A COMPENSATED DISTRIBUTED LINE DECOUPLING APPROACH

FOR REAL TIME APPLICATIONS

By

Babikir Elnouman Babikir Mohamed Ahmed

Abdelrahman A. Karrar
Associate Professor of Electrical Engineering
(Chair)

Ahmed H. Eltom
Professor of Engineering
(Committee Member)

Gary L. Kobet
Adjunct Professor of Electrical Engineering
(Committee Member)

Vahid R. Disfani
Assistant Professor of Electrical Engineering
(Committee Member)

A COMPENSATED DISTRIBUTED LINE DECOUPLING APPROACH FOR REAL TIME APPLICATIONS

By

Babikir Elnouman Babikir Mohamed Ahmed

A Thesis Submitted to the Faculty of the University of
Tennessee at Chattanooga in Partial
Fulfillment of the Requirements of the Degree of
Master of Science: Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

May 2020

ABSTRACT

This work proposes a new method, called the Compensated Distributed Line Decoupling (CDLD), to decouple distribution networks among parallel processing cores in a real-time multi-core environment. Due to the short length of distribution lines, decoupling the network for real-time processing is challenging. Previous studies proposed the Stublines and State Space Nodal (SSN) solvers, but both approaches have limitations. The proposed method addresses these limitations to implement improvements. Specifically, the CDLD method can be used as an enhanced Stubline decoupling, improving on its accuracy and transient response, or it can be combined with the SSN solver to improve its computational performance and remove bottleneck issues. The CDLD method was tested on three IEEE systems in real-time, and significant improvement was realized in network response and computational performance compared to the prevailing methods. The combined SSN-CDLD method proved to be the most promising approach for network decoupling.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Real-time simulators have been widely used in the design and planning of power systems for a long time. With the inclusion of distributed energy resources in the recent years, simulation of large distribution networks in real-time have become more and more complex [1].

There are a limited number of commercial players offering real-time computational hardware and software technologies. The real-time platform used in this work is RT-LAB from Opal-RT Technologies, the main reason being the ability to develop the work in MATLAB/Simulink®, and to use specific solvers such as the SSN solver, which is offered by Opal RT in their real-time MATLAB plugin. A description of the SSN solver and its significance for this work will be provided later in chapter 2.

1.2 Problem Statement

Decoupling becomes a necessity when attempting to run large networks in real-time. The computational burden of these systems needs to be redistributed among several processor cores to achieve real-time simulation. Decoupling distribution level networks for real-time multi-core processing is considered one of the challenging research topics. This is due to the fact that distribution lines are normally of short length and if modelled as distributed parameter lines or so called “Bergeron-type line”, then the propagation delay of these lines will be far less than the

typical time-step used for real-time applications. Therefore, several methods for network decoupling such as the Stubline and State-Space Nodal (SSN) have been proposed in the literature but both have limitations. This work consequently addresses the need for, and proposes an improved decoupling method.

1.3 Objective

This work aims to develop a new decoupling method for the implementation of distribution networks on a real-time simulator with multi-CPU computation. This method improves on the limitations of previous decoupling techniques, and it can additionally be used to enhance the computational performance of the SSN method.

1.4 Thesis Layout

The remainder of the thesis is structured as follows:

- Chapter 2: this chapter provides an overview of the literature on various decoupling techniques used for real-time simulations.
- Chapter 3: this chapter discusses the fundamentals of modeling with RT-LAB software and how to take advantage of the various features provided by ARTEMiS.
- Chapter 4: this chapter presents the concepts and theory of the proposed CDLD method along with a description of the distribution test feeders used to validate the method.
- Chapter 5: this chapter presents simulation results when applying the proposed method on different test systems. Moreover, a comparison is made between the performance of the CDLD, State-Space Nodal (SSN) and a combination of the two called SSN-CDLD.
- Chapter 6: this chapter concludes the contributions and findings of this work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The need for fast, flexible and scalable real-time simulators is strongly felt in electric utility studies today, particularly when investigating the response of hardware-in-the-loop equipment such as relays and other intelligent electronic devices. Digital real-time simulators (DRTS) are indispensable when testing and commissioning new protection and control equipment to ensure the correct and reliable performance of their intended functions. This is corresponding to the high capital costs involved in operating and safeguarding electrical power grids. In recent years, power system analysis and operation have also become increasingly complex with the inclusion of distributed energy resources (DER) with intermittent and asynchronous generation. It is important to understand and analyze its effect on the overall power system using DRTS [1], [2].

The solver used is a fixed time-step solver that processes the inputs, the model computations and then outputs within this time-step. If the predetermined time-step cannot accommodate all these processes in real-time, then an ‘overrun’ is said to occur, and this time-step will be omitted as shown in figure 2.1. The computation will resume at the next time step. This will cause a departure from the real-time capabilities, and may be solved by increasing the time-step, or simplifying the model [3].

It should also be possible to attempt redistributing the computational burden among computational cores, or assigning additional computational cores if available. The RT-Lab solver has a console subsystem running on a host PC, and which interfaces the computational blocks on the real-time target. These computational blocks are arranged in a master subsystem and a number of slave subsystems, whose number depends on the size of the model and the number of computational cores available at real-time target.

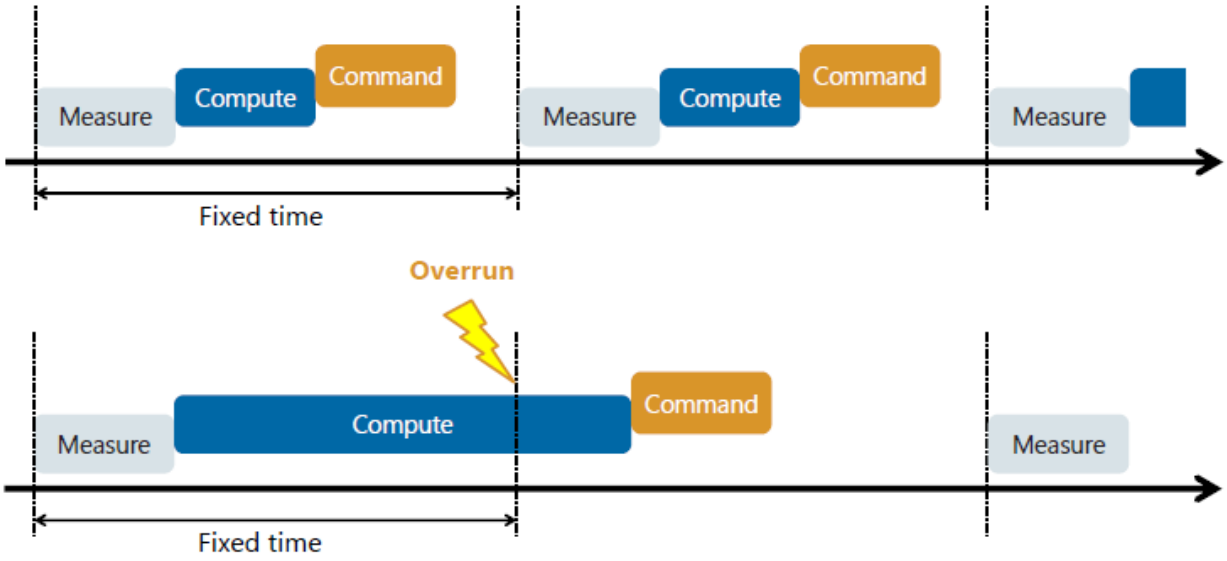

Figure 2.1

Overrun in real-time simulation

For real-time simulations of transmission networks, the classical approach for decoupling the network among computation cores is to exploit the inherent propagation delay in transmission lines. These lines are typically modelled as so called ‘Bergeron-type’ or frequency-dependent, and for decoupling to be successful, the propagation delay for these lines needs to be greater than one computational time-step in the digital real-time simulation.

A typical high voltage network comprises many of these lines, offering suitable candidates for decoupling points in the network, and thus making possible parallel computation on several CPU cores [2], [4]. Distribution networks, on the other hand, comprise lines of much shorter length. The cutoff length for a particular simulation time-step can be approximated by assuming the propagation at the speed of light (in reality it is slight less), and multiplying this speed with one time-step. For a 50 μs time-step, this gives a cutoff line length of 15 km. Distribution lines are typically much shorter, and are usually modeled as π-lines in real-time analysis. An example is the segment between nodes 703 and 730 (of phase "A") taken from the IEEE 37 node test feeder [5]. This section is of 0.1829 km length, and has a positive sequence capacitance of 0.123 μF/km and positive sequence inductance of 1.106 mH/km giving a propagation delay of 2.13 μs. In CPU based real-time simulations, it is typically very difficult to go below 10 µs for large systems without experiencing computational overruns.

2.2 Decoupling Techniques for Distribution Grids

Parallel computation of a distribution feeder across multiples cores on the target simulator is always a challenging task especially for large distribution systems. Over the past years, various techniques were used to parallelize the computations for real-time simulation [6].

2.2.1 Stublines

As a workaround the time step limitation, one method of decoupling is employing the so-called 'ARTEMiS Stubline', which is an actual Bergeron line model adjusted to produce a one time-step propagation delay. This Stubline block, shown in figure 2.2, will allow the decoupling of state-space equations of systems on both sides of the Stubline [7].

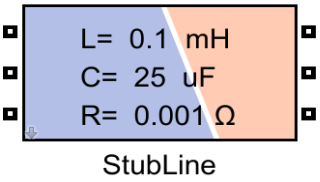

Figure 2.2

ARTEMiS Stubline Block

The common practice is to deduct an equal part of the inductance of an already existing line section from all three phases so that the deducted part (L_s) will represent a balanced line and it can easily be modeled by a Stubline block. This Stubline will add more capacitance to the model than in reality at the point of insertion. To find the capacitance calculated by the Stubline block, we look at the expression for a distributed line propagation delay (T):

$$T = \sqrt{LC} \tag{2.1}$$

Then, the Stubline capacitance C_s to produce this one time-step delay is:

$$C_s = \frac{T^2}{L_s} \tag{2.2}$$

The implementation works better when a transformer is available at a suitable decoupling point so that its series impedance (which is typically large) may serve as the same for the stubline, and then only a small shunt capacitance needs to be added to give the required delay. If no transformer can be found, then an existing line will have to be used. In this case, the added capacitance will be large with potential adverse effects on the accuracy of the simulation, as well as introduction of unintended transients [2], [6]. Furthermore, distribution lines with mutual coupling between the phases, as it is the case of overhead line which is the common configuration type of distribution feeders, offer additional complications for the Stubline

implementation, making available only part of the series impedance as a Stubline. The use of Stubline blocks for parallelization is therefore not considered to be the ideal solution.

2.2.2 State-Space Nodal (SSN)

2.2.2.1 ARTEMiS

ARTEMiS, Advanced Real-Time Electro-Magnetic Simulator, is a Simulink plug-in to the SimPowerSystems (SPS) tool that allow real-time simulation of SimPowerSystems models in the RT-LAB environment. The objective is to complete all iterations of the model within the specified time-step while maintaining the simulation accuracy to a certain level. This can be a problem as real-time simulations demand the use of fixed-step solvers [7]. ARTEMiS offers several real-time solvers but a new one called State-Space Nodal was developed and integrated for the first time to ARTEMiS version 6.0. This solver combines the accuracy of SPS solvers with the natural ability of the nodal method to handle circuits with large number of switches.

2.2.2.2 SSN Solver

The State-Space Nodal method can be seen as a nodal method that solves both steady state and transient inaccuracies [1]. The SSN method is a novel approach that decouples the network into groups separated by virtual ‘nodes’. The user selects how the SSN groups are divided. Each group is solved separately as discretized state-space equations with the unknown nodal voltages implied in the equations. Each state space group then results in a single nodal equation which is solved simultaneously with the other groups using nodal analysis.

One of the difficulties associated with any state space approach is the expensive time spent on recalculation of the state space matrices whenever a network switching event occurs.

Specific implementations of the SSN method solve this by arranging a limited number of switches (ideally less than 10) in each group and then pre-calculating all matrix permutations for the possible switch combinations in the group before real-time execution [7], [8].

2.2.2.3 SSN Delay-Free Parallelization

Since ARTEMiS version 6.3, the SSN solver can now be executed across multiple cores without any delays. This resulted in a significant improvement to the performance of the solver. The SSN solver handles allocation of processor cores to the SSN groups such that the total computation time is minimized.

One would expect that increasing the numbers of groups and the cores allocated to them should speed up the computation and minimize the burden on each core. However, the Opal-RT experience shows that this is not necessarily the case, where they indicate that the gains in from the parallel process tend to saturate when using more than five or six cores [7]. One of the possible reasons may be the associated increase in the dimension of the nodal solution portion when increasing number of cores [6].

2.2.2.4 State-Space Nodal Theory

Discrete equations must be formulated to allow digital simulation of any continuous system with a state equation of [8], [9], [10], [11],

$$\dot{x} = [A_k]x + [B_k]\,u \tag{2.3}$$

$$y = [C_k]x + [D_k]\,u \tag{2.4}$$

where x and u are the state variables and inputs of the system, respectively. The vector y is the vector of outputs. The state-space matrices A_k, B_k, C_k and D_k are the kth permutation of switches.

The trapezoidal rule is used to develop the difference equation of the system,

$$x(t+\Delta t) = x(t) + \frac{\Delta t}{2}[\dot{x}(t+\Delta t) + \dot{x}(t)] \tag{2.5}$$

Where (t) is the last time step known solution values and $(t+\Delta t)$ is the current time step.

