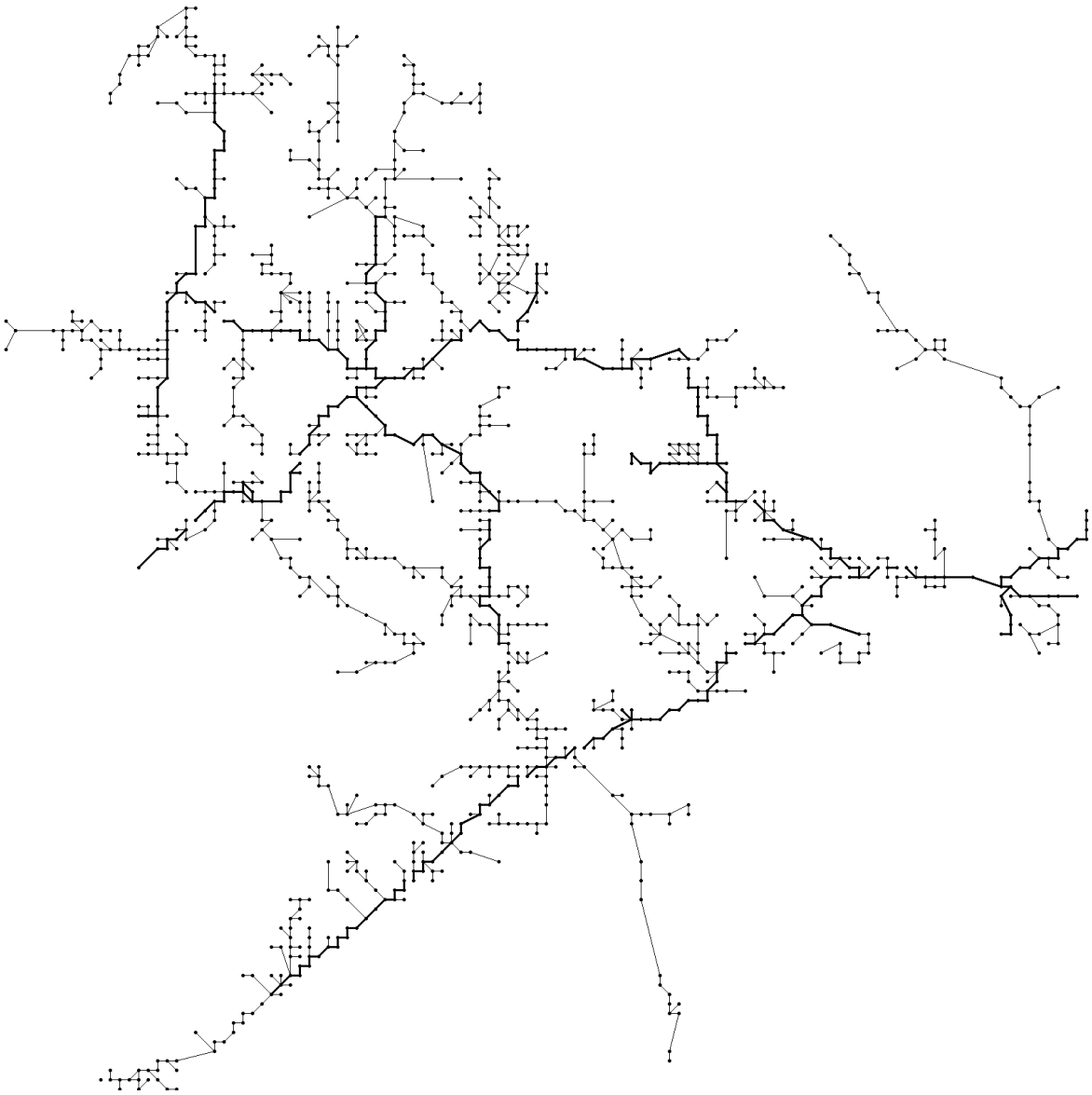
Voltages (kV) calculated by OpenDSS and DSSim-PC/RT for EPRI's circuit 7 (27 Buses randomly selected)

