Guidelines for Developing Low Frequency Model Equivalents in Emerging Electrical Grids

Al-Eryani, Sameh Mohammed Anas

UNIVERSITY OF CALGARY

Guidelines for Developing Low Frequency Model Equivalents in Emerging Electrical Grids

by

Sameh Mohammed Anas Al-Eryani

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

DECEMBER, 2018

Abstract

Dynamic model equivalents of power systems is an ongoing topic that continues to be relevant despite computing power advancements. Today's electrical grids are going through significant changes. The integration and installation of technologies that use power electronic devices pose a continuous need for stability studies in many fronts. Literature focuses on dynamic model equivalent techniques, their development and validation. However, the literature lacks comprehensive guidelines to produce equivalent models consistently. This thesis presents a general procedure and guidelines to develop low frequency dynamic model equivalents. The proposed procedure and guidelines are aimed at developing consistent and reliable low frequency model equivalents. The guidelines will be demonstrated on a test system to validate the recommendations and show the impact of not following a consistent methodology in developing the equivalent. Finally, the procedure and guidelines will be applied on Alberta Interconnected Electric System to demonstrate the application on a real system model.

Preface

This thesis is an original work by the author. No part of this thesis has been previously published.

Acknowledgment

First, I would like to express my sincerest thanks to my supervisor Dr. Hamidreza Zareipour and my co-supervisor Dr. Ashraf Ul Haque for their support, their patience and guidance. Second, I would like to thank my wife, my children and my parents for their patience, support and encouragement through my education journey.

To my parents

To my love and life partner, Areeg

To my children, Mohammed and Noor

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
AIES	Alberta Interconnected Electric System
DSA	Dynamic Security Assessment
EMT	Electromagnetic Transients
FACT	Flexible AC Transmission
FDNE	Frequency Dependent Network Equivalent
HVDC	High Voltage Direct Current
ISO	Independent System Operator
MSE	Mean Square Error
M-TLNE	Modified Two-Layer Network Equivalent
PMU	Phasor Measurement Unit
REI	Radial, Equivalent, Independent
RTDS	Real Time Dynamic Simulation
TLNE	Two-Layer Network Equivalent
U of C	University of Calgary
WECC	Western Electricity Coordinating Council

Epigraph

The superiority of the learned man over the devout worshiper is like that of the full moon to the rest of the stars (i.e., in brightness).

- Prophet Mohamed

Chapter 1

Introduction

1.1. Research Motivation

Currently, there have been no universally accepted guidelines available for producing dynamic equivalent models for power systems to ensure consistency and reliability of reduced models. As such, individual facility owners around the world face challenges producing reduced dynamic equivalent models independently rather they need to depend on equipment manufacturers and consultants to develop the model. Developing model equivalents without a consistent set of guidelines will not expose vulnerabilities required to design robust solutions or reach reasonable study conclusions. Moreover, due to the dynamic change in today's electrical grids and rapid integration of converter technologies, more studies will have to be carried out by facility owners, and in the absence of a consistent methodologies for producing equivalents, challenges will arise. The topic of dynamic model equivalents is broad and under a lot of research. It is expected that this research will trigger further development to turn users around the world towards more consistent approach and less proprietary methodologies when developing equivalent models.

1.2. Literature Review

Previous literature on dynamic model equivalents is specific and focused mainly on developing and testing various equivalencing techniques. Work on power system equivalents in literature dates back to 1949. In [1], Ward presented his famous method to reduce static networks. Brown and Cloues documented one of the earliest attempts to develop dynamic model equivalents for stability studies in [2]. Podmore [3] used linearized simulations to develop low frequency model equivalents.

Wang et al. [4] described authors' experience with dynamic reduction of large power systems for transient and small signal stability studies. The paper was based on the application of DYNRED software to develop low frequency equivalents. The paper showed the result of developing equivalents for three large North American power systems. Namely, West Coast interconnected US-Canada system, East Cost interconnected US-Canada system and mid-continent interconnected US-Canada system. Similarly, Price et al. [5] showed results of applying the same software to develop and equivalent for New York Power Pool power system. Other papers showing experiences with low frequency model equivalents are [6] and [7].