By substituting the state equation (2.3) into (2.5)

$$x(t+\Delta t) = x(t) + \frac{\Delta t}{2}\left([A_k]\, x(t+\Delta t) + [B_k]\, u(t+\Delta t) + [A_k]\, x(t) + [B_k] u(t) \right) \tag{2.6}$$

Then, collecting similar terms,

$$\left([I] - \frac{\Delta t}{2}[A_k]\right) x(t+\Delta t) = \left([I] + \frac{\Delta t}{2}[A_k]\right) x(t) + \frac{\Delta t}{2}[B_k]\big(u(t+\Delta t) + u(t)\big) \tag{2.7}$$

Rearranging equation (2.7) to give $x(t+\Delta t)$,

$$\begin{aligned} x(t+\Delta t) = {} & \left[[I] - \frac{\Delta t}{2}[A_k]\right]^{-1} \left[[I] + \frac{\Delta t}{2}[A_k]\right] x(t) \\ & + \left[[I] - \frac{\Delta t}{2}[A_k]\right]^{-1} \frac{\Delta t}{2}[B_k]\big(u(t+\Delta t) + u(t)\big) \end{aligned} \tag{2.8}$$

This discretized state equation can simply be rewritten as

$$x_{t+\Delta t} = \hat{A}_k x_t + \hat{B}_k u_t + \hat{B}_k u_{t+\Delta t} \tag{2.9}$$

Where, $\hat{A}_k = \left[[I] - \frac{\Delta t}{2}[A_k]\right]^{-1} \left[[I] + \frac{\Delta t}{2}[A_k]\right]$, $\hat{B}_k = \left[[I] - \frac{\Delta t}{2}[A_k]\right]^{-1} \frac{\Delta t}{2}[B_k]$

In the SSN method, $\hat{A}_k$ and $\hat{B}_k$ terms are pre-calculated before real-time execution for all permutation of switches that results in different state-space matrices. Equation (2.9) along with (2.4) can be modified as

$$x_{t+\Delta t} = \hat{A}_k x_t + \hat{B}_k u_t + \begin{bmatrix} \hat{B}_{k_i} & \hat{B}_{k_n} \end{bmatrix} \begin{bmatrix} u_{i_{t+\Delta t}} \\ u_{n_{t+\Delta t}} \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} y_{i_{t+\Delta t}} \\ y_{n_{t+\Delta t}} \end{bmatrix} = \begin{bmatrix} C_{k_i} \\ C_{k_n} \end{bmatrix} x_{t+\Delta t} + \begin{bmatrix} D_{k_{ii}} & D_{k_{in}} \\ D_{k_{ni}} & D_{k_{nn}} \end{bmatrix} \begin{bmatrix} u_{i_{t+\Delta t}} \\ u_{n_{t+\Delta t}} \end{bmatrix} \tag{2.11}$$

Where i and n subscripts refer to internal sources and external nodal injections, respectively. Now to solve for the external nodal output, $y_{n_{t+\Delta t}}$, of a state-space group, equation (2.10) is substituted into the 2nd row of (2.11),

$$y_{n_{t+\Delta t}} = C_{k_n}\left(\hat{A}_k x_t + \hat{B}_k u_t + \hat{B}_{k_i} u_{i_{t+\Delta t}}\right) + D_{k_{ni}} u_{i_{t+\Delta t}} + \left(C_{k_n}\hat{B}_{k_n} + D_{k_{nn}}\right) u_{n_{t+\Delta t}} \tag{2.12}$$

Equation (2.12) has two types of terms, one with known variables from past history with (t) and one which is unknown $(t + \Delta t)$. Therefore, equation (2.12) can be written as,

$$y_{n_{t+\Delta t}} = y_{k_{hist}} + W_{k_n}\, u_{n_{t+\Delta t}} \tag{2.13}$$

Where

$$y_{k_{hist}} = C_{k_n}\left(\hat{A}_k x_t + \hat{B}_k u_t + \hat{B}_{k_i} u_{i_{t+\Delta t}}\right) + D_{k_{ni}} u_{i_{t+\Delta t}} \tag{2.14}$$

And,

$$W_{k_n} = C_{k_n}\hat{B}_{k_n} + D_{k_{nn}} \tag{2.15}$$

Equation (2.13) can be interpreted in two ways:

- When y_n stands for current injections entering a group and u_n is for node voltages, then $y_{k_{hist}}$ reflects history current sources ($i_{k_{hist}}$) and W_{k_n} is an admittance matrix. This type of group is then called a V-type SSN group and equation (2.13) describes a Norton equivalent.
- When y_n stands for voltages and u_n is for current injections, then $y_{k_{hist}}$ reflects history voltage sources ($v_{k_{hist}}$) and W_{k_n} is an impedance matrix. This type of group is an I-type SSN group and equation (2.13) will describe a Thévenin equivalent.

One can come across a system with a combination of the two types of groups (V-type and I-type). In such case, equation (2.13) can be rewritten into a mixed-type group equation as follows,

$$\begin{bmatrix} i_n^V \\ v_n^I \end{bmatrix} = \begin{bmatrix} i_{k_{hist}} \\ v_{k_{hist}} \end{bmatrix} + W_{k_n} \begin{bmatrix} v_n^V \\ i_n^I \end{bmatrix} \tag{2.16}$$

where the superscripts V and I denote the SSN group type.

The mixed-type equation (2.16) can easily be transformed into a nodal representation by rearranging all current vectors to the left-hand side. Then, W_{k_n} becomes the admittance matrix for the mixed-type SSN group and will be inserted to the global nodal admittance matrix, Y_N,

$$i_{N_{t+\Delta t}} = Y_N \; v_{N_{t+\Delta t}} \tag{2.17}$$

Where i_N represents the vector of known nodal current injections, and v_N, the vector of all unknown nodal voltages.

The modified-augmented-nodal analysis (MANA), introduced in [12], is used to eliminate any topological restrictions from the classical nodal analysis method by avoiding matrix inversions. MANA can insert the lower part of equation (2.16) into the main network equations and simultaneously solve it with other group equations [8], [11]. In MANA, equation (2.17) can be written as

$$b_{N_{t+\Delta t}} = A_N \; x_{N_{t+\Delta t}} \tag{2.18}$$

Where subscript N implies MANA matrices and vectors (not state-space matrices), and x_N is the vector of unknown voltages and currents. The V-type rows from equation (2.16) is inserted directly into the MANA equation (2.18) while the I-type variables are regrouped on the right-hand side and listed in x_N with their coefficients inserted into A_N.

The history components ($i_{k_{hist}}$ and $v_{k_{hist}}$) take part in b_N with a negative sign. The matrix A_N will change between time-steps whenever the W_{k_n} matrix of any SSN group changes for a switching event.

2.2.2.5 SSN Speed Improvement Over State-Space Methods

The SSN solver provides numerous benefits over all state-space methods in both real-time and non-real-time applications. The SSN grouping approach reduces the size and complexity of state-space equations for each group.

The SSN groups can be solved in parallel and the number of pre-calculated matrix sets for the different switching topologies can become substantially reduced [7], [8]. These groups are connected only through the nodal interfacing equations. The computational burden of equation (2.18) is negligible when compared to much larger group equations.

Figure 2.3 shows a three-phase system that has two breakers (switches S1 and S2) and two pi-lines. The state variables here are the capacitor voltages and inductor currents. The number of state variables for any system is donated by m; and for the given example, $m = 15$ states.

The total number of matrices (A_k, B_k, C_k and D_k) to be stored in memory by the state-space method, for different permutation of switches, is 2^S; where S is the number of switches found in the system. Therefore, to run this system in real-time, the state-space solver will have to pre-calculate and save $2^6 = 64$ matrices in memory before execution.

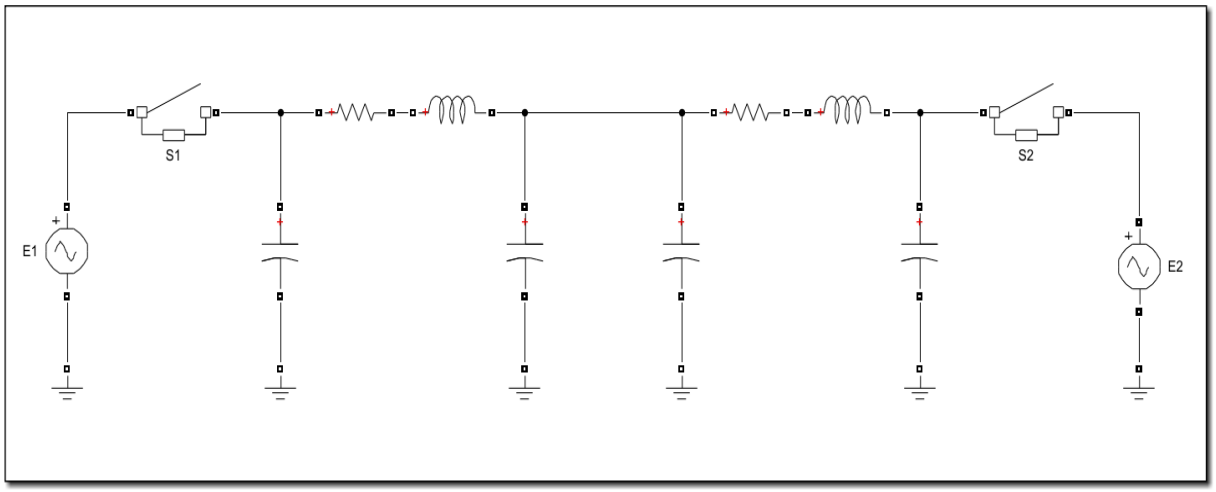

Figure 2.3

Example of three phase system using State-space method

Using the delay-free SSN method, the system can be divided and separated into two groups using 3 nodal connection points (one per each phase) as shown in figure 2.4.

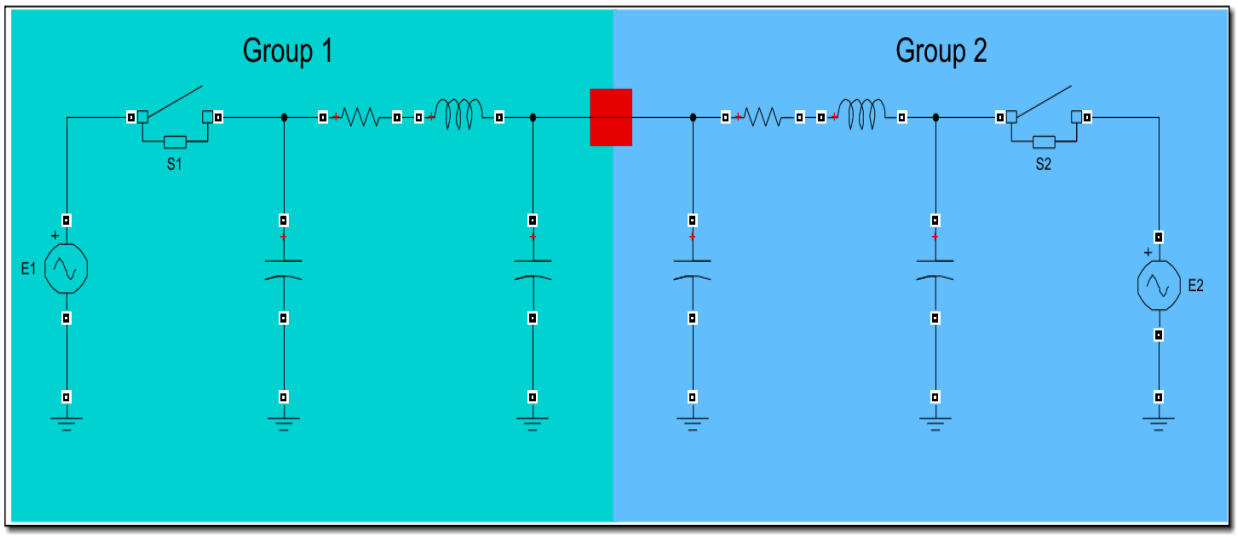

Figure 2.4

Example of three phase system using SSN method

With discretization of the SSN method, the state space equation will become

$$\dot{x} = {}^{A1_i}_{A2_j}\, x + {}^{B1_i}_{B2_j}\, u \tag{2.19}$$

Where i and j are the state-space matrices index (ith and jth permutation of switches) for group 1 and 2, respectively. The two SSN groups are linked only through the nodal interfacing equation with a 3x3 admittance matrix. It is clear that the number of states for each group is 9 states, or simply (m +3)/2. Moreover, the number of switches for each SSN group in figure 2.4 is 3, and hence, the total number of pre-calculated matrix sets to be saved is now $2 \times 2^3 = 16$ matrices. This substantial drop in the number of pre-calculated matrices will have a direct impact in reducing the memory requirements and computational burden on the processor cores.

2.2.2.6 SSN Limitation and Switch Management

A large number of switches and switching independencies would create a challenge with state-space approaches however, as would subsystems with a large number of states such as frequency-dependent, modal or phase domain lines. Again, specific SSN implementations attempt to overcome these limitations by solving them in the main nodal admittance domain along with the nodal equations. Custom coded nodal groups specific to the user are also interfaced to the main nodal loop [10].

CHAPTER 3

RT-LAB and ARTEMiS Guide to Real-Time Simulation

3.1 RT-LAB Software

RT-LAB™ is a distributed real-time platform fully integrated with MATLAB/Simulink, designed by OPAL-RT Technologies, that improves the design process by taking engineers from Simulink dynamic models to real-time simulations with hardware-in-the-loop (HIL), in a very short time, at a low cost. It is flexible enough to be applied to the most complex simulation and control problem, whether it is for real-time hardware-in-the-loop applications or for speeding up model execution, control and test [13].

RT-LAB uses Simulink to define models that will be executed by the real-time multiprocessing system and defines its own simulation parameters through Simulink's. RT-LAB software runs on a hardware configuration consisting of the following: command station (host PC), compilation node, target nodes and I/O boards as shown in figure 3.1.

3.1.1 Command Station (The host)

RT-LAB software is configured on a Windows or Linux computer called the command station. The Command Station is a PC workstation that serves as your interface. The Command Station enables you to:

- Edit and modify models.
- See model data.

- Run the original model under its simulation software (Simulink).
- Distribute code.
- Control the simulator's Go/Stop sequences.

Simulations can be run entirely on the command station computer, but they are typically run on one or more target nodes.

3.1.2 Compilation Node and Target Nodes

For real-time simulation, the preferred operating system for the target nodes is OPAL-RT Linux (x86-based). When there are multiple target nodes, one of them is designated as the compilation node. The Command Station and target node(s) communicate with each other using communication links (TCP/IP) and for hardware-in-the-loop simulations target nodes may also communicate with other devices through I/O boards. The target nodes are real-time processing and communication computers that use commercial processors that performs:

- Real-time execution of the model’s simulation.
- Real-time communication between the nodes and I/Os.
- Acquisition of the model’s internal variables and external outputs through I/O modules.
- Implementation of user-performed online parameters modification.
- Recording data on local hard drive, if desired.
- Supervision of execution of the model’s simulation and communication with other nodes.

The compilation node is used to compile generated C code. Any target node could be the compilation node.

3.1.3 Input/output boards

Various analog, digital and timer I/O boards are supported by RT-LAB. These enable connections to external equipment for applications such as HIL. Interfaces for I/O devices are configured through custom blocks that need only be added and connected to the graphic model's blocks. RT-LAB's automatic code generator will map the model's data onto the physical I/O cards.