		OpenDSS				DSSim-PC/RT		
Bus	Node	VLN	Angle	pu	Base	VLN	Angle	Error
SOURCEBUS	1	67.96	-3.7	1.0236	115	67.959412	-3.7	8.64957E-06
	2	67.82	-123.8	1.0215	115	67.819614	-123.8	5.6857E-06
	3	67.776	116.4	1.0208	115	67.775939	116.4	8.97068E-07
CKT7	1	7.3576	-33.7	1.0219	12.47	7.3575818	-33.7	2.47924E-06
	2	7.3621	-153.9	1.0226	12.47	7.3620889	-153.9	1.50266E-06
	3	7.3421	86.2	1.0198	12.47	7.3420444	86.2	7.57152E-06
318405	1	7.3576	-33.7	1.0219	12.47	7.357552	-33.7	6.52915E-06
	2	7.3621	-153.9	1.0226	12.47	7.3620334	-153.9	9.04475E-06
	3	7.3421	86.2	1.0198	12.47	7.3420429	86.2	7.7703E-06
318412	1	7.3576	-33.7	1.0219	12.47	7.3575939	-33.7	8.32267E-07
	2	7.3621	-153.9	1.0226	12.47	7.3620889	-153.9	1.50234E-06
	3	7.3421	86.2	1.0198	12.47	7.3420661	86.2	4.62063E-06
318414	1	7.3576	-33.7	1.0219	12.47	7.3575289	-33.7	9.6662E-06
	2	7.3621	-153.9	1.0226	12.47	7.3620475	-153.9	7.13448E-06
	3	7.3421	86.2	1.0198	12.47	7.3420349	86.2	8.86841E-06
157345	1	7.3483	-33.8	1.0207	12.47	7.348248	-33.8	7.0802E-06
	2	7.3578	-154	1.022	12.47	7.3577477	-154	7.10649E-06
	3	7.334	86.1	1.0187	12.47	7.3339592	86.1	5.55972E-06
157346	1	7.3483	-33.8	1.0207	12.47	7.3482806	-33.8	2.63761E-06
	2	7.3578	-154	1.022	12.47	7.3577294	-154	9.59678E-06
	3	7.334	86.1	1.0187	12.47	7.3339614	86.1	5.26576E-06
165442	1	7.3481	-33.8	1.0206	12.47	7.3480664	-33.8	4.57177E-06
	2	7.3577	-154	1.022	12.47	7.3576665	-154	4.55671E-06
	3	7.3338	86.1	1.0186	12.47	7.3337408	86.1	8.0671E-06
298160	1	7.3395	-34	1.0194	12.47	7.3394845	-34	2.11392E-06
	2	7.3526	-154.1	1.0213	12.47	7.3525777	-154.1	3.03432E-06
	3	7.3232	85.9	1.0172	12.47	7.3231776	85.9	3.06536E-06
165448	1	7.3334	-34.1	1.0186	12.47	7.3333843	-34.1	2.13493E-06
	2	7.3484	-154.2	1.0207	12.47	7.3483279	-154.2	9.80722E-06
	3	7.3174	85.9	1.0164	12.47	7.31736	85.9	5.47089E-06
275354	1	7.3213	-34.3	1.0169	12.47	7.3212576	-34.3	5.78839E-06
	2	7.3416	-154.4	1.0197	12.47	7.3415478	-154.4	7.11589E-06
	3	7.3023	85.6	1.0143	12.47	7.3022988	85.6	1.687E-07
165449	1	7.3203	-34.4	1.0168	12.47	7.3202688	-34.4	4.26521E-06
	2	7.3411	-154.4	1.0197	12.47	7.3410824	-154.4	2.39272E-06
	3	7.3011	85.6	1.0141	12.47	7.30103	85.6	9.58303E-06
165450	1	7.3203	-34.4	1.0168	12.47	7.3202326	-34.4	9.20171E-06
	2	7.3411	-154.4	1.0197	12.47	7.3410984	-154.4	2.14006E-07
	3	7.3011	85.6	1.0141	12.47	7.3010873	85.6	1.73667E-06
165453	1	7.3162	-34.4	1.0162	12.47	7.3161785	-34.4	2.94064E-06
	2	7.3382	-154.4	1.0193	12.47	7.3381437	-154.4	7.67387E-06
	3	7.2962	85.6	1.0134	12.47	7.2961436	85.6	7.72632E-06

165454	1	7.3157	-34.4	1.0161	12.47	7.315663	-34.4	5.06003E-06
	2	7.3376	-154.4	1.0192	12.47	7.3375347	-154.4	8.90346E-06
	3	7.2956	85.6	1.0133	12.47	7.2955737	85.6	3.60707E-06
165451	1	7.3172	-34.4	1.0163	12.47	7.3171891	-34.4	1.48938E-06
	2	7.3394	-154.4	1.0194	12.47	7.339336	-154.4	8.71969E-06
	3	7.2971	85.6	1.0135	12.47	7.297096	85.6	5.41868E-07
238402	1	7.3153	-34.4	1.0161	12.47	7.3152722	-34.4	3.80121E-06
	2	7.3373	-154.4	1.0191	12.47	7.3372487	-154.4	6.99146E-06
	3	7.2953	85.6	1.0133	12.47	7.2952898	85.6	1.39157E-06
165455	1	7.3127	-34.4	1.0157	12.47	7.3126496	-34.4	6.89647E-06
	2	7.3347	-154.4	1.0188	12.47	7.3346868	-154.4	1.79523E-06
	3	7.2927	85.6	1.0129	12.47	7.2926683	85.6	4.3536E-06
165456	1	7.3127	-34.4	1.0157	12.47	7.3126842	-34.4	2.15398E-06
	2	7.3347	-154.4	1.0188	12.47	7.3346571	-154.4	5.844E-06
	3	7.2927	85.6	1.0129	12.47	7.2926789	85.6	2.89178E-06
165452	1	7.3137	-34.4	1.0159	12.47	7.3136457	-34.4	7.42866E-06
	2	7.3357	-154.4	1.0189	12.47	7.3356491	-154.4	6.93381E-06
	3	7.2937	85.6	1.0131	12.47	7.2936527	85.6	6.49063E-06
283563	1	7.3129	-34.4	1.0157	12.47	7.3128873	-34.4	1.73792E-06
	2	7.3349	-154.4	1.0188	12.47	7.3348291	-154.4	9.66896E-06
	3	7.2929	85.6	1.013	12.47	7.2928758	85.6	3.31221E-06
165457	1	7.3126	-34.4	1.0157	12.47	7.3125471	-34.4	7.22958E-06
	2	7.3346	-154.4	1.0188	12.47	7.334599	-154.4	1.39025E-07
	3	7.2926	85.6	1.0129	12.47	7.2925811	85.6	2.58937E-06
165458	1	7.3163	-34.4	1.0162	12.47	7.3162369	-34.4	8.62641E-06
	2	7.3385	-154.4	1.0193	12.47	7.3384458	-154.4	7.38751E-06
	3	7.2962	85.6	1.0134	12.47	7.2961379	85.6	8.51812E-06
165459	1	7.3163	-34.4	1.0162	12.47	7.3162382	-34.4	8.44528E-06
	2	7.3385	-154.4	1.0193	12.47	7.3384276	-154.4	9.87244E-06
	3	7.2963	85.6	1.0134	12.47	7.2962438	85.6	7.69901E-06
S3X_1000569	1	0.1182	-65.8	0.9843	0.208	0.1181999	-65.8	9.32046E-07
	2	0.1188	174.2	0.98894	0.208	0.1187592	174.2	6.47388E-06
	3	0.1186	54	0.98793	0.208	0.1186395	54	3.96313E-06
X_1000746	1	0.1192	-65.9	0.99279	0.208	0.1192195	-65.9	4.27175E-06
	2	0.1197	174.2	0.99674	0.208	0.1196994	174.2	5.11747E-06
	3	0.1196	54	0.9955	0.208	0.119549	54	8.13335E-06
S1X_1000746	1	0.1182	-65.8	0.98398	0.208	0.1181592	-65.8	6.61951E-06
	2	0.1187	174.2	0.98858	0.208	0.1187199	174.2	9.82758E-07
	3	0.1186	54	0.98759	0.208	0.118599	54	8.57795E-06