In [8–13], various techniques for developing low frequency equivalents using phasor measurement unit (PMU) data are described. Using data from measurements to develop equivalent models is promising with the extensive application of PMUs around the world. However, it still lacks applications on real systems and most applications were theoretical. In [8], the authors present a technique to estimate angle and frequencies of coherent areas in large power systems. The equivalent groups are found using the slow coherency technique (described in [14]) and PMUs are used to develop a state estimator based on the reduced system that only reflects that inter-area modes of the original system model. The authors of [9] presented a dynamic equivalent models and showed that model identifiability can be achieved with pre, during and post fault data. The authors also showed how parameter estimation can be more accurate using post-fault steady state data. Application on Henan Power Company was presented in their second paper [10]. In [11], a least squares method algorithm is presented to calibrate parameters of reduced equivalent models as an attempt to achieve higher confidence in reduced models by using PMU data to verify reduced model parameters. Paper [12] presents how to derive aggregate models to represent interarea models. It assumes that there is previous knowledge about interarea oscillations and uses PMU on the transfer path between areas to estimate parameters of the equivalent models. The paper also covers how to deal with noisy measurements and proper placement of PMUs in complex networks. Finally, in [13], two limitations associated with classical model reduction techniques are discussed with proposed solutions. Firstly, classical model reduction methods are static and do not look at changes to equivalent models resulting from changes to the operating conditions of the system. Coherency identification using PMUs is presented to monitor changes in coherency of generator units (i.e., changes in equivalent models) with varying steady state conditions. Secondly, classical model reduction methods cannot handle generators that are found to be incoherent with any other generators and are typically left in the reduced model in full detail. Such generators are left in the model without any knowledge on the impact of these generators on the equivalent resulting in unnecessarily larger equivalent models. A sensitivity analysis is presented to look at the relationship of tie-line flows against the input of the incoherent generators to achieve higher reduction in the equivalent model by examining the impact of the individual incoherent generator on the tie-line flow and eliminating incoherent generators that have little impact on the tie-line.

For high frequency model equivalents, papers by Abdel-Rahman et al. [15] and Mater et al. [16] showed the development of multi-layer frequency-based model equivalent techniques for high frequency model equivalents. The mentioned papers developed novel methods to overcome the computational burden of traditional equivalents that use lumped parameters by splitting the equivalent to two layers leading to a more efficient equivalent model.

In [17–20], various attempts were presented to develop wide-band dynamic model equivalents. The equivalent models were developed using hybird methods which combine low frequency model equivalents with high frequency model equivalents and run both in parallel with various means of communications. This approach is effective in lowering hardware requirements to study large systems on real time dynamic simulator (RTDS). In [17], the authors present a hybrid equivalent consisting of a modified transient stability analysis (low frequency) model and a an electromagnetic transient (high frequency) model. A method based on energy balance is used to interface the two models. In the subsequent paper [18], the authors use coherency methods to reduce the low frequency model resulting in a smaller low frequency model. In [19], a technique is proposed to deal with inaccuracy of hybrid equivalent models for faults in the external system. It shows how electrical proximity of the fault in the electrical system can be used to change external equivalent models dynamically. In [20], solutions to two limitations faced with development and application of hybrid equivalents are attempted. The first limitation is the interaction protocol between the low frequency and high frequency equivalent models and how to achieve best accuracy with fastest simulation time. The authors propose a combined automatic interaction protocol that uses the best features of available protocols and avoids the limitations of accuracy and simulation time. The second limitation is that previous literature only included positive sequence equivalents and simulation of unbalanced faults required extending the study area to ensure that the unbalance does not appear at the boundary to the external system. The authors presented a three-sequence simulation algorithm that allows the low frequency equivalent of the external system to be used for unbalanced faults in the internal system.

This list is not exhaustive, but shares a common theme which is the natural development of new techniques. What the literature lacks is a comprehensive and generic procedures and guideline to develop different type of dynamic model equivalents for different types of studies. In [21], IEEE PES General Systems Subcommittee presented a report-style survey of dynamic equivalent techniques as they relate to electromagnetic transient simulations. The mentioned paper is the only published paper showing a summary of techniques. It is clear that literature still lacks a comprehensive up-to-date guidelines on dynamic equivalencing techniques of power system models.