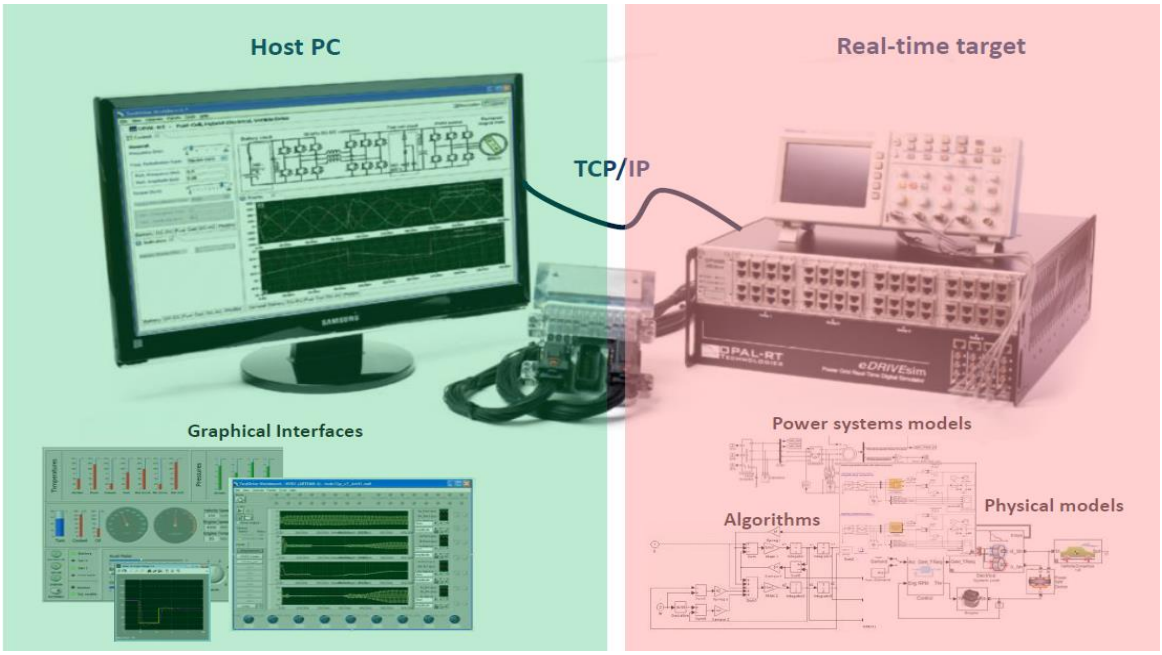

Figure 3.1

RT-LAB hardware configuration

3.2 RT-LAB Modeling Fundamentals

The starting point for any real-time simulation using RT-LAB environment is implementing the system model in Simulink. Then, the steps discussed below are necessary to realize the model with Opal-RT's software and hardware [7], [13], [14].

3.2.1 Grouping into Subsystems

In RT-LAB platforms, subsystems are classified to either computational subsystems or GUI subsystem (Console/Host). The console subsystem contains user interface blocks, such as scopes, displays and controls that will be displayed on the Host PC. The computation subsystems, which includes a master and any number of slave subsystems, contain the computational elements of the model, such as mathematical operations, I/O blocks, generators, transmission lines, transformers and loads.

The computation subsystem will be executed in real-time on one CPU core of the real-time target, and therefore, a slave subsystem is only needed for models with large computational elements that require to be distributed across multiple nodes to achieve real-time simulation. The data between two computation subsystems is exchanged synchronously through shared memory, while the exchange between any computation subsystem and the GUI subsystem is done asynchronously through the TCP/IP link as shown in figure 3.2. Special care when naming RT-LAB subsystems as each type should start with a unique identifier summarized in table 3.1.

Table 3.1

Subsystem Naming in RT-LAB

Subsystem	**Identifier (Prefix)**	**Type**
Console	SC_	GUI Subsystem
Master	SM_	Computation Subsystem
Slave	SS_	Computation Subsystem

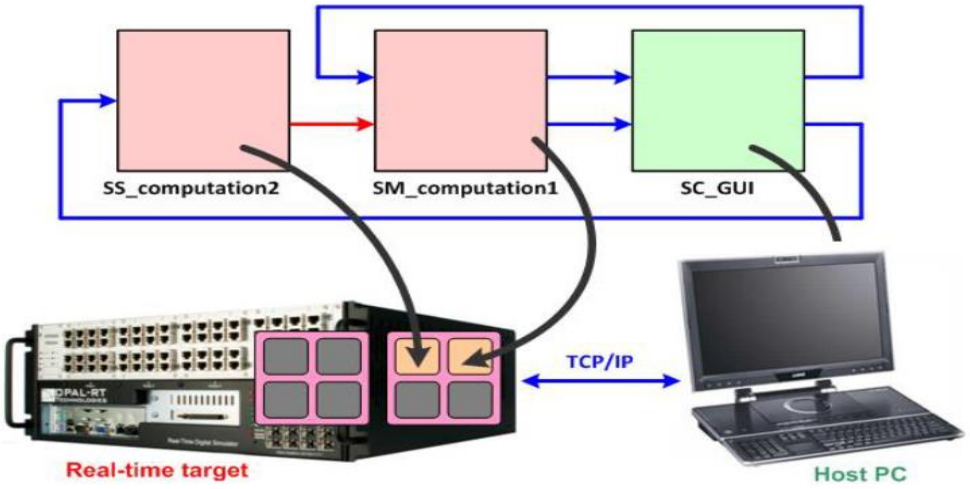

Figure 3.2

Assignation of Subsystems to different CPU cores

3.2.2 Adding OpComm Blocks

OpComm blocks are responsible for the communication between subsystems. All subsystems (SM, SS, SC) inputs must first go through an OpComm block before any operations can be done on the received signals. Figure 3.3 shows OpComm block connected to the inputs of a master subsystem.

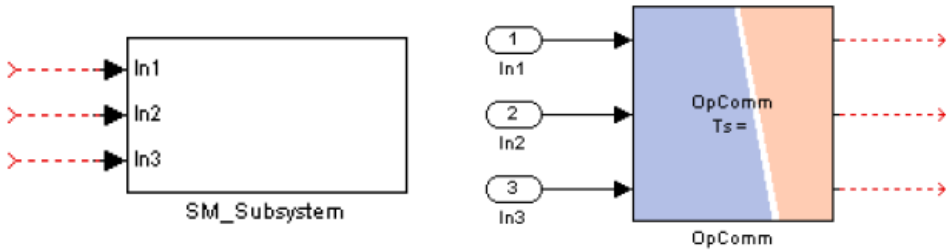

Figure 3.3

OpComm Block

For computational subsystems (SM and SS), two OpComm blocks will be needed. One to receive real-time synchronized signals from other computational subsystems, and the other one receives asynchronous signals from the GUI subsystem. On the other hand, one OpComm is normally enough for the console GUI subsystem (SC).

3.2.3 Setting Simulation Parameters

RT-LAB real-time simulations are performed in a discrete time with constant step, also known as fixed step, as time moves forward in equal duration of time. Therefore, the following configuration parameters within the Simulink model need to be set as follows:

- Set the Type to Fixed-step.
- Set the Stop time to inf.
- Select any fixed step trapezoidal Solver.
- Set the Fixed-step size (Sample time): the value is in seconds.
- Under “All Parameters”, search for “Block reduction” and make sure it is unchecked.

3.2.4 Executing the RT-LAB Compatible Model

Before building and loading the model with RT-LAB, it should be run off-line just to see if there are any errors available. If the model does not run under Simulink, it will surely will not work in real-time and these errors need to be addressed. Once they are resolved, the model is set for real-time simulation with RT-LAB. When building a Simulink model with RT-LAB, you select the development node (target node) on OPAL-RT Linux and then start building the model. During this process RT-LAB will separate the model to a number of files corresponding to the number of subsystems (SM, SS and SC).

After model separation, RT-LAB generates C code for each individual model and transfers these codes through an internal RT-LAB process (OpalD) to the simulator. The target compiler, on the other hand, builds and links the files to generate real-time executables to be transferred back to the host computer.

Once the build process is complete, the user will then need to assign subsystems to targets (subsystems can be run on the same target or on different ones). This can be done on the "Assignation" tab for the model as shown in figure 3.4. Normally, RT Lab assigns one processor core for each subsystem but when using the SSN solver, the user can assign more than one core for a subsystem. Furthermore, eXtra High Performance (XHP) mode can be enabled on this assignation tab (figure 3.4). XHP mode allows very fast computation of the real-time model on the target system. Finally, in order to run the model in real-time, it should be loaded first and then executed (Run). Once the simulation is finished, the user should reset the model.

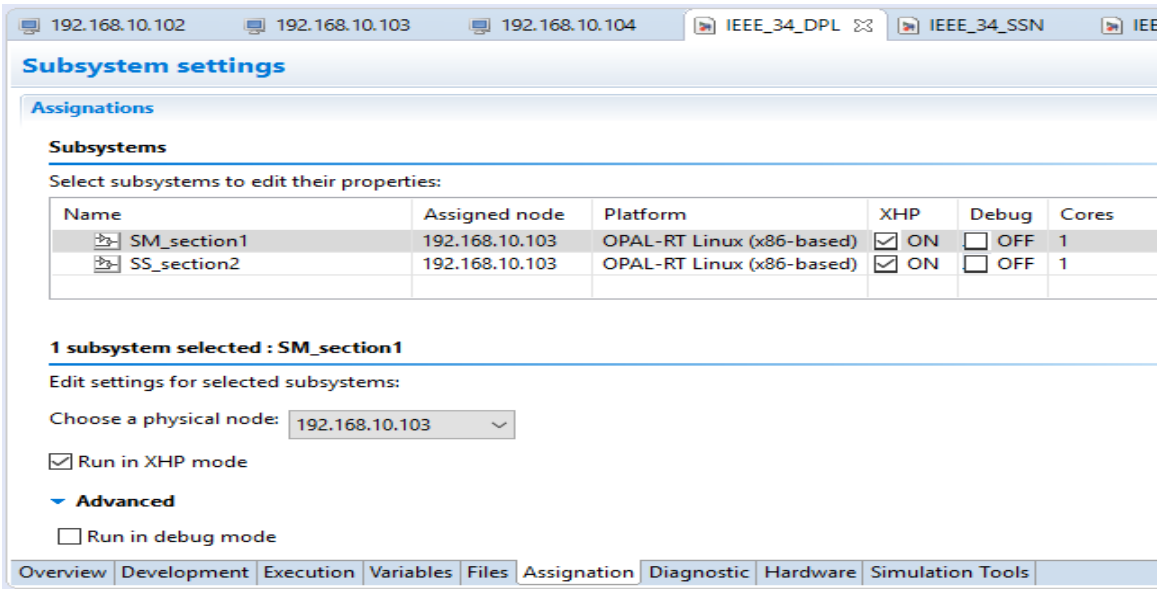

Figure 3.4

RT-LAB Assignation Tab

3.3 Useful RT-LAB Blocks Used in this Work

3.3.1 OpWriteFile Block

The OpWriteFile block, shown in figure 3.5, found in the RT-LAB library in Simulink allows recording signals to a binary file on target, one column for each time-step [14]. Multiple signals can be recorded into a MAT file using a "Mux" block to create a vector. The first row is the simulation time; the second row is the first element of the input vector and so on.

The OpWriteFile block parameters are inserted as follows:

1. Specify the "Variable name".
2. Specify the "File name".
3. Specify the Number of Samples per signal ($Nbss$). It is best to calculate that using: $Nbss = \frac{T_{end} - T_{start}}{Ts}$, where Ts the sample time; T_{end} and T_{start} are the simulation end and start times.
4. Calculate the Buffer size in bytes (SizeBuf) as follows,

$$\text{SizeBuf} = (Nsig + 1) \times 8 \times Nbss$$

   Where $Nsig$ is the number of signals to be recorded.
5. Finally specify the "File size limit" by calculating the minimum file size from using: $MinSize = Nbss \times SizeBuf$.

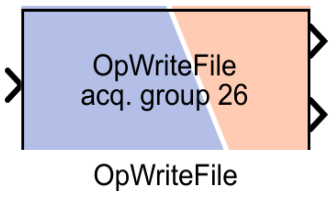

Figure 3.5

OpWriteFile Block

3.3.2 OpMonitor Block

The OpMonitor block, shown in figure 3.6, allows to retrieve timing information from the model (time values are given in microseconds). This block is intended to be used in an RT-LAB model only, and will return default values when used off-line in Simulink [14]. Information is relevant only to the subsystem where this block is inserted, and those information includes:

- Computation time (Computing only).
- Real Step Size (computing + overhead + synchronization).
- Idle time.
- Number of overruns.

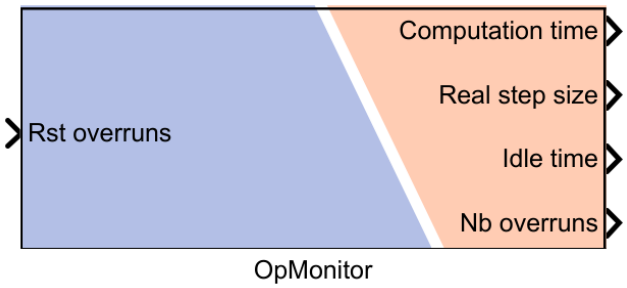

Figure 3.6

OpMonitor Block

3.4 ARTEMiS Blockset Library

ARTEMiS blocks were designed intentionally to enhance the accuracy and implementations of power system models for real-time simulation. ARTEMiS solvers (e.g. SSN solver discussed in Chapter 2) can also provide faster real-time simulation of SimPowerSystems (SPS) models when compared to typical SPS trapezoidal solvers.

ARTEMiS comes with special blocks for real-time simulation such as ARTEMiS Distributed Parameters Line and ARTEMiS Stubline (discussed in section 2.2.1) that enables distributed simulation of power systems on several CPU cores using RT-LAB.

ARTEMiS also provides special model options such as SSN saturable transformer, Inlined Thyristor Valve Compensation (ITVC), 3-level NCP inverter, ARTEMiS Distributed Parameters Line with variable internal fault distance, ARTEMiS TSB 2-Level, 3-Level NPC TSB with high-impedance capability, ARTEMiS MMC 1P cell and 2P cell. In this section, the most commonly used ARTEMiS blocks will be discussed [7], [14].

3.4.1 ARTEMiS Guide Block

The ARTEMiS Guide block, shown in figure 3.7, is the main discrete simulation parameter control block of ARTEMiS from which various ARTEMiS solvers can be chosen. The block pre-calculates and discretizes all state-space matrices for all combinations of the switch topologies, therefore, allowing faster real-time simulations. The ARTEMiS Guide block must be placed at the top-level of the Simulink model.

On the “General” tap of the ARTEMiS guide block, one can enable the State-Space Nodal method (SSN) to be used in SPS subsystems. On the “State-Space Solver Options” tap, the ARTEMiS discretization method is set to one of available methods: art3, art3hd and art5 (default). However, when the SSN method is enabled, an SSN solver has to be set on “SSN Options” tap.