IEEE 8500 Nodes test system

GRAPHICAL REPRESENTATION, 2 LAYERS USING DSSIM-PC/RT



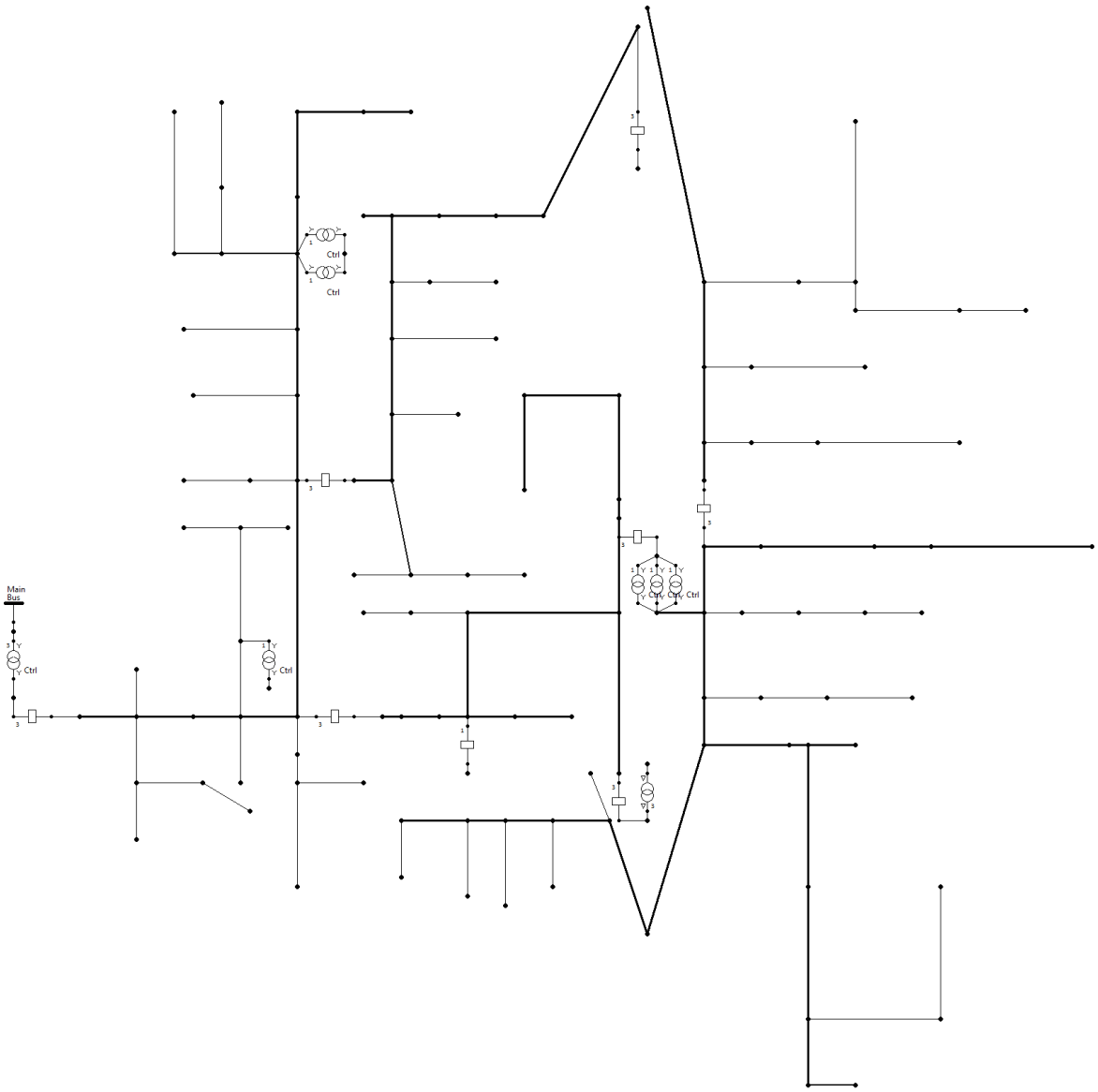
Voltages (kV) calculated by OpenDSS and DSSim-PC/RT for IEEE 8500 nodes (25 Buses randomly selected)