1.3. Research Objectives

Up until today, model equivalents depended heavily on the knowledge and experience of the individual power system study engineer and less focused on consistently developing equivalents that can be used by multiple users. The goal of this research project is to create a general procedure and guidelines for developing low frequency dynamic model equivalents. Together they will act as an all-encompassing reference for the industry to produce consistent low frequency equivalent models.

1.4. Thesis Organization

The thesis will be organized as follows: Chapter 2 presents a high level overview of dynamic model equivalents, their applications and the available techniques for developing the different types of equivalents. Chapter 3 is a focused review of low frequency model equivalencing using coherency techniques. Then in Chapter 4, a general procedure and a set of guidelines will be proposed for developing the low frequency equivalents that are validated by demonstrations on a basic benchmark test system. Chapter 5 shows the application of the proposed procedure and guidelines on the Alberta Interconnected Electric System. Finally, Chapter 6 provides a concluding summary and discuses future research opportunities in the topic of dynamic model equivalents.

Chapter 2

Background

2.1. Introduction to Dynamic Model Equivalents

Dynamic model equivalents is a well-presented topic in literature. It dates to 1950s [2] when computing power was very limited and the need to simulate large electric systems was apparent. The original necessity for dynamic equivalents was the need for faster off-line simulations, and even with computing power advancements, today's applications for dynamic model equivalents are still around improving computing time and reducing hardware resource requirements. Some examples include simulating large interconnected systems for dynamic security assessments, running detailed studies with micro-second time step and in real time dynamic simulator applications.

Today's electrical grids are going through significant changes. In particular, the integration and installation of technologies that use power electronic devices pose a continuous need for stability studies in many fronts. For example, the Alberta Interconnected Electric System (AIES) had its first two high voltage direct current (HVDC) links energized in late 2015 connecting the northern part of the system with the southern part. Wind power has also been continually growing in the province, with a current installed capacity of about 1,500 MW. In addition, in November 2015 the provincial government in Alberta announced its Climate Leadership Plan [22]. The plan accelerates the transition from coal based generation to renewable electricity sources by 2030. Specifically, 5,000 MW of new renewable generation, most likely wind and solar generations, will be added to the AIES. Energy storage installations is also expected to be part of the transition to a cleaner grid. One of the key challenges shaping the transition will be to maintain the system's reliability and security. In a system with HVDC links and significant penetration of power electronics-based resources, designing and testing control systems and operational requirements proves challenging and calls for consistency in model equivalency techniques.

The next sections provide background on the different type of dynamic model equivalents, available techniques and their applications. Available techniques for developing the different types of dynamic model equivalents are shown in Fig. 2.1.