Figure 3.7

ARTEMiS Guide Block

3.4.2 ARTEMiS Distributed Parameters Line (DPL)

The ARTEMiS DPL block, shown in figure 3.8, implements an N-phases distributed parameters line model with lumped losses similar to the SPS distributed parameters line block. However, the ARTEMiS line allows decoupling big electrical systems into smaller subnetworks, thus reducing the total size of the precomputed state-space matrices. Therefore, ARTEMiS DPL can be used to distribute an electrical circuit over a cluster of PC cores by exploiting the inherent propagation delay of the line to split the circuit without affecting the dynamics of the system.

RT-LAB allows the user to insert ARTEMiS DPL block (same applies to ARTEMiS Stubline block) at the top-level of the model and connecting the physical ports of the block to the real-time subsystems (SM and SS). Also note that signals and ports of this block do not have to pass through an OpComm block.

One important fact about the ARTEMiS distributed parameters line block is the fact it does not initialize in steady-state, and hence, unexpected transient behavior at the beginning of the simulation may occur.

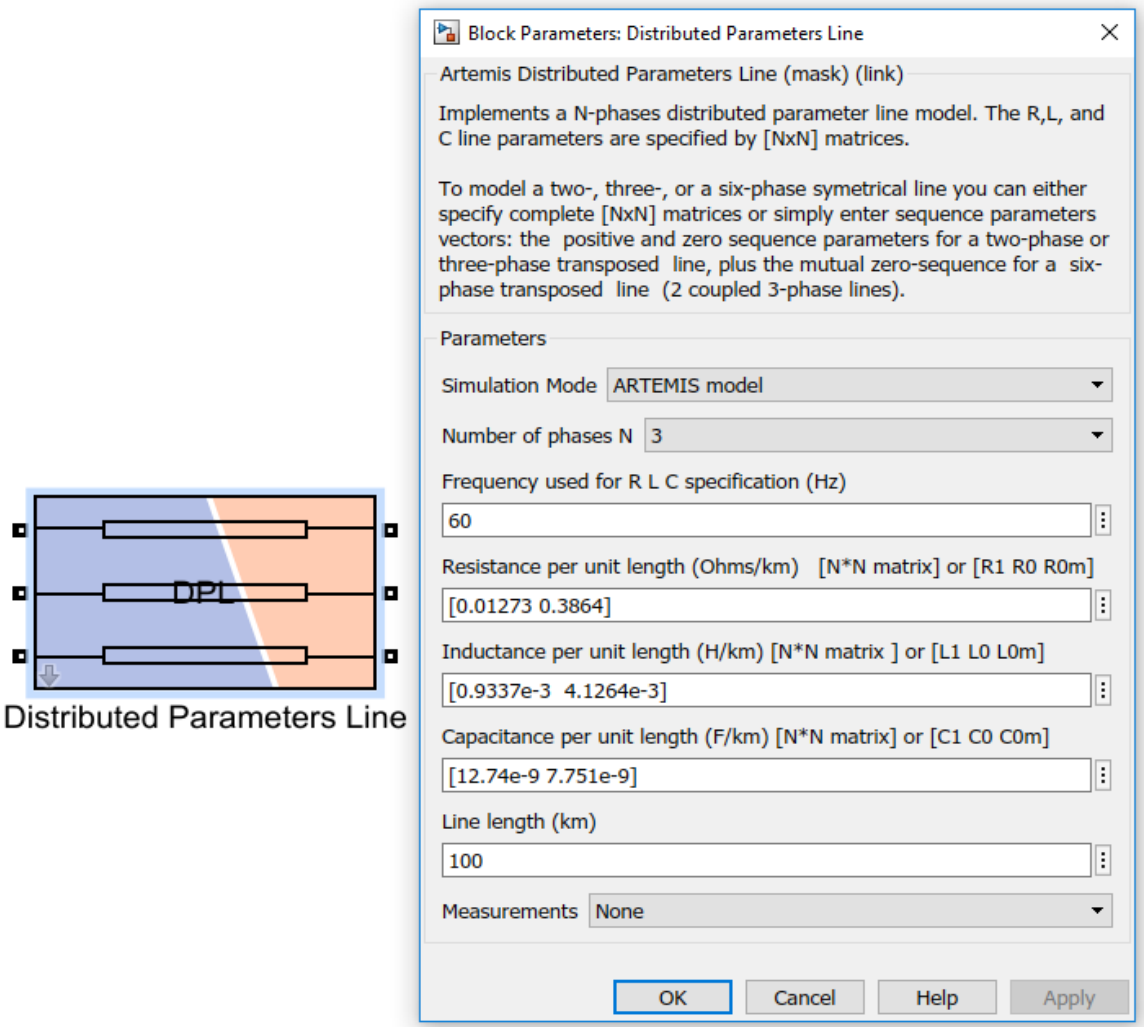

Figure 3.8

ARTEMiS Distributed Parameters Line Block and Parameters

Figure 3.8 also shows a screenshot of the ARTEMiS DPL block parameters which are listed and described briefly here:

- Simulation Mode: to define the mathematical model of the block (ARTEMiS or SPS).
- Number of phases N.
- Frequency used for RLC specifications (Hz): the default is 60 Hz.

- Resistance per unit length: entered as an N-by-N matrix or the sequence parameters in ohms/km.
- Inductance per unit length: entered as an N-by-N matrix or the sequence parameters in ohms/km.
- Capacitance per unit length: entered as an N-by-N matrix or the sequence parameters in ohms/km.
- Line length: in km.

3.4.3 ARTEMiS-SSN Nodal Interface Blocks (NIB)

The ARTEMiS-SSN Nodal Interface Blocks, shown in figure 3.9, are used to define nodes and SSN groups in a SimPowerSystems model. The NIB are available in 1ph, 2ph, 3ph, or 6ph and can have multiple ports (from 2 to 16).

The NIB port type must be defined correctly depending on the type of the State-space group connected: inductive type SSN groups require a V-type interface block, and capacitive type SSN groups require an I-type interface block.

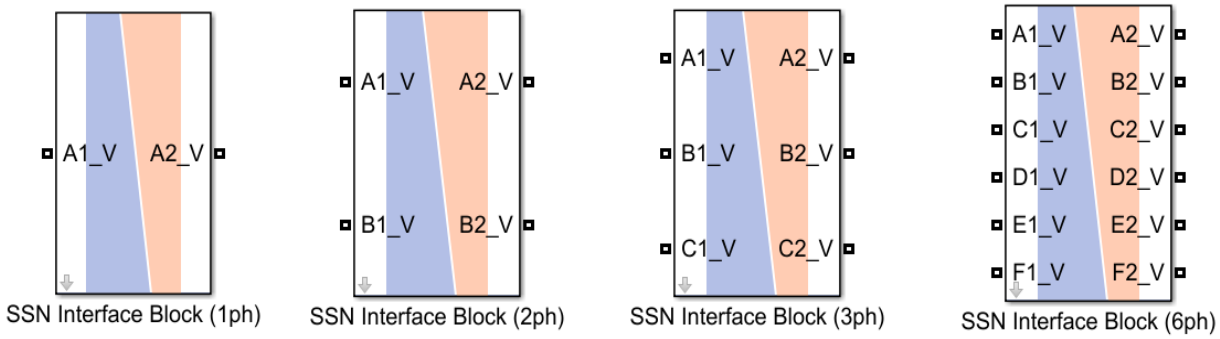

Figure 3.9

ARTEMiS-SSN Nodal Interface Blocks

CHAPTER 4

METHODOLOGY

This chapter presents a simple method for decoupling between RT-LAB subsystems that greatly improves on the Stubline method discussed in section 2.2.1. The steady-state error is almost eliminated by compensating for the added capacitance, and the transients introduced are mitigated through damping when found to be excessive. Furthermore, an analytical methodology for handling lines with mutual coupling - balanced or otherwise - is presented.

4.1 Description of Concept

In the Compensated Distributed Line Decoupling (CDLD) approach, we exchange an existing line in the distribution grid (Figure 4.1) in the location of dividing the subsystem with an ARTEMiS distributed parameter line (DPL) to provide a one-time step delay. The original line which is replaced may be modeled as a constant parameter π-line, or a distributed line.

The replacement decoupling line will have the same per unit length impedance of the original line $Z = R + jL$, except that a distributed capacitance, C_T, will be added to give the one-time step delay. To do this, we revisit the expression for a distributed line propagation delay from equation 2.1; for a lossless line this is,

$$T = \sqrt{LC} \tag{4.1}$$

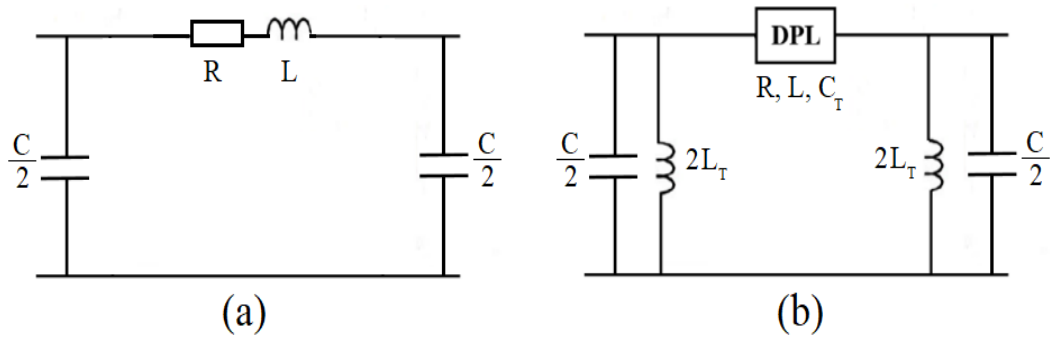

Figure 4.1

Original pi-line and the equivalent DPL with shunt compensation

For lines with resistance, the propagation delay may be computed by taking the imaginary component of the propagation constant $\sqrt{\mathrm{ZY}} = \sqrt{(\mathrm{R} + \mathrm{j}\omega\mathrm{L})\mathrm{j}\omega C} = \alpha + j\beta$. The time delay T is computed as β/ω, which works out to be,

$$T = \sqrt{\mathrm{LC}}\sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{1 + \left(\frac{R}{\omega L}\right)^2}} \tag{4.2}$$

Clearly, at high frequencies, (4.2) approaches (4.1), assuming frequency independency of parameters. Then, setting $T =$ one computational time step T_s requires tuning the capacitance to the value C_T, where

$$C_T = \frac{T_s^2}{L} \tag{4.3}$$

The capacitance C_T, added as the distributed line capacitance per unit length, is thus tuned to give a unit time step delay at higher frequencies. Equation (4.2) tells us that the propagation delay is larger at lower frequencies, where R becomes important. The original capacitance of the π-line at both ends is left untouched. However, an inductance L_T needs to be introduced to compensate for the added distributed capacitance C_T.

As an approximation, L_T will not be distributed but will be divided into two parts – each equal to $2L_T$ – on either side of the line as shown in figure 4.1 (b). The inductance value is obtained as matching the capacitive impedance at fundamental power frequency (60 Hz):

$$L_T = \frac{1}{\omega^2 \cdot C_T l} \tag{4.4}$$

Where l is the length of the distributed parameter line. The compensation can also be confined to one side of the line only with no discernable effect on accuracy.

Using compensation with the delay gives an overall model, which is equivalent to the original at the correct frequency of compensation. The equivalency here refers to the overall shunt and series impedance; however, the new line will introduce an additional time delay between its ports.

4.2 Practical Implementation of the Proposed Method

The three-phase π-line to be replaced with a distributed line may have uncoupled phases or mutually coupled phases. In the case of the former (which is typical of cable segments), the procedure is simple, since each phase is treated as a single phase line, and its added distributed capacitance and corresponding compensating reactance are calculated independent of other phases.

If the original line has mutual coupling (typical of overhead segments), the replacement becomes more complex. The problem is further complicated by the fact that the line will typically be unbalanced, as is the nature of distribution lines. In the classical Stubline method [6], the unbalanced line is dealt with by deducting an equal part of the feeder impedance from all three phases so that the deducted part represents a balanced line section that can be represented by a Stubline block.

Clearly, this is an inefficient approach, making an already short line even shorter for Stubline representation. The shorter line will need even more capacitance to affect the time delay, raising the potential for inadvertent transients. An alternative proposed in this work is described in what follows.

The idea is to extract all propagation modes for the unbalanced line and then to apply the CDLD method to the mode with the smallest propagation delay. For balanced (transposed) lines, the modes are extracted in the time domain using either the Clarke or Karrenbauer transformation [15], [16].

On the other hand, modes of unbalanced lines are uncoupled through an eigenvalue-based approach. The procedure, which is well documented in EMTP analysis [17], requires finding the eigenvalues and eigenvectors of the quantities ZY_T and Y_TZ, where $Y_T = j\omega C_T$ (it is recommended to use the highest natural frequency – the quarter length frequency, $f_1 = \frac{1}{4T} = \frac{1}{4T_s}$), and then uses these eigenvectors to obtain the modal impedance and admittance matrices Z_m and Y_{Tm}. The modal propagation matrix is then formed as $\sqrt{Z_m Y_{Tm}}$. For the unbalanced three-phase line, this will be a diagonal matrix with three modes.

The objective is to select Y_{Tm} (working from Y_T) such that the mode with shortest propagation delay time is equal to the computational time step T_s. Since we do not have Y_T (capacitive admittance matrix added to achieve this objective), we first need to start with an approximation for C_T by assuming a transposed (balanced) three-phase line. The impedance matrix Z is converted to the sequence domain using, for example, the Clarke Transform, $T_{0\alpha\beta}$, which gives the symmetrical components in the time domain. Then,

$$Z_{012} = {T_{0\alpha\beta}}^{-1} \cdot Z \cdot T_{0\alpha\beta} \tag{4.5}$$

Any off-diagonal elements in Z_{012} are discarded, leaving only diagonal elements representing the positive, negative and zero sequence components, Z_1, Z_2 and Z_0. If the line was balanced, then these impedances are identical to those obtained through the symmetrical components transformation.

In this case, there will be some difference and we search for whichever impedance has the lowest inductive component. Assume that this is the positive sequence inductance, L_1. The propagation delay for positive sequence wave can then be set to T_s by replacing L in (4.3) with the positive sequence inductance, L_1.