		OpenDSS				DSSim-PC/RT		
Bus	Node	VLN	Angle	pu	Base	VLN	Angle	Error
SOURCEBUS	1	69.715	-120	1.05	115	69.714643	-120	5.12082E-06
	2	69.715	120	1.05	115	69.71499353	120	9.28315E-08
	3	69.715	-34.5	1.05	115	69.71488657	-34.5	1.62705E-06
_HVMV_SUB_LSB	1	7.5521	-154.3	1.049	12.47	7.552053924	-154.3	6.10104E-06
	2	7.5387	85.9	1.0471	12.47	7.538668108	85.9	4.23039E-06
	3	7.5591	-34.5	1.0499	12.47	7.559054479	-34.5	6.022E-06
HVMV_SUB_48332	1	7.5521	-154.3	1.049	12.47	7.552061158	-154.3	5.14316E-06

	2	7.5387	85.9	1.0471	12.47	7.538685445	85.9	1.9307E-06
	3	7.5591	-44.9	1.0499	12.47	7.559050698	-44.9	6.52222E-06
M1009763	1	7.2335	-167.2	1.0047	12.47	7.23349829	-167.2	2.36417E-07
	2	7.2987	79.5	1.0138	12.47	7.298631416	79.5	9.3967E-06
	3	7.3635	-167.2	1.0228	12.47	7.36349723	-167.2	3.76212E-07
L2673322	2	7.2986	80.4	1.0138	12.47	7.298537665	80.4	8.54074E-06
M1069148	3	7.4153	80.4	1.03	12.47	7.415244167	80.4	7.52939E-06
L2673309	3	7.415	-165.4	1.0299	12.47	7.414929911	-165.4	9.45234E-06
M1069588	2	7.2517	-165.4	1.0072	12.47	7.251634173	-165.4	9.07745E-06
L2804270	2	7.2517	-167	1.0072	12.47	7.251688145	-167	1.63482E-06
M3036164	2	7.1669	-167	0.99546	12.47	7.166897002	-167	4.18244E-07
L2935553	2	7.1654	-156.5	0.99526	12.47	7.165341556	-156.5	8.15647E-06
M1209749	2	7.3757	-156.5	1.0245	12.47	7.375693413	-156.5	8.93111E-07
L2748840	2	7.3754	-160.5	1.0244	12.47	7.375339325	-160.5	8.22664E-06
M1108264	2	7.4157	-160.5	1.03	12.47	7.41565851	-160.5	5.59484E-06
M1108263	2	7.4153	-44.9	1.03	12.47	7.415296933	-44.9	4.13646E-07
L3085398	1	7.28	-167.1	1.0112	12.47	7.279977456	-167.1	3.0967E-06
	2	7.3427	79.5	1.0199	12.47	7.342642403	79.5	7.84415E-06
	3	7.3749	-44.9	1.0243	12.47	7.374850332	-44.9	6.73468E-06
M1026891	1	7.2768	-167.1	1.0107	12.47	7.27676022	-167.1	5.46675E-06
	2	7.3363	79.5	1.019	12.47	7.336262372	79.5	5.129E-06
	3	7.3737	-44.4	1.0242	12.47	7.373658963	-44.4	5.56534E-06
L3216367	1	7.3504	-166.7	1.0209	12.47	7.350393174	-166.7	9.28663E-07
	2	7.3053	79.2	1.0147	12.47	7.305274319	79.2	3.51534E-06
	3	7.3111	-44.4	1.0155	12.47	7.311063869	-44.4	4.94195E-06
M1069468	1	7.3497	-166.7	1.0209	12.47	7.349649303	-166.7	6.89777E-06
	2	7.3031	79.1	1.0144	12.47	7.303034791	79.1	8.92897E-06
	3	7.3103	-159	1.0154	12.47	7.310257571	-159	5.80397E-06
M1142800	2	7.1968	-159	0.99962	12.47	7.19678941	-159	1.47143E-06
X2970258C	1	0.12029	-101.6	1.0017	0.208	0.12028974	-101.6	2.15745E-06
	2	0.12036	-167.4	1.0022	0.208	0.12035985	-167.4	1.24942E-06
X2748124B	1	0.11794	12.6	0.98213	0.208	0.117939027	12.6	8.25035E-06
	2	0.11794	-39.3	0.98207	0.208	0.117939632	-39.3	3.12044E-06
X2879773A	1	0.11869	140.7	0.98838	0.208	0.118689611	140.7	3.27757E-06
	2	0.11879	80	0.98918	0.208	0.118789373	80	5.28035E-06
X2841626C	1	0.12238	-99.9	1.019	0.208	0.122378978	-99.9	8.35425E-06
	2	0.12258	-44.6	1.0208	0.208	0.122578892	-44.6	9.03967E-06
X2991905A	1	0.12159	135.4	1.0125	0.208	0.121589854	135.4	1.19695E-06
	2	0.12147	-44.6	1.0115	0.208	0.121469499	-44.6	4.12408E-06