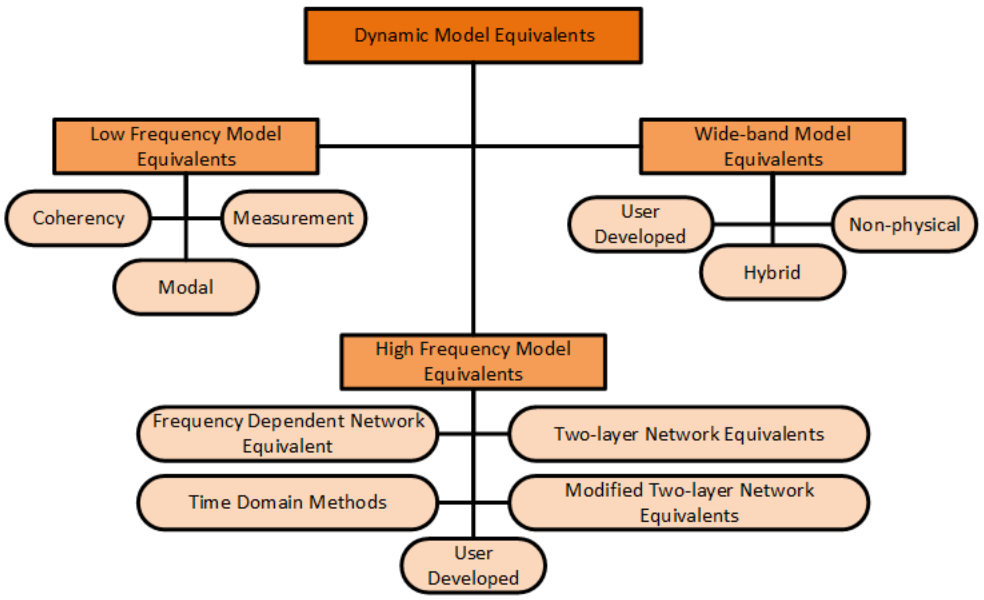

Dynamic Model Equivalents
Low Frequency Model Equivalents
Wide-band Model Equivalents
Coherency
Measurement
Modal
User Developed
Non-physical
Hybrid
High Frequency Model Equivalents
Frequency Dependent Network Equivalent
Two-layer Network Equivalents
Time Domain Methods
Modified Two-layer Network Equivalents
User Developed

Figure 2.1: Dynamic equivalent techniques

2.2. Low Frequency Model Equivalents

Low frequency dynamic models are used in simulating electromechanical dynamics such as rotor-angle stability of synchronous machines. In transient dynamic simulations, the fundamental frequency steady-state current and voltage are used to simulate electromechanical dynamics using time-varying phasors. For that reason, large time steps in the range of 4 ms to 8 ms can be used to make these models run accurately at a reasonable speed [2]. The main goal of reducing transient models is to reduce the number of generators and nodes with the goal of preserving the electromechanical low frequency oscillation modes, which are typically in the range of 0.2-2 Hz [4].

2.2.1. Techniques for Developing Low Frequency Model Equivalents

Available techniques for low frequency model equivalents include either a single or a combination of the following methods: coherency methods, modal methods and measurement methods. In this section, each of these methods are further discussed.

2.2.1.1. Coherency Methods

Literature on coherency-based methods dates back to 1970s [3, 14]. In coherency methods, equivalents are obtained by observing generator responses (i.e., swing curves) to a specific perturbation. Then, machines that are deemed coherent and responded similarly to the disturbance are aggregated to an equivalent large machine. Coherency methods have had a lot of attention in literature [3–6, 14, 21, 23] and are a part of the commercially available EPRI/Powertech software DYNRED. Depending on the method used, generally, coherency-based methods have the following advantages:

- are disturbance independent,
- produce a physical system equivalent model, and
- tested with large real systems

The assumption of being disturbance-independent is based on the fact that low frequency electromechanical oscillations between coherent groups in external systems will not change significantly when the disturbance magnitude changes. This is due to the large electrical distance between coherent groups. Disturbance duration is limited because of tight settings of protection relays and hence does not affect the identified coherent groups [3, 14]. Coherency-based methods preserve the physical representation of the system by replacing each coherent group of generators by an equivalent generator represented by a nonlinear model, which is what current simulation software use [14, 21]. The coherency based methods, specifically the methods available in DYNRED software, have been applied successfully in New York Power Pool, the West Coast, East Coast and Mid-continent interconnected US-Canada system [4, 5].

The guidelines presented in this thesis are based on coherency methods. The concept of coherency and its available techniques will be discussed in more detail in Chapter 3.

2.2.1.2. Modal Methods

Early applications of low frequency equivalents in literature used modal methods. They are based on the concept that some modes will not be sensitive to some disturbances in specific areas of the system , i.e., the study area, and can be removed from the external study area. This technique is difficult to use because it is hard to determine which modes to be eliminated as well as it is difficult to use a state matrix in available simulation programs [5]. However, modal methods can be used in coherency-based methods to identify coherent groups [3–5, 21].

2.2.1.3. Measurement Methods

Measurement-based methods for reducing low frequency models are based on the estimation of equivalent reduced model parameters based on specific measurements using PMU. These techniques are relatively new in literature without sufficient testing done on real systems. It also requires that PMU units are placed in the network strategically to capture enough dynamics of the system. These techniques can be used alone [8–10, 12] or in combination with coherency-based methods [11] to verify the reduced equivalent models. These techniques can also be used to identify the topology of a network [24].