The value C_T, thus obtained represents a first approximation for the added capacitance of the distributed line. This value is applied for all three sequence components as a diagonal compensating sequence matrix:

$$C_{012} = \begin{bmatrix} C_T & 0 & 0 \\ 0 & C_T & 0 \\ 0 & 0 & C_T \end{bmatrix} \tag{4.6}$$

The propagation delay for the remaining modes (zero and negative sequence) will be higher since both L_0 and L_2 are greater than L_1. The capacitance matrix C_{012} can now be transformed into the phase domain using the inverse transform,

$$C' = T_{0\alpha\beta} \cdot C_{012} \cdot {T_{0\alpha\beta}}^{-1} \tag{4.7}$$

Actually, $C' = C_{012}$ here since it is a scalar matrix, unaffected by orthogonal transformations. The diagonal admittance $Y' = j\omega C'$ represents a first approximation for the required distributed line added capacitive component. Next, we make use of the matrix ESMM (Eigenvalues of a Scalar Multiple of a Matrix) property [18], which states that if a square matrix A has eigenvalues λ, then kA, where k is a scalar multiplier will have eigenvalues $k\lambda$, while the eigenvectors remain unchanged.

Let us now use modal analysis and set the admittance matrix for the distributed line to be $Y = k \cdot Y'$, where Y is the correct capacitance matrix for the unbalanced line, and k is some scalar multiplier used here as a correction factor. We then proceed to find the eigenvectors T_v and T_i corresponding to the complex products $Z \cdot Y$ and $Y \cdot Z$. Since we do not yet have a value for k, we note that these eigenvectors will be the same as those obtained using $Z \cdot Y'$ and $Y' \cdot Z$. We then use T_v and T_i to find the modal matrices Z_m and Y_m according to,

$$Z_m = {T_v}^{-1} \cdot Z \cdot T_i \tag{4.8}$$

$$Y'_m = {T_i}^{-1} \cdot Y' \cdot T_v \tag{4.9}$$

From (4.8) and (4.9),

$$Z_m \cdot Y'_m = {T_v}^{-1} Z \cdot Y' T_v \tag{4.10}$$

Multiplying (4.10) by the scalar k gives:

$$k(Z_m \cdot Y'_m) = {T_v}^{-1} Z \cdot (kY') T_v = {T_v}^{-1} Z \cdot Y T_v \tag{4.11}$$

Where $k(Z_m \cdot Y'_m)$ is a diagonal matrix, and its square root $\sqrt{k}\sqrt{Z_m Y'_m}$ contains the three propagation constants for the unbalanced line. Using equation (4.3), with all parameters replaced with modal counterparts, we simply set the lowest mode delay to one-time step, and find the value for k.

4.3 Additional Considerations for Very Short Lines

Very short lines are encountered at the lower primary distribution voltages, e.g. 2.3 – 13.8 kV. These lines are typically 125 – 300 ft. in span but can vary depending on terrain. Owing to the very low wave propagation times for these type of lines, the capacitance C_T necessary to increase the propagation delay to unit time step will be quite large.

This large capacitance in turn is observed to cause poorly damped oscillatory transients during switching events, particularly evident in the voltage waveform. Because the compensating inductor L_T is only tuned to compensate $C_T l$ at power frequency, it remains largely ineffective at the much higher switching frequencies.

Obviously it is impossible to compensate for all frequencies, so the next best approach is to attempt mitigating the oscillatory component through damping. Any damping resistor however will need to be introduced through a coupling capacitor, so that it does not unduly affect steady-state conditions.

Let us imagine that, for switching analysis, the phase-ground Thévenin impedance at the terminus of the distributed line, consists of a capacitor to ground C in parallel with inductance L. The capacitance will be mainly contributed by the added line component $C_T l$. We are spared from a deterministic calculation of L, if we are able to observe the dominant underdamped frequency of oscillation of the phase voltage (ω_0).

Figure 4.2 shows the LC network in parallel with a damping resistor R introduced through coupling capacitor C'.

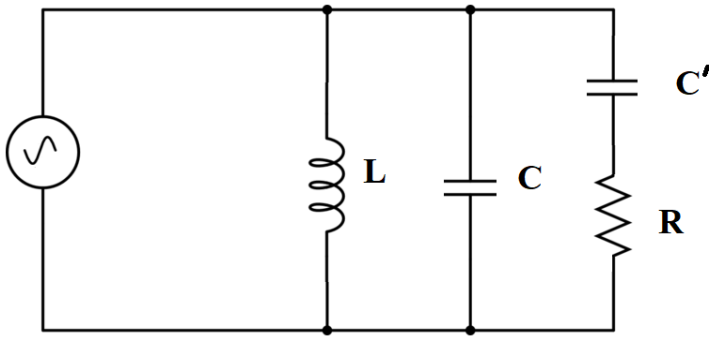

Figure 4.2

System LC network in parallel with RC' damping network

The natural oscillatory response in the complex frequency domain for such a network is obtained through the expression,

$$s^3 + \frac{1}{R}\left(\frac{1}{C} + \frac{1}{C'}\right)s^2 + {\omega_0}^2 s + \frac{{\omega_0}^2}{RC'} = 0 \tag{4.12}$$

Where,

$$\omega_0 = \frac{1}{\sqrt{\mathrm{LC}}} \tag{4.13}$$

Equivalent equation for a general third order circuit

$$s^3 + (\lambda + 2\zeta\omega_n)s^2 + (2\lambda\zeta\omega_n + {\omega_n}^2)s + \lambda{\omega_n}^2 = 0 \tag{4.14}$$

Comparing second terms

$$\lambda + 2\zeta\omega_n = \frac{1}{R}\left(\frac{1}{C} + \frac{1}{C'}\right) \tag{4.15}$$

$$\lambda + 2\zeta\omega_n = \frac{a}{RC'} \tag{4.16}$$

Where:

$$a = C'\left(\frac{1}{C} + \frac{1}{C'}\right) = 1 + \frac{C'}{C} \tag{4.17}$$

Let $x = \frac{\omega_0}{\omega_n}$. Multiply (4.16) by x

$$\lambda x + 2\zeta\omega_0 = \frac{a}{RC'}x$$

Or

$$\frac{\lambda}{\omega_0}x + 2\zeta = \frac{a}{\omega_0 RC'}x \tag{4.18}$$

Comparing third terms

$$2\lambda\zeta\omega_n + {\omega_n}^2 = {\omega_0}^2$$

$$\frac{2\lambda\zeta}{\omega_0}\left(\frac{\omega_0}{\omega_n}\right) + 1 = \left(\frac{\omega_0}{\omega_n}\right)^2$$

$$\frac{2\lambda\zeta}{\omega_0}x + 1 = x^2 \tag{4.19}$$

Multiply (4.18) by 2ζ and subtract (4.19)

$$4\zeta^2 - 1 = \frac{2\zeta a}{\omega_0 RC'}x - x^2 \tag{4.20}$$

Comparing fourth terms

$$\lambda{\omega_n}^2 = \frac{{\omega_0}^2}{RC'}$$

$$\lambda = \frac{x^2}{RC'} \tag{4.21}$$

Substitute λ in (4.19)

$$\frac{2\zeta}{\omega_0 RC'}x^3 + 1 = x^2 \tag{4.22}$$

Multiply (4.20) by x^2 and (4.22) by a and add

$$4\zeta^2x^2 - x^2 + a = -x^4 + ax^2$$

or

$$4\zeta^2x^2 = a(x^2 - 1) - x^4 + x^2 \tag{4.23}$$

Taking derivatives of (4.23) with respect to damping resistance R

$$8\zeta\frac{d\zeta}{dR}x^2 + 8\zeta^2x\frac{dx}{dR} = 2ax\frac{dx}{dR} - 4x^3\frac{dx}{dR} + 2x\frac{dx}{dR} \tag{4.24}$$

At maximum damping $\frac{d\zeta}{dR} = 0$,

Then

$$4\zeta^2x = ax - 2x^3 + x \tag{4.25}$$

Multiply (4.25) by x and subtract from (4.23)

$$0 = -a + x^4 \tag{4.26}$$

Thus, at maximum damping

$$x = \frac{\omega_0}{\omega_n} = \sqrt[4]{a} \tag{4.27}$$

Maximum damping is found by substituting in (4.25)

$$4\zeta_{max}^2 = a - 2\sqrt{a} + 1$$

Or

$$\zeta_{max} = \frac{1}{2}\left(\sqrt{a} - 1\right) \tag{4.28}$$

And resistance necessary to achieve maximum damping may be found using (4.22)

$$\frac{\sqrt{a} - 1}{\omega_0 R C'} a^{0.75} + 1 = \sqrt{a}$$

Or

$$R = \frac{a^{\left(\frac{3}{4}\right)}}{\omega_0 C'} \tag{4.29}$$

Although equations (4.28) and (4.29) may be used to control the transient oscillation effectively, we are also constrained to represent steady-state conditions accurately. The compensating inductance L_T is now shunted by R in series with C'. We thus need to change it to an inductance L'_T which, together with R and C' offers a fair reproduction of L_T at power frequency. The combined compensating impedance is expressed as,

$$\frac{j\omega L'_T (R - j\frac{1}{\omega C'})}{j\omega L'_T + R - j\frac{1}{\omega C'})} = \frac{j\omega L'_T (\omega R C' - j1)}{R\omega C + j(\omega^2 L'_T C' - 1)} \tag{4.30}$$

The magnitude and phase of this impedance as a function of frequency are shown in Figure 4.3. The peak of the magnitude has a location determined by $\omega_{peak} = \frac{1}{\sqrt{L'_T C'}}$, selected close to power frequency, at 100 Hz or less.

This frequency separates the behavior of the circuit to act as an inductor (phase angle ≈ 90°) in the vicinity of power frequency and as an RC circuit in the domain of switching frequencies. All that remains is to set the magnitude of the impedance at power frequency to be equal to $\omega_{60}L_T$, where $\omega_{60} = 2\pi 60$, and to select the value of the resistance that gives maximum damping. For clarity, the three equations required for design of the circuit are repeated as,

$$L'_T C' = \left(1/\omega_{peak}\right)^2$$

$$L'_T \frac{\sqrt{(\omega_{60}RC')^2 + 1}}{\sqrt{(\omega_{60}RC')^2 + (\omega_{60}{}^2 L'_T C' - 1)^2}} = L_T$$

$$R = \frac{a^{\left(\frac{3}{4}\right)}}{\omega_0 C'} \tag{4.31}$$

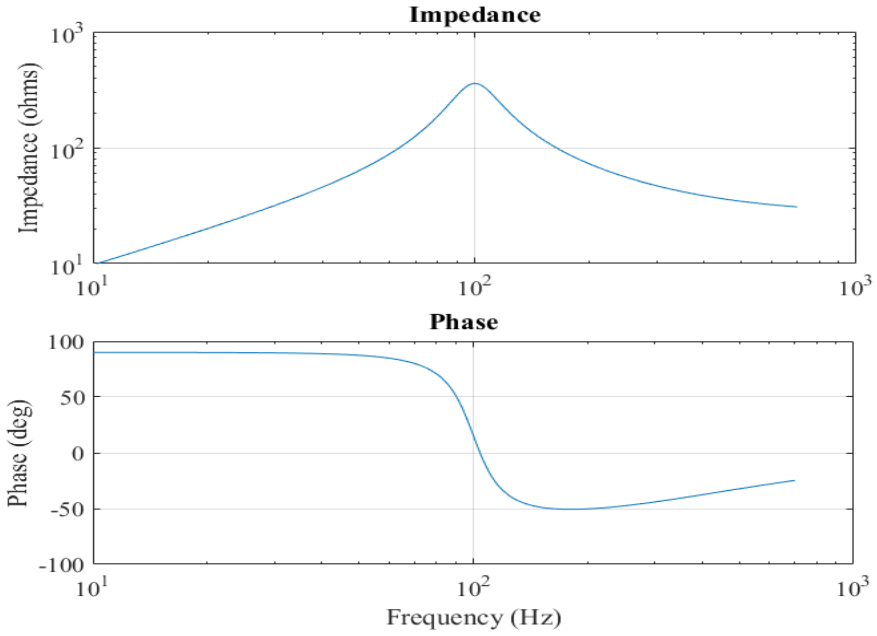

Figure 4.3

Frequency characteristics of the compensating RLC impedance

The undamped oscillatory frequency ω_0 is determined by observing the network behavior subject to the required switching event, and prior to applying the RC damping circuit. We also find that pushing ω_{peak} closer to power frequency of 60 Hz gives a more effective damping circuit, although at the expense of a greater departure of the steady state angle from 90°.

The set of equations (4.31) are easily solved with a numerical solver, using starting values $L'_T = L_T$, $C' = C_T l$, and any positive nonzero value for R.

The damping resistors are incrementally designed and applied at the terminals of the decoupling points, starting with the first (closest to the source), observing the response, and then implementing the second, and so on. It is found that usually two or three damping resistors are sufficient, as further into the distribution system the oscillatory frequency is low enough to be adequately damped by the system loading.

Figure 4.4 shows snapshots of a magnified portion of the waveform obtained from the IEEE 123-bus distribution system (described in section 5.2), when subject to a line-to-line fault at node 23. Undamped frequency f_0 was measured at the receiving terminals of the decoupling lines and used to incrementally design damping resistors at the respective terminals. Figure 4.4(b) shows the undamped frequency at the terminal of the first decoupling line, measured as 725 Hz, and used to design a damping circuit giving $\zeta_{max} = 0.124$ at this point. Upon application of this, and two other damping resistors at second and third decoupling points, the response was improved as shown in Figure 4.4(c). The response of the original un-decoupled system, for comparison is shown in Figure 4.4(a).

Damping does not eliminate the oscillation in the voltage recovery, but reduces it considerably. The current waveform typically displays no discrepancy compared to original, as is shown in chapter 5.