IEEE 123 Nodes test system

GRAPHICAL REPRESENTATION, 2 LAYERS USING DSSIM-PC/RT



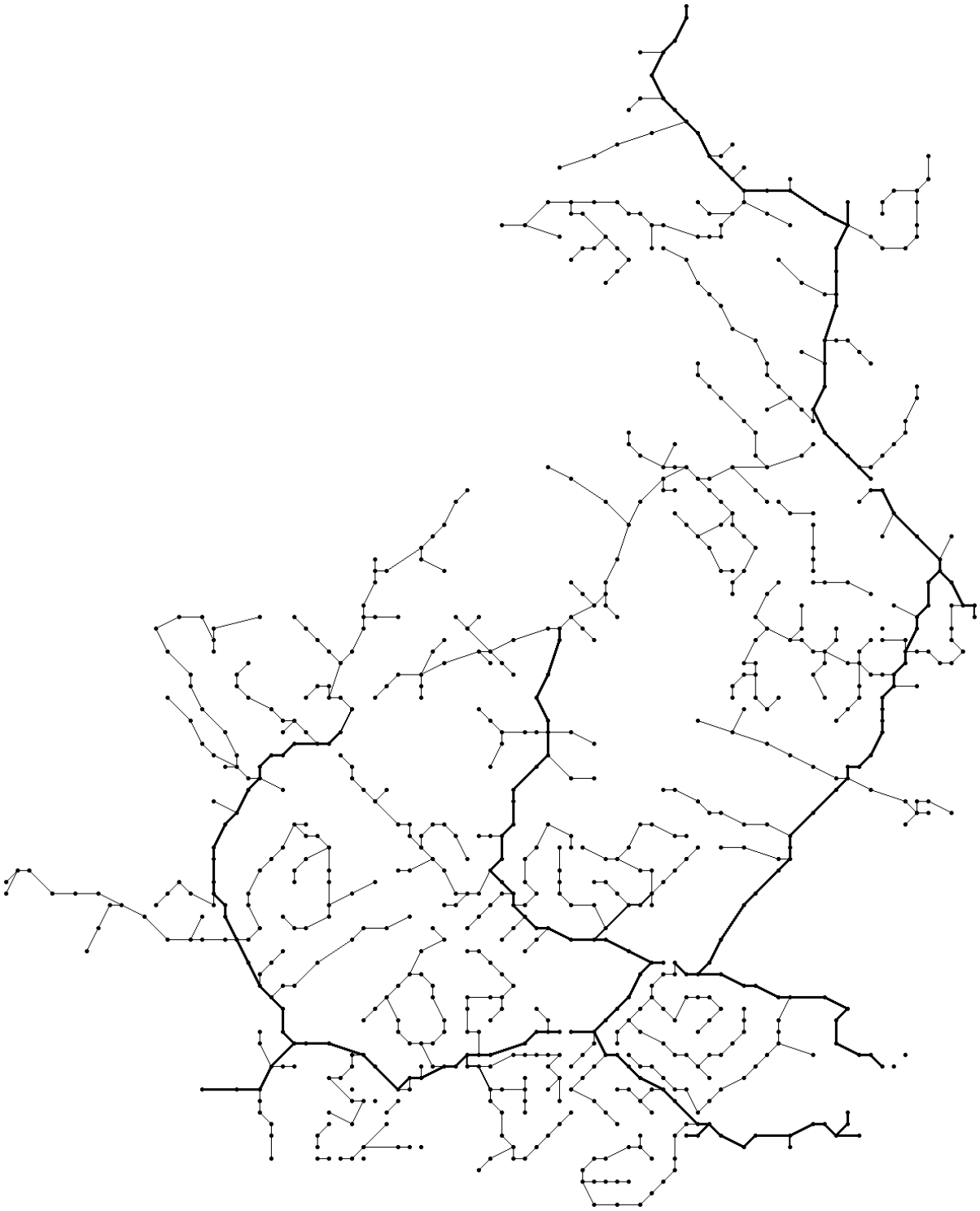
Voltages (kV) calculated by OpenDSS and DSSim-PC/RT for IEEE 123 nodes (26 Buses randomly selected)

			OpenDSS			DSSim-PC/RT		
Bus	Node	VLN	Angle	pu	Base	VLN	Angle	Error
150	1	2.4018	0	0.99999	4.16	2.401792508	0	3.11944E-06
	2	2.4018	-120	0.99999	4.16	2.401777548	-120	9.34792E-06
	3	2.4018	120	0.99999	4.16	2.401799303	120	2.90124E-07
150R	1	2.5068	0	1.0437	4.16	2.506789259	0	4.28457E-06
	2	2.5068	-120	1.0437	4.16	2.506793657	-120	2.53052E-06
	3	2.5068	120	1.0437	4.16	2.506784234	120	6.28918E-06

149	1	2.506	0	1.0434	4.16	2.505989707	0	4.10749E-06
	2	2.5063	-120	1.0435	4.16	2.50627559	-120	9.73943E-06
	3	2.5062	120	1.0435	4.16	2.506191118	120	3.54391E-06
1	1	2.4761	-0.6	1.0309	4.16	2.476084154	-0.6	6.39972E-06
	2	2.5004	-120.3	1.0411	4.16	2.500392875	-120.3	2.84962E-06
	3	2.4849	119.6	1.0346	4.16	2.484880796	119.6	7.72846E-06
2	2	2.4999	-120.3	1.0408	4.16	2.499892796	-120.3	2.88175E-06
3	3	2.4809	119.6	1.033	4.16	2.480891795	119.6	3.30712E-06
7	1	2.4538	-1.1	1.0216	4.16	2.453784391	-1.1	6.36135E-06
	2	2.4962	-120.6	1.0393	4.16	2.496193761	-120.6	2.49948E-06
	3	2.4712	119.4	1.0289	4.16	2.471182209	119.4	7.19923E-06
4	3	2.4797	119.6	1.0325	4.16	2.479675591	119.6	9.84359E-06
5	3	2.4778	119.6	1.0317	4.16	2.477780864	119.6	7.72311E-06
6	3	2.4762	119.5	1.031	4.16	2.476176585	119.5	9.45617E-06
8	1	2.4393	-1.4	1.0156	4.16	2.439293277	-1.4	2.75593E-06
	2	2.4933	-120.7	1.0381	4.16	2.493299432	-120.7	2.27984E-07
	3	2.4621	119.2	1.0251	4.16	2.462077115	119.2	9.29473E-06
12	2	2.4926	-120.7	1.0378	4.16	2.492599793	-120.7	8.30617E-08
9	1	2.4358	-1.5	1.0142	4.16	2.435784228	-1.5	6.47528E-06
13	1	2.4201	-1.9	1.0076	4.16	2.420088533	-1.9	4.73814E-06
	2	2.488	-121	1.0359	4.16	2.487994537	-121	2.19561E-06
	3	2.4486	118.9	1.0195	4.16	2.44858444	118.9	6.35462E-06
9R	1	2.4205	-1.5	1.0078	4.16	2.420497332	-1.5	1.10226E-06
14	1	2.4165	-1.5	1.0061	4.16	2.416496628	-1.5	1.39541E-06
34	3	2.4462	118.9	1.0185	4.16	2.446183438	118.9	6.77062E-06
18	1	2.3985	-2.3	0.99862	4.16	2.398479809	-2.3	8.4181E-06
	2	2.4781	-121.2	1.0318	4.16	2.478085956	-121.2	5.66705E-06
	3	2.4308	118.8	1.0121	4.16	2.430794148	118.8	2.40753E-06
11	1	2.4149	-1.5	1.0055	4.16	2.414881755	-1.5	7.55513E-06
10	1	2.4157	-1.5	1.0058	4.16	2.415684814	-1.5	6.28629E-06
15	3	2.4453	118.9	1.0181	4.16	2.445288366	118.9	4.75782E-06
16	3	2.4429	118.9	1.0171	4.16	2.442876306	118.9	9.69905E-06
61S	1	2.3721	-3.5	0.98765	4.16	2.372077443	-3.5	9.50946E-06
	2	2.4626	-122	1.0253	4.16	2.46259645	-122	1.4416E-06
	3	2.4137	117.8	1.005	4.16	2.41368051	117.8	8.07459E-06
300_OPEN	1	2.3779	-2.5	0.99007	4.16	2.377889857	-2.5	4.26564E-06
	2	2.461	-121.5	1.0246	4.16	2.46099059	-121.5	3.82346E-06
	3	2.4175	118.6	1.0065	4.16	2.417476102	118.6	9.88553E-06
94_OPEN	1	2.3952	-2.5	0.99727	4.16	2.395182304	-2.5	7.388E-06
610	1	0.27587	-2.7	0.99546	0.48	0.275867637	-2.7	8.56585E-06
	2	0.27972	-122	1.0093	0.48	0.279717265	-122	9.77917E-06
	3	0.28074	117	1.013	0.48	0.280738835	117	4.1506E-06

EPRI'S CIRCUIT 5

GRAPHICAL REPRESENTATION, 2 LAYERS USING DSSIM-PC/RT



Voltage (kV) calculated by OpenDSS and DSSim-PC/RT for EPRI's circuit 5 (26 Buses randomly selected)