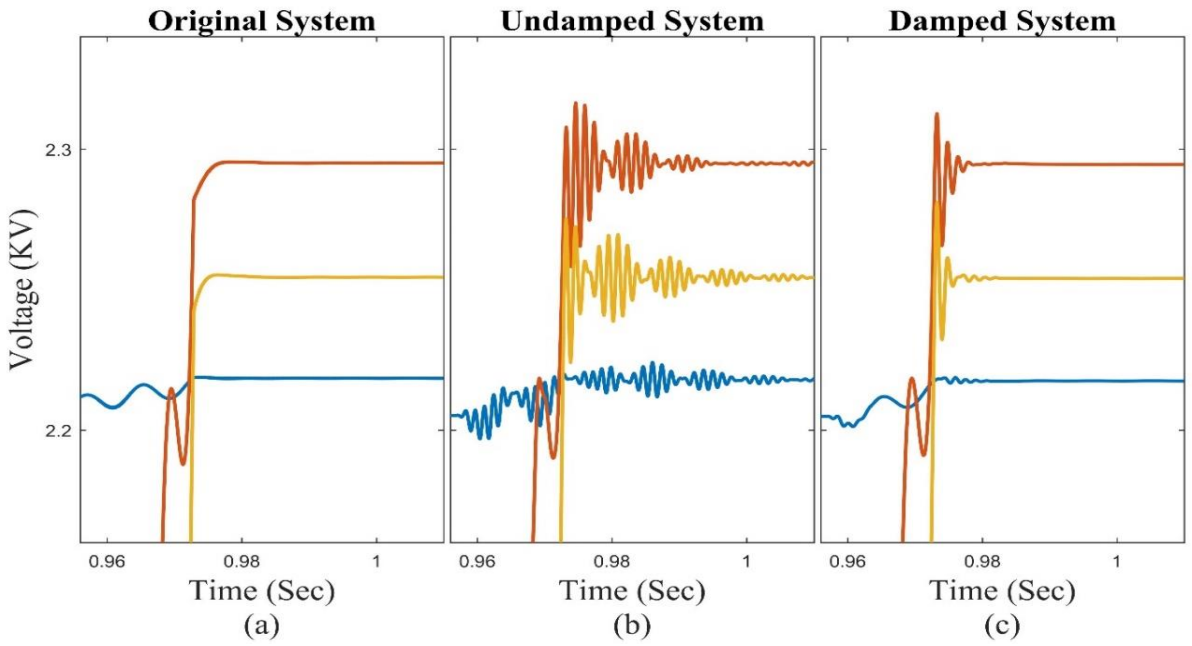

Figure 4.4

A magnified portion of the recovery voltage after a line-line fault

Figure 4.5 shows the final CDLD model when used to decouple a distribution feeder with extremely short lines by adding a damping circuit to the DPL terminal.

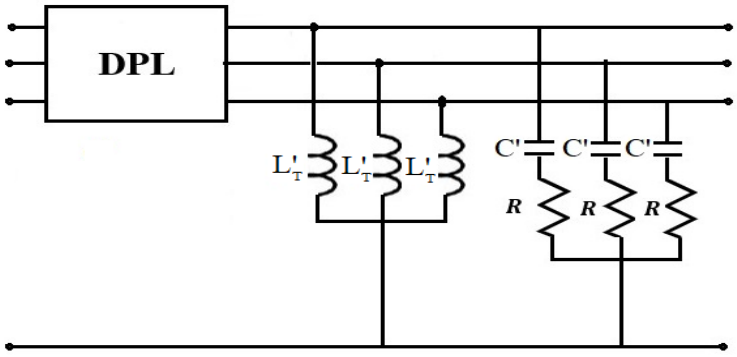

Figure 4.5

Final CDLD Configuration with Damping Circuit

4.4 Integration with SSN (SSN-CDLD)

The method may be used alone, or in conjunction with the SSN method, where it frees up the SSN from the time congestion due to the build-up of the nodal admittance size.

This SSN-CDLD method allows the SSN model to be decoupled into any number of subsystems while maintaining to a large degree the accuracy of the original model, and hence, removing computational bottlenecks. Five or six cores can be allocated to each subsystem and that will increase the parallelism gains for the SSN method with large distribution networks.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

To examine the proposed CDLD method, three IEEE distribution systems were decoupled, built and executed in real time with RT-LAB [14]. The real-time digital simulator (target) used in this study is eMEGAsim OP5600, developed by OPAL-RT Technologies.

5.2 IEEE Test Feeders

5.2.1 IEEE 34 Node Distribution Feeder

This feeder is a real Arizona-based feeder (Figure 5.1). The nominal voltage of the feeder is 25 kV. It is characterized by long line segments, lightly unbalanced loads (spot and distributed loads), two in-line regulators, shunt capacitors and an in-line transformer to step-down the voltage to 4.16 kV for a small portion of the feeder [5]. Spot loads are located at the node, while distributed loads are considered to be connected at the middle of line segments.

5.2.2 IEEE 37 Node Distribution Feeder

This feeder, rated at 4.8 kV, is situated in California (Figure 5.2). All line segments of the feeder are underground cables with spot loads, which are extremely unbalanced. The substation voltage regulator consists of two single phase units connected in open delta [5].

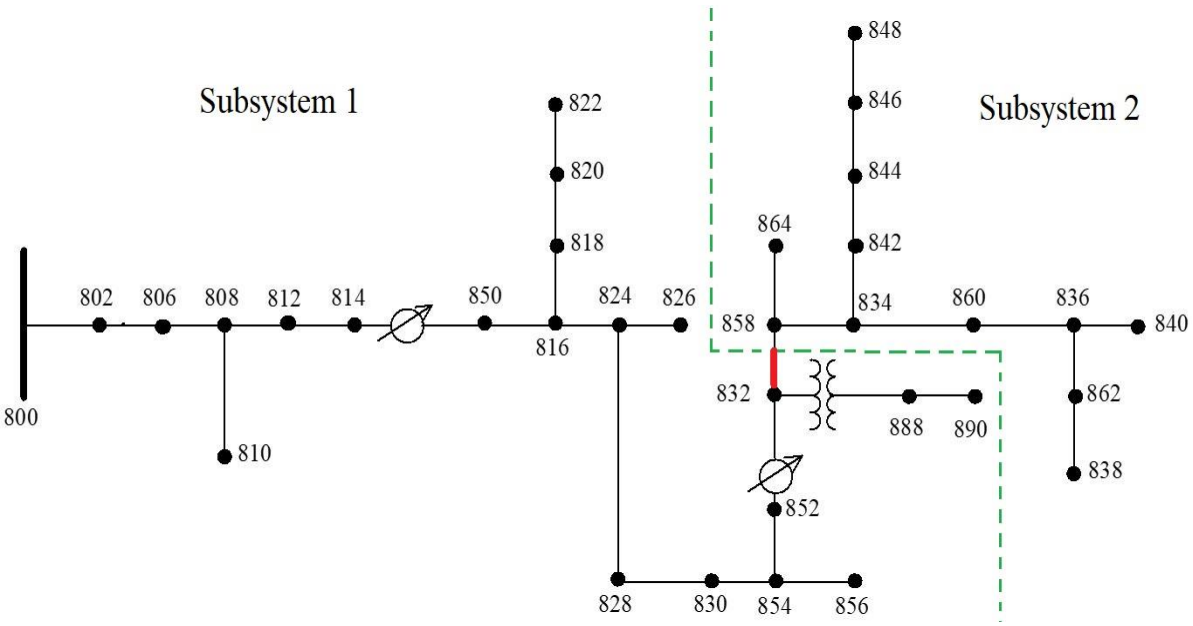

Figure 5.1

IEEE 34 Node Test Feeder

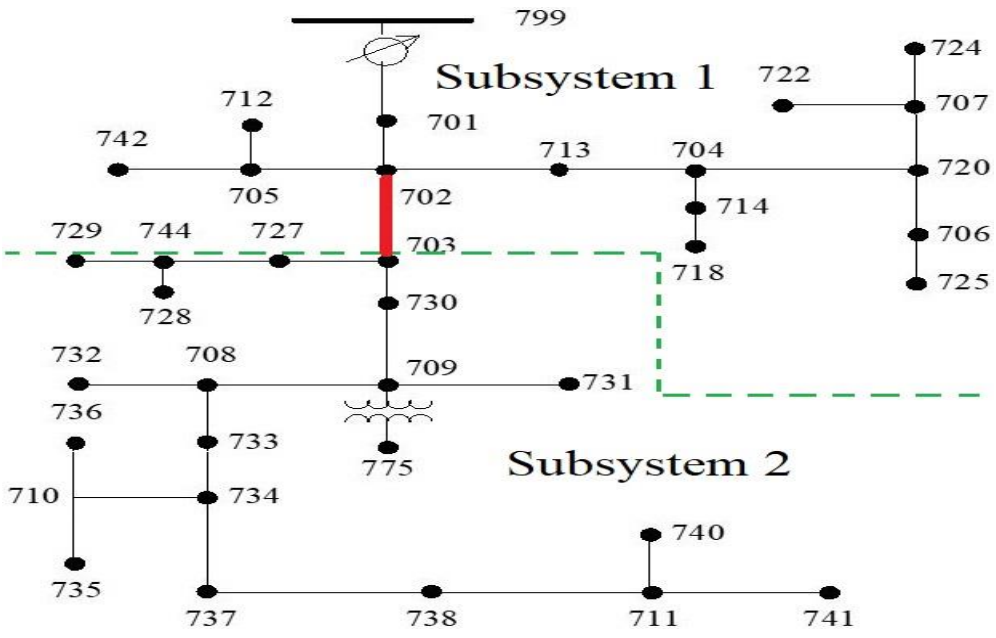

Figure 5.2

IEEE 34 Node Test Feeder

5.2.3 IEEE 123 Node Distribution Feeder

This comprehensive test feeder (Figure 5.3) operates at a nominal voltage of 4.16 kV and it is characterized by both overhead and underground line segments, unbalanced loading (spot loads), shunt capacitor and three step-type voltage regulators. There are switches in the feeder to provide alternate paths of power-flow [5].

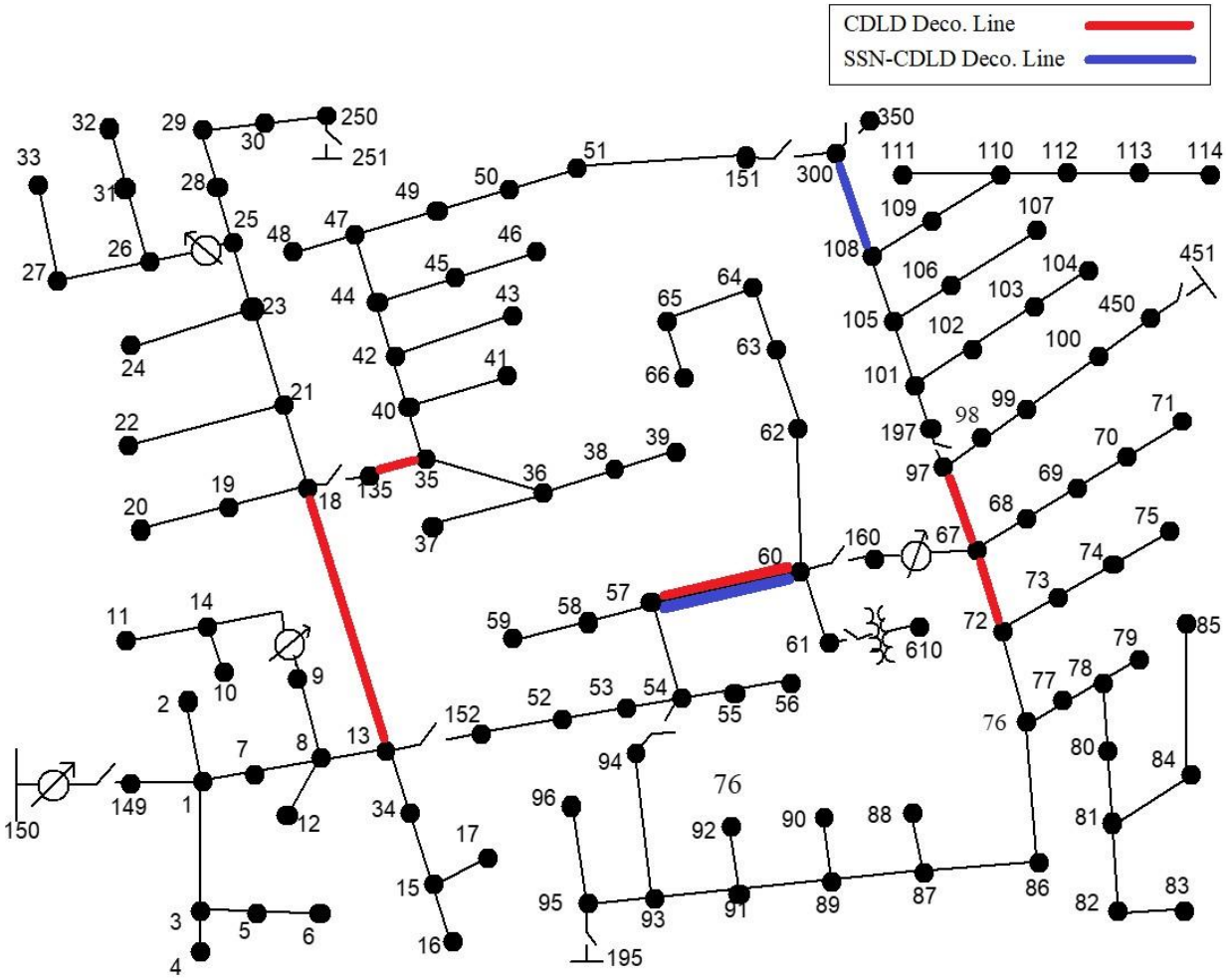

Figure 5.3

IEEE 123 Node Test Feeder

5.3 CDLD Method Validation

The IEEE 34 and 37 node systems feeders were used to test the performance of proposed method and compare it to the original uncoupled system (offline) for both steady-state and transient behaviors. Moreover, the IEEE 123 node feeder was used to compare the CDLD, SSN and a combination of the two (SSN-CDLD). The OPAL-RT ARTEMiS solver (Art5) was used for the IEEE 34 and 37 node systems while the SPS solver was used for the IEEE 123 feeder because the high number of switches in the regulating transformer does not permit using Art5. Finally, the performance of the three methods in terms of computational speed is also discussed.

5.3.1 IEEE 34 Node Distribution Feeder Results

This feeder was built and executed in real-time using two cores with a step size of 40 µs. Because of the distributed load on line 832-858, only the first half segment (2450 ft.) was used to decouple the feeder into two subsystems using the CDLD procedure as shown in figure 5.1. Damping was found to be unnecessary as this feeder contains long line sections.

The transient response of the decoupled system was tested by applying both ground and phase faults for 3 cycles. However, the steady-state RMS voltage and current error values were measured after the fault has been cleared and it is summarized in table 5.1. These results were taken from the percentage error graph as shown in figure 5.4 at node 834.