			OpenDSS			DSSim-PC		
Bus	Node	VLN	Angle	pu	Base	VLN	Angle	Error
SOURCEBUS	1	69.715	0	1.05	115	69.714815	0	2.6587E-06
	2	69.715	-120	1.05	115	69.714554	-120	6.39352E-06
	3	69.715	120	1.05	115	69.71488	120	1.71967E-06
_MDV_SUB_1_LSB	1	7.3186	-33.4	1.0165	12.47	7.3185928	-33.4	9.78864E-07
	2	7.3058	-153.7	1.0148	12.47	7.3057826	-153.7	2.3824E-06
	3	7.3192	86.7	1.0166	12.47	7.3191765	86.7	3.20958E-06
MDV201	1	7.3186	-33.4	1.0165	12.47	7.3185584	-33.4	5.67862E-06
	2	7.3058	-153.7	1.0148	12.47	7.3057919	-153.7	1.11483E-06
	3	7.3192	86.7	1.0166	12.47	7.3191538	86.7	6.31822E-06
94743	1	7.1122	-34.2	0.98787	12.47	7.1121745	-34.2	3.58265E-06
94744	1	7.1121	-34.2	0.98785	12.47	7.1120759	-34.2	3.38994E-06
62244	3	7.1706	85.3	0.99598	12.47	7.1705323	85.3	9.44547E-06
62235	3	7.1698	85.3	0.99587	12.47	7.1697448	85.3	7.69875E-06
1144235	1	7.1499	-34.7	0.9931	12.47	7.1498504	-34.7	6.93136E-06
1144234	1	7.1495	-34.7	0.99304	12.47	7.1494748	-34.7	3.52963E-06
39754	2	6.9907	-155.3	0.97099	12.47	6.9906874	-155.3	1.80892E-06
MDV201_1160486ELB_INT	2	6.9907	-155.3	0.97099	12.47	6.9906927	-155.3	1.04679E-06
1160486	2	6.9907	-155.3	0.97099	12.47	6.9906935	-155.3	9.32578E-07
1160476	2	6.9954	-155.3	0.97164	12.47	6.9953782	-155.3	3.12256E-06
1160473	2	6.9944	-155.3	0.9715	12.47	6.9943558	-155.3	6.32285E-06
8163	1	7.2001	-34.3	1.0001	12.47	7.2000317	-34.3	9.48066E-06
	2	7.1931	-154.5	0.99911	12.47	7.193041	-154.5	8.19711E-06
	3	7.1945	85.4	0.9993	12.47	7.1944403	85.4	8.30227E-06
8164	1	7.1978	-34.3	0.99976	12.47	7.1977867	-34.3	1.84645E-06
	2	7.1913	-154.5	0.99885	12.47	7.1912592	-154.5	5.67021E-06
	3	7.1886	85.4	0.99848	12.47	7.1885455	85.4	7.5805E-06
103730	2	7.0883	-155.2	0.98454	12.47	7.0882416	-155.2	8.23438E-06
39610	2	7.0882	-155.2	0.98454	12.47	7.0881901	-155.2	1.39781E-06
835	1	7.2494	-33.8	1.0069	12.47	7.2493335	-33.8	9.16654E-06
842	1	7.2486	-33.8	1.0068	12.47	7.2485833	-33.8	2.29996E-06
39775	2	6.9977	-155.3	0.97196	12.47	6.9976621	-155.3	5.41534E-06
39760	2	6.9973	-155.3	0.9719	12.47	6.9972513	-155.3	6.95917E-06
15004	2	6.9933	-155.3	0.97135	12.47	6.9932329	-155.3	9.60112E-06
MDV201_1160480ELB_INT	2	6.9932	-155.3	0.97134	12.47	6.9931586	-155.3	5.91866E-06
1160480	2	6.9932	-155.3	0.97134	12.47	6.9931317	-155.3	9.77168E-06
1160492	1	7.13	-34.2	0.99033	12.47	7.1299687	-34.2	4.39491E-06

INTRODUCTION DSSIM-PC/RT SIMULATOR

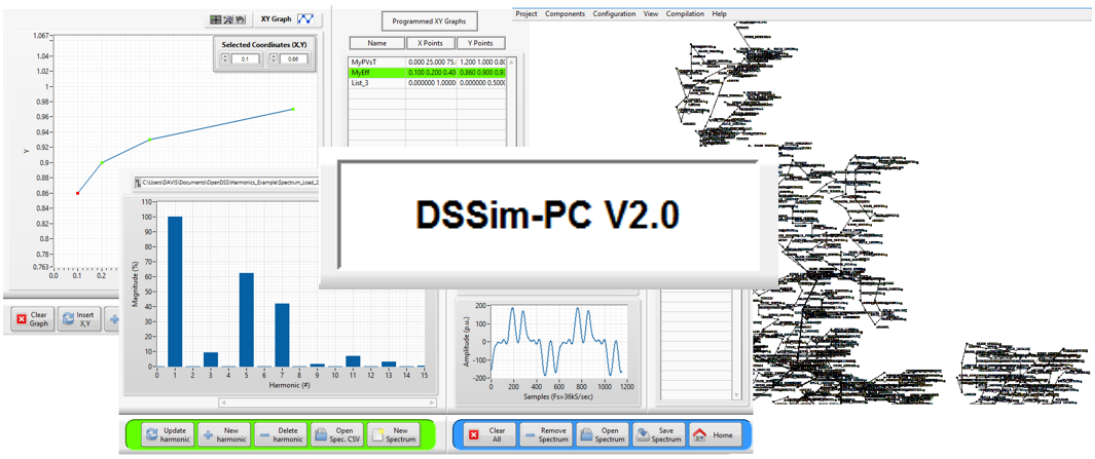


Figure A-1. DSSim-PC V2.0

As a product of this thesis, the Distribution System Simulator called DSSim was developed. This simulator counts with two different versions oriented to cover two different simulation environments: The conventional off line simulation and Real-time simulation, both platforms are called DSSim-PC and DSSim-RT respectively.