Table 5.1

IEEE 34 System Steady State Performance

Location	RMS Voltage error (%)	RMS Current Error (%)
Node 816	0	0.04
Node 834	0.02	0.05

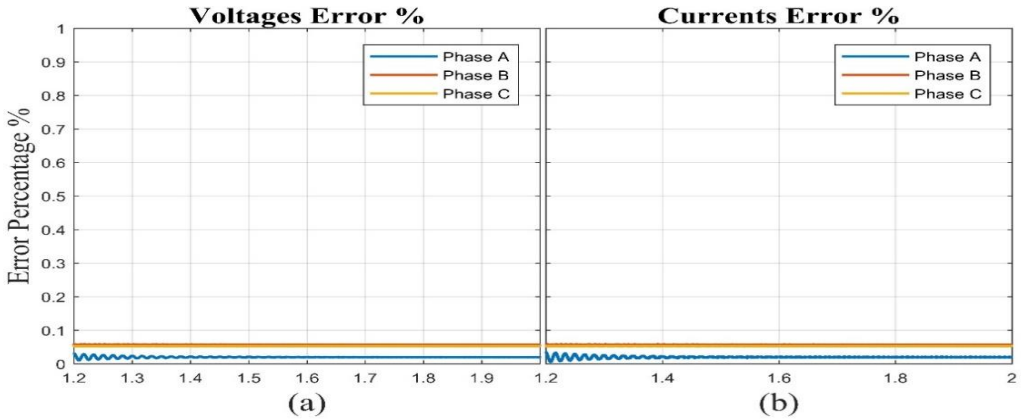

Figure 5.4

Voltage and Current Steady-State Error Percentage for IEEE 34 System

Figures 5.5 and 5.6 show the CDLD results compared to the original uncoupled system for line to ground (phase A) and line to line (phase B-C) faults respectively. The figures show that the two systems transient behavior are identical for both voltage and current.

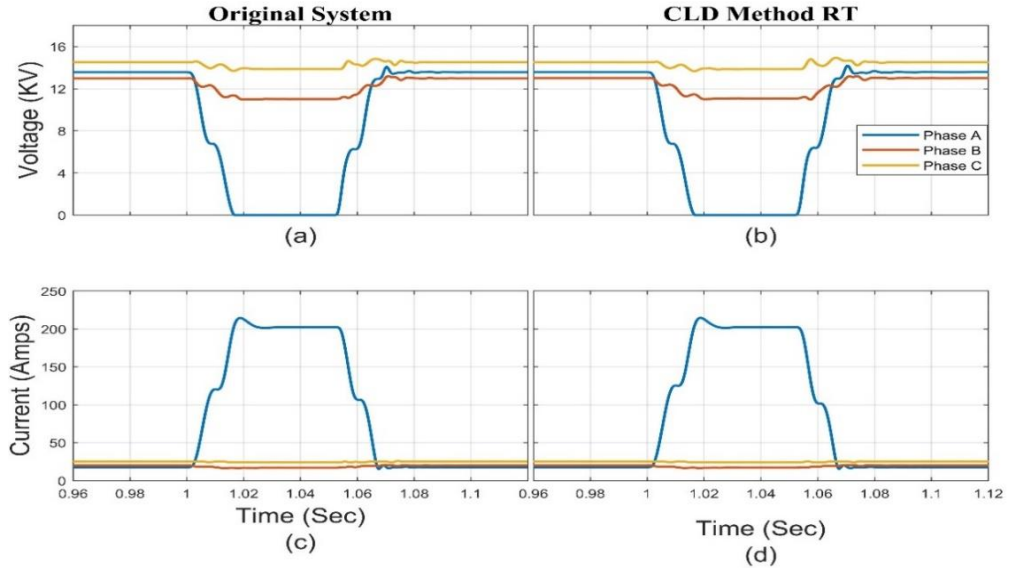

Figure 5.5

RMS Voltage and Current for line to ground (phase A) fault at Node 834

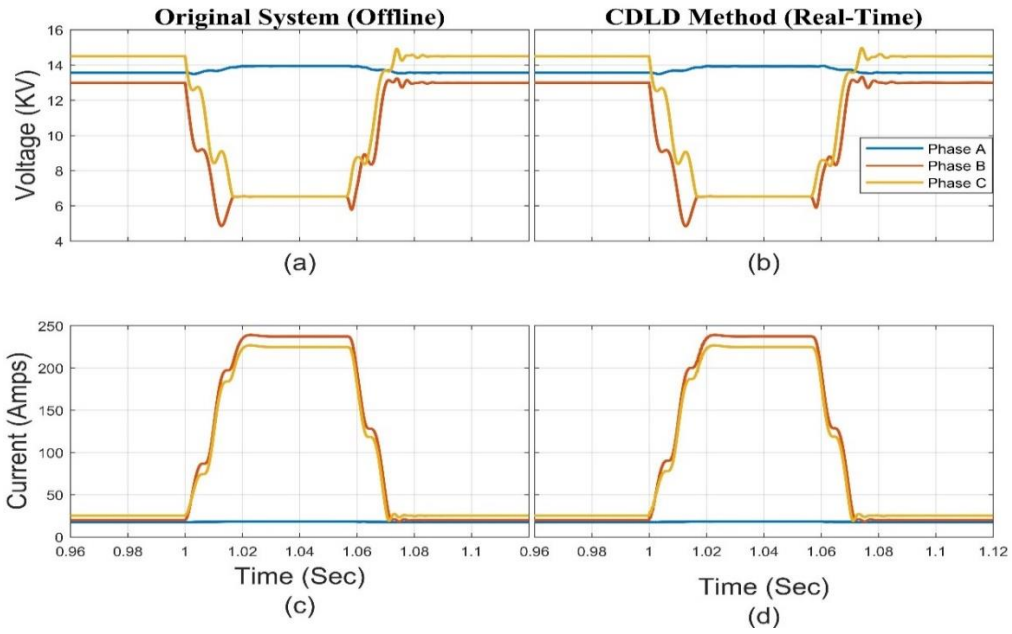

Figure 5.6

RMS Voltage and Current for line to line (phase B-C) fault at Node 834

5.3.2 IEEE 37 Node Distribution Feeder Results

This feeder has also been divided into two subsystems (2 cores) and the simulation of the whole distribution network was realized with a time step of 40 µs.

The line segment between nodes 702-703, which is 1320 ft. long, was used to decouple the two subsystems using the CDLD method as shown in figure 5.2. However, each phase was handled independently as the line segments of the feeder have uncoupled phases (underground cables).

A damping circuit was connected to node 703 (decoupling point terminal on 2nd subsystem) to damp the small oscillation found in the recovery voltage. The steady-state RMS voltage and current values were then measured on the two subsystems and compared to the offline simulation in table 5.2.

Table 5.2

IEEE 37 System Steady State Performance

Location	RMS Voltage error (%)	RMS Current Error (%)
Node 713	0.01	0
Node 703	0.02	0.02

The decoupled system transient behavior was tested by applying both ground and phase faults for 3 cycles. Figures 5.7 and 5.8 show the CDLD results compared to the original uncoupled system for line to ground (phase A) and line to line (phase B-C) faults respectively. The figures show that the two systems transient behavior are identical for both voltage and current.

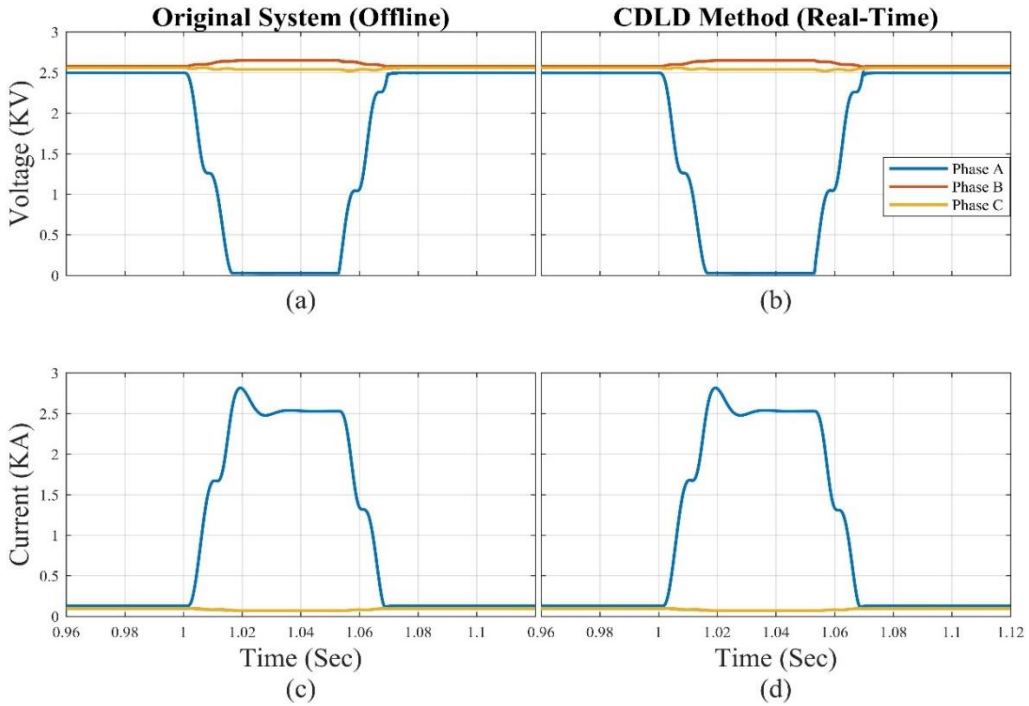

Figure 5.7

RMS Voltage and Current for line to ground (phase A) fault at bus 703

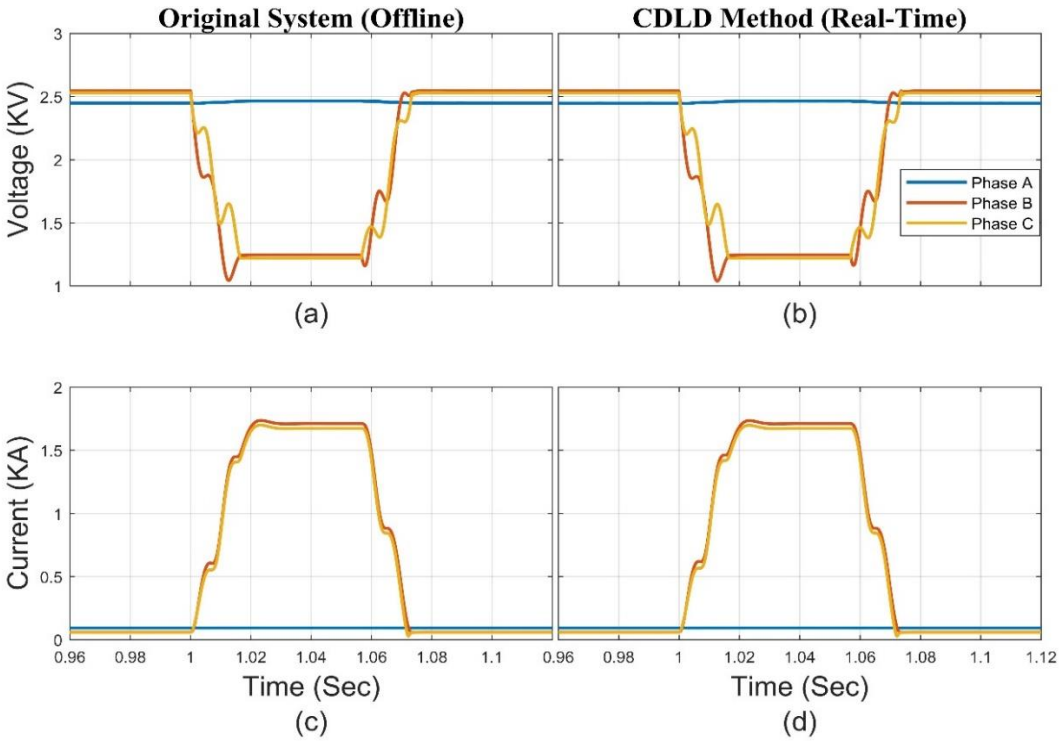

Figure 5.8

RMS Voltage and Current for line to line (phase B-C) fault at bus 733

5.3.3 IEEE 123 Node Distribution Feeder Results

The IEEE 123 node system is a larger and more complex distribution feeder compared to the IEEE 34 and 37, and with lines of much shorter length. Therefore, it is used to compare between the decoupling methods discussed in this work: CDLD, SSN and SSN-CDLD. The simulation of the feeder using the three methods was realized with a time step of 50µs using 5 cores.

For the CDLD model, lines 13-18, 135-35, 57-60, 67-72 and 67-97 were replaced by a distributed parameter line (ARTEMiS DPL) to decouple the five subsystems using the CDLD

procedure as shown in figure 5.3. Three damping circuits (peak at 100 Hz) were connected incrementally to nodes 18, 35 and 60 to damp the oscillations found in the recovery voltage at each of the locations. For the SSN-CDLD model, lines 57-60 and 108-300 were used to separate the SSN model into two subsystems using the CDLD procedure as shown in figure 5.3. A damping circuit was introduced at node 60 to damp the small oscillatory behavior found in the voltage waveform.

A trapezoidal SPS (SimPowerSystems™) solver is used for the CDLD model, which contains an SPS-based OLTC (on-load-tap-changer), because the number of switches per subsystem will exceeds the ARTEMiS solver limit (15 switch). However, both SSN and SSN-CDLD models were built and executed in real-time using ARTEMiS Art5 solver, which has its own custom-coded OLTC [10]. This SSN OLTC model has a variable secondary inductance and resistance as well as turn-ratio which can be modified during real-time simulation.

All three methods were simulated and steady-state voltage and current (RMS) values were measured at several points throughout the feeder and compared to the original uncoupled system (offline). Results are summarized in table 5.3.

Table 5.3

IEEE 123 System Steady State Performance

Location	**RMS Voltage error (%)**			**RMS Current Error (%)**		
	CDLD	**SSN**	**SSN-CDLD**	**CDLD**	**SSN**	**SSN-CDLD**
Node 7	0	0.01	0.01	0.38	0.44	0.37
Node 18	0.04	0.02	0.02	0.4	0.55	0.55
Node 35	0.07	0.02	0.02	0.07	0.03	0.03
Node 60	0.03	0.01	0.02	0.02	0.79	0.80
Node 89	0.03	0.46	0.47	0.03	0.46	0.48

For the CDLD method, the small steady-state current error (0.38% and 0.4%) at beginning of the feeder is mainly due to the three introduced damping circuits which are connected to nodes 18, 35 and 60. The SSN error, on the other hand, is likely caused by the use of the SSN-specific OLTC transformer, in comparison to the SPS-based OLTC transformer used in the original system.

The proposed CDLD and SSN-CDLD transient behaviors were tested by applying both ground and phase faults for 3 cycles. Figures 5.9 and 5.10 show their results compared to the original system (offline) for line to ground (phase A) and line to line (phase B-C) faults respectively.

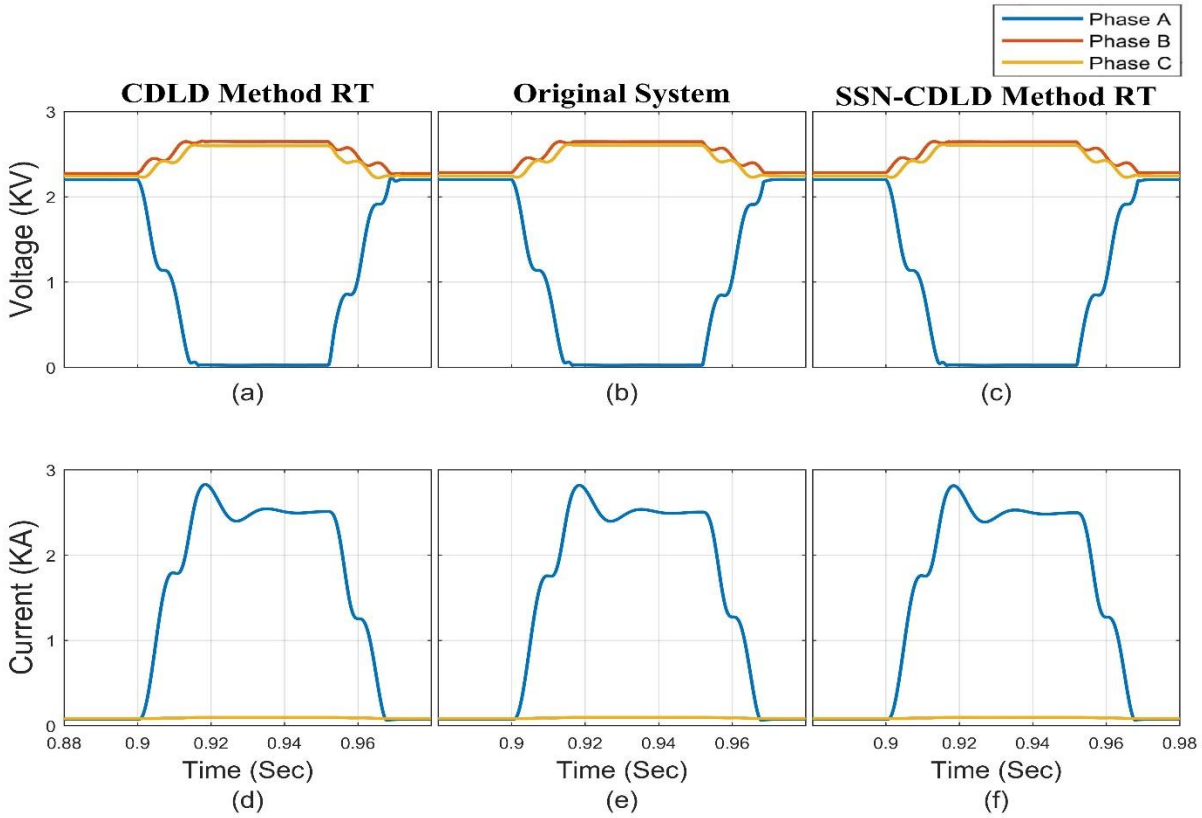

Figure 5.9

RMS Voltage and Current for line to ground (phase A) fault at node 44

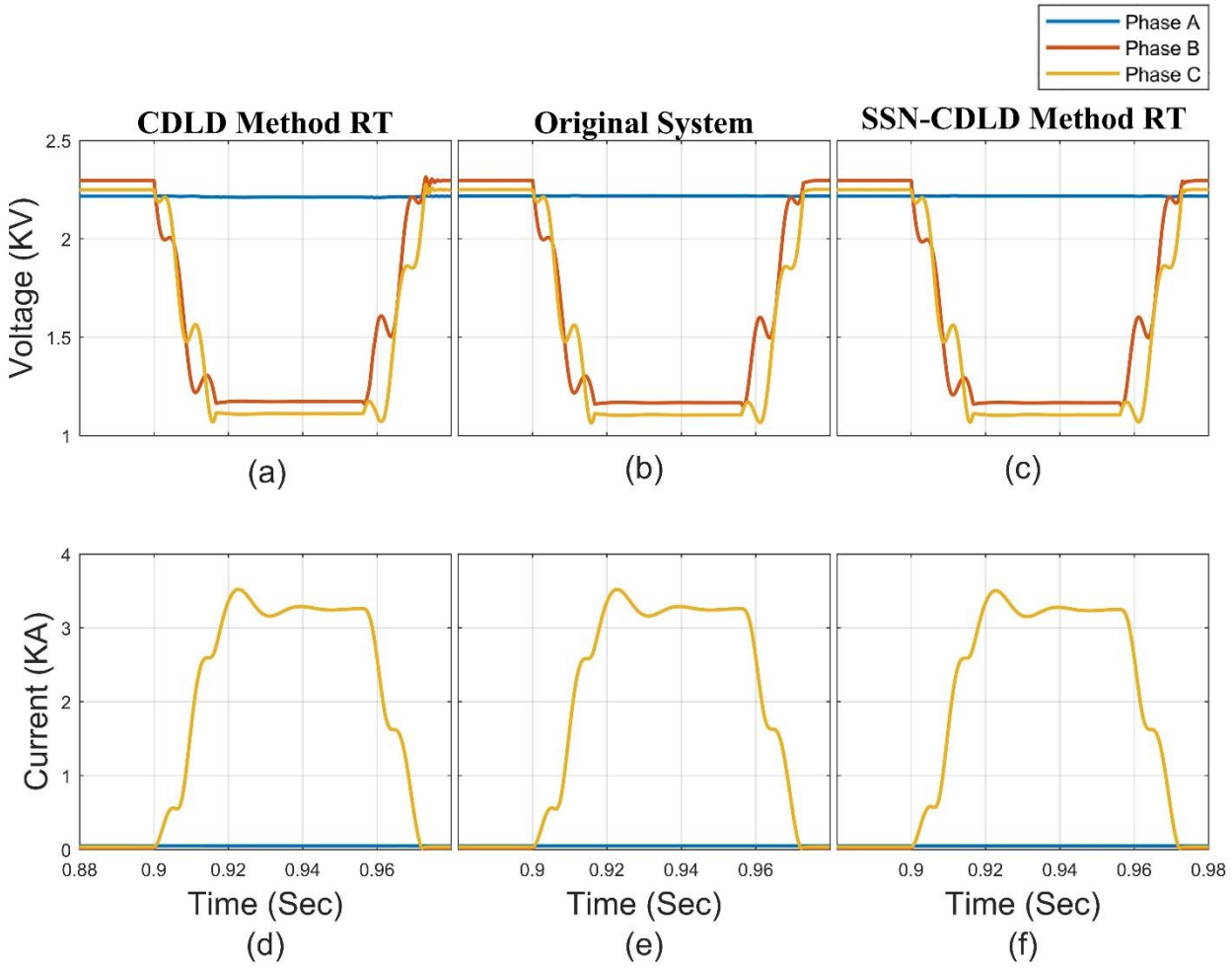

Figure 5.10

Voltage and Current (RMS) for line to line (phase B-C) fault at node 23

From the figures, it can be seen that the RMS voltage and current waveforms of the decoupled systems are almost identical to the original uncoupled system. The only discernible difference is the slight notch in the recovery voltage in figure 5.10 (a) (magnified in chapter 4, figure 4.4 (c)). On the other hand, the current displays no noticeable difference compared to original. The new decoupling method therefore shows an adequate performance for transient behavior on the three IEEE distribution test feeders.

5.4 Real-Time Performance Comparison of Decoupling Methods

Table 5.4 below shows the performance of the three IEEE systems running on the OP5600 real-time simulator, in terms of mean computation time (MCT) and number of overruns.

Table 5.4

Mean Computation Times for the IEEE Systems in Real Time

Systems	**Method**	**Step Size (μs)**	**No. of cores**	**MCT (μs)**	**overruns**
IEEE 34 Node System	CDLD	40	2	12.0	0
	SSN			16.6	0
IEEE 37 Node System	CDLD	40	2	12.1	0
	SSN			16.6	0
IEEE 123 Node System	CDLD	50	5	24.1	1
	SSN			43.4	2
	SSN-CDLD			20.8	0

As shown in table 5.4, the CDLD method is observed to require significantly less computation times than SSN for all IEEE test systems. However, both suffer from one or two occasional overruns (jitter) when used to decouple the 123 node system.

SPS-based solver was used for the CDLD model because it contains more than 15 switches per subsystem. The single overrun in the CDLD model was traced to the instant of tap changing, where the SPS-based solver is forced to re-compute the state-space matrices because of the switching event which results in overshoots.

Sometimes when the model computation time is too close to the real-time limit, jitter could cause random overruns as it is the case for the IEEE 123 SSN model (2 overruns). One would assume that allocating more cores (6 cores) to the SSN model will also solve the overrun issue. On the contrary, the performance will become even worse as table 5.5 reveals (39,826 overruns). However, adding more cores, when using the CDLD method, results in a lower computational time and eliminates the occasional overruns (overshoots).

Table 5.5

Mean Computation Time in μs vs No. of Cores for the IEEE 123 System

No. of Cores	**CDLD**		**SSN**		**SSN-CDLD**	
	MCT	**Overruns**	**MCT**	**Overruns**	**MCT**	**Overruns**
1	-	-	-	39,988	-	-
2	-	-	-	39,988	36.87	0
3	-	-	-	39,988	29.09	0
4	-	13,333	45.36	31	23.07	0
5	24.05	1	43.42	2	20.82	0
6	17.86	0	-	39,826	20.29	0

The SSN-CDLD can overcome the disadvantages of both the SSN and CDLD methods. As shown in table 5.5, parallelism gains increase with the number of allocated cores, and hence, very large distribution grids could be simulated in real-time using the SSN-CDLD method. In addition, this method uses the ARTEMiS solver which precomputes all state-space matrices before the real-time execution. This reduces the computation time and eliminates the occasional overruns even when running the 123 node system using only 2 cores.

CHAPTER 6

CONCLUSION

6.1 Conclusion

A new method for decoupling distribution networks for parallel processing cores is presented. The new approach, the Compensated Distributed Line Decoupling (CDLD) method, is an extension of the of Stubline idea, with significant improvements. The first improvement is the use of modal analysis for determination of added capacitance. The second is compensating the added capacitance with external inductance for steady state accuracy. The third is mitigating transient effects due to added capacitance with optimized damping.

When tested on three IEEE benchmark systems and compared to the SSN method, CDLD offered significant improvements in computational performance without serious degradation of accuracy.

A combined SSN-CDLD approach offered the best improvements overall. It was possible, using the combined approach to (1) employ less decoupling lines, (2) work with less cores, and (3) reduce mean computation time significantly compared to SSN alone.

REFERENCES

[1] C. Dufour, V. Jalili-Marandi, and J. Bélanger, "Real-Time Simulation Using Transient Stability, ElectroMagnetic Transient and FPGA-Based High-Resolution Solvers," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, 10-16 Nov. 2012 2012, pp. 283-288, doi: 10.1109/SC.Companion.2012.46.

[2] C. Dufour and G. Sapienza, "Testing 750 node distribution grids and devices," in *2015 International Symposium on Smart Electric Distribution Systems and Technologies (EDST)*, 8-11 Sept. 2015 2015, pp. 572-578, doi: 10.1109/SEDST.2015.7315273.

[3] M. D. O. Faruque *et al.*, "Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis," *IEEE Power and Energy Technology Systems Journal*, vol. 2, no. 2, pp. 63-73, 2015, doi: 10.1109/JPETS.2015.2427370.

[4] A. Teninge, Y. Besanger, F. Colas, H. Fakham, and X. Guillaud, "Real-time simulation of a medium scale distribution network: Decoupling method for multi-CPU computation," in *2012 Complexity in Engineering (COMPENG). Proceedings*, 11-13 June 2012 2012, pp. 1-6, doi: 10.1109/CompEng.2012.6242944.

[5] W. H. Kersting, "Radial distribution test feeders," in *2001 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.01CH37194)*, 28 Jan.-1 Feb. 2001 2001, vol. 2, pp. 908-912 vol.2, doi: 10.1109/PESW.2001.916993.

[6] H. Hooshyar, L. Vanfretti, and C. Dufour, "Delay-free parallelization for real-time simulation of a large active distribution grid model," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 23-26 Oct. 2016 2016, pp. 6278-6284, doi: 10.1109/IECON.2016.7793885.

[7] "Artemis User guide, Version 7.3," ed. Montreal, QC, Canada: Opal-RT Technologies, 2017.

[8] C. Dufour, J. Mahseredjian, and J. Bélanger, "A Combined State-Space Nodal Method for the Simulation of Power System Transients," *IEEE Transactions on Power Delivery*, vol. 26, no. 2, pp. 928-935, 2011, doi: 10.1109/TPWRD.2010.2090364.

[9] N. Watson, J. Arrillaga, and J. Arrillaga, *Power systems electromagnetic transients simulation*. Iet, 2003.

[10] C. D. H. S. J. Mahseredjian and J. Bélanger, "Custom-Coded Models in the State Space Nodal Solver of ARTEMiS."

[11] J. Mahseredjian, C. Dufour, U. Karaagac, and J. Bélanger, "Simulation of power system transients using state-space grouping through nodal analysis," in *Proc. Int. Conf. Power Systems Transients, Netherlands,* 2011, pp. 14-17.

[12] J. Mahseredjian, "Simulation des transitoires électromagnétiques dans les réseaux électriques," *Édition Les Techniques de l'Ingénieur,* 2008.

[13] "RT-LAB User Guide, Version 2019.2," ed. Montreal, QC, Canada: Opal-RT Technologies, 2019.

[14] *RT-LAB*. (2019). Opal-RT Technologies.

[15] C. Edith, "Circuit analysis of AC power systems," *VOL I y,* vol. 2, 1950.

[16] J. A. B. Faria and J. H. Briceno, "On the modal analysis of asymmetrical three-phase transmission lines using standard transformation matrices," *IEEE Transactions on Power Delivery,* vol. 12, no. 4, pp. 1760-1765, 1997, doi: 10.1109/61.634202.

[17] H. W. Dommel, "Digital Computer Solution of Electromagnetic Transients in Single-and Multiphase Networks," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-88, no. 4, pp. 388-399, 1969, doi: 10.1109/TPAS.1969.292459.

[18] R. A. Beezer, *A first course in linear algebra*. Beezer, 2008.

VITA

Babikir Mohamed Ahmed was born in Jeddah, Saudi Arabia in 1994 to the parents of Elnouman and Hala. In 2016, he earned his Bachelor of Science (Honors) in Electrical and Electronic Engineering with First Class at the University of Khartoum, Sudan. After graduation, he worked as an electrical engineer at the Sudanese Electricity Distribution Company (SEDC) until he was awarded a graduate assistantship, in August 2018, from the University of Tennessee at Chattanooga to pursue his master's degree in Electrical Engineering. Currently, Babikir works for Mesa Associates Inc. in Chattanooga, Tennessee, and he is to graduate on May 2020.