In the beginning, the aim of this development was to collaborate for improving existing open source simulation tools, adding advanced graphical user interfaces and interface to co-simulate with other programming languages. The open source simulator used for this experiment was EPRI's OpenDSS. The first version of DSSim-RT was released the Nov-10-2012 with a limited GUI and focused into attend Real-time experiments.

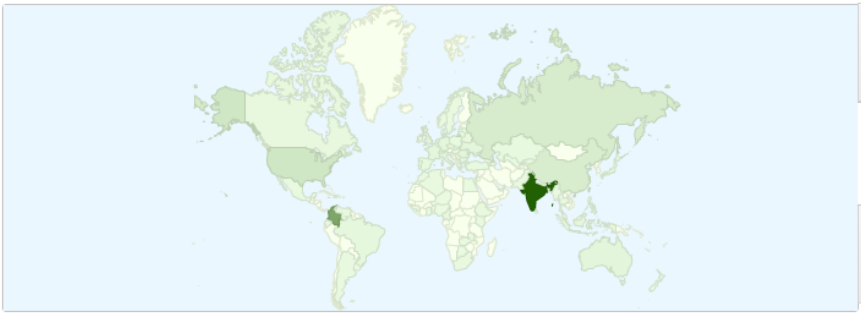
The main concern of this development is to try to answer the following question: Which are the features that a graphical simulator should include to satisfy the needs of the modern engineers? To answer this question and to contribute to the development of open source simulators, DSSim-PC was released to the public on July 20 of 2013 using a free account on sourceforge.net. Since then, this platform has been downloaded by more than 3600 users around the world, where the main users are located in India, Colombia, United States and Russia within 86 countries reported. Figure A-1 shows the download statistics generated by the sourceforge.net portal on Jul-06-2015.

As a consequence, this software has been evolving continuously; including automatic updates, new elements and models to the source code of OpenDSS and graphical reports, such as 3D projections of the load profile in time and animated tools to follow the evolution of the voltage profile in time. Additionally, it is possible to store and export the data of a simulation in both versions of DSSim. Basically, DSSim-PC became the test platform to improve the interfaces and GUI of both platforms.

One of the most appreciated features is the TCP communication server of DSSim, which allows to connect any programming language and even control the simulation using customized code. This characteristic was used to improve the communication between DSSim-RT and the remote equipment connected, which is used for generating waveforms and collect data from the real world. The TCP server allows also to exchange data with other simulators such as Opal-RT hardware, a development that in 2013 helps to generate the beta version of "DSSim-OP".

Brought to you by: davismont
Home / Installer (Change File)

Date Range: 2013-06-22 to 2015-07-07



DOWNLOADS
3.643
In the selected date range

TOP COUNTRY
India
43% of downloaders

TOP OS
Windows
51% of downloaders

OS downloads as: Percent

	Country	Android	Linux	Macintosh	Unknown	Windows	Total
1.	India	0%	14%	0%	81%	5%	1,575
2.	Colombia	0%	0%	1%	0%	98%	855
3.	United States	0%	1%	1%	40%	59%	194
4.	Russia	0%	1%	0%	1%	98%	138
5.	China	2%	4%	2%	7%	85%	133
6.	Ukraine	2%	0%	0%	0%	98%	48
7.	United Kingdom	0%	2%	21%	38%	40%	48
8.	Australia	0%	0%	0%	11%	89%	47
9.	Romania	0%	0%	0%	0%	100%	41

Figure A-1. Download Statistics from the website sourceforge.net

This software is very efficient because it is built using the actor model as a framework, which allow to handle the different parts of the software as independent programs interacting by sending messages between them. As a consequence, the inclusion of graphical objects to the simulation does not affects the performance of the simulation; a feature that is possible because each independent program (actor) can be executed in a different thread and core within a multicore processor, which is nowadays conventional computing architecture.

With these results the next step was to improve the solver of the simulator by incorporating mechanisms to make the solution algorithm multicore. As a result, the methodology called Diakoptics based on actors (A-Diakoptics) was created, revealing a path for taking conventional methods for power system analysis to a multicore environment, using nowadays computing architectures.

Then, for improving the integration between OpenDSS and DSSim-PC/RT a new module is introduced. This module is the translator from .DSS (OpenDSS) to .DSP (DSSim), which originally was developed only for the RT version with some limitations. Now, this module is fully integrated to DSSim-PC and allows to translate scripts from OpenDSS directly into DSSim-PC/RT. This tool has been also appreciated by the users and it has reduced the time for implementing large-scale power systems in DSSim-PC/RT, and to perform large modification to these systems as well.

There are registers and evidence that this software DSSim-PC is being used widely in many projects in academia and industry as mentioned in Chapter 5. This evidence and the constant evolution of DSSim-PC/RT, shows that this kind of developments are well appreciated in research and development, which we hope, will lead this tool to more advanced versions. Currently, DSSim-PC/RT is in the version 2.0 with accelerated graphics, advanced graphical assistants to configure devices, new meters and monitors for collecting data and advanced

commands for controlling the simulation remotely. But the main goal is arriving, this is to incorporate the methodology A-Diakoptics to the OpenDSSEngine provided with OpenDSS, which is expected will extend the advantages of this method to the conventional users of OpenDSS.