2.2.2. Applications of Low Frequency Model Equivalents

Today's computing power is capable of running transient studies of large systems with reasonable speed. However, contrary to this fact, one application that requires low frequency equivalents is on-line dynamic security assessments (DSA) [6, 14]. DSA studies support real time operations and require to be run fast. For these studies, a low frequency equivalent is required for interconnected systems. In addition, today's interconnected power grids are complex and suffer challenges on the availability of accurate models, especially between different jurisdictions and when equipment manufacturers make their own proprietary models. For example, Alberta system model currently models interconnections to British Columbia and Montana using machines with no proper dynamic equivalent available to model these interconnections. Currently, study engineers have to use full WECC models whenever the study has influence on the inter-tie lines to these neighboring systems. With that in mind, a reduced equivalent, with acceptable performance, is required to carry stability studies. Low frequency equivalents are also relevant when studying power-electronics impact on low-frequency oscillations [21].

2.3. High Frequency Model Equivalents

High frequency models are used in simulating high frequency power system phenomena such as lightning and switching. Such studies are typically conducted in electromagnetic transient environment. In electromagnetic simulations, models use high order differential equations with time steps in the micro second range to simulate dynamic behavior, making them very accurate in a wide range of frequencies [25]. However, accuracy comes at high cost of computational power, making electromagnetic simulations limited to small networks, and otherwise requiring extensive reduction of large networks. Hence, developing high frequency model equivalents is heavily dependent on the phenomenon under study and the range of frequencies of interest. The purpose of high frequency model reduction is to reduce the number of components with higher order models that have insignificant impact on the response of the study area and maintain components that contribute to the frequencies under study.

2.3.1. Techniques for Developing High Frequency Model Equivalents

High frequency model equivalent techniques produce non-physical equivalents. All these techniques convert the external system to a fitted frequency dependent function.

2.3.1.1. User-developed High Frequency Equivalents

User-developed high frequency equivalents are produced by translating the network to a 60 Hz model that is compatible with EMT environment. The 60 Hz model is used to preserve the power flow and short circuit levels of the original network. Then, the study area is selected and modeled in detail starting from the bus of interest and upstream until the required frequency range captured in the study area covers the phenomenon being studied.

2.3.1.2. Frequency Dependent Network Equivalents (FDNE)

This method models the frequency-dependent terminal admittance of a network using either a lumped parameter circuit model or a rational function model. In this method, calculated frequency response of the terminal admittance is fitted to a function of an appropriate order. Various methods are reported in literature to fit the frequency response [7]. The most powerful is the vector fitting technique reported in [19]. In addition, [20] details a computer code for rational approximation of frequency dependent admittance matrices.

2.3.1.3. Two-layer Network Equivalents (TLNE)

Constructing an FDNE for a complex system results in a high order equivalent that uses computing resources for simulation and is not practical for real-time simulations. The two-layer network equivalent (TLNE) solves this by partitioning the FDNE into two parts [15, 21], namely: a surface layer and a deep layer. Since most high frequency responses further away from the study area are due to immediate transmission lines, the surface layer takes care of that. The deep layer takes care of lower frequency response attributed to the external area. Both layers are obtained using vector fitting methods [26]. The deep layer can also be obtained using genetic algorithms as shown in [27] to get a better approximation. This technique was successfully applied to a portion of AIES as shown in [27].

2.3.1.4. Modified Two-layer Network Equivalents (M-TLNE)

The modified two-layer network equivalent introduces a simplification to the surface layer by neglecting the frequency dependence of the transmission lines characteristic impedance and replacing it with a first-order rational function [16, 21]. This method has been tested on an implementation of a real-time simulator of a sample test system [28]. The main advantage of this technique is that it produces a reduced-order equivalent that is less computationally intensive, especially in case of multi-port external systems with long transmission lines [21].

2.3.1.5. Time Domain Methods

There are various methods that use a signal in time domain to calculate a reduced order equivalent. In these methods, either a simulated or measured signal is used to calculate an equivalent for the external system [29–33].

2.3.2. Applications of High Frequency Model Equivalents

Application of dynamic equivalents in electromagnetic studies is very common. Inclusion of low frequency models such as generators and generator controls (e.g., exciters) in electromagnetic studies are becoming more common with studies involving power electronic technologies (e.g., HVDC converters) [21]. Such studies are required when designing power electronics to mitigate low-frequency oscillations and when studying interactions of power electronics with the rest of the system. These studies need to run in electromagnetic environment because power electronic controllers have detailed models that require electromagnetic environment to accurately simulate their dynamic behavior. Also, they can lead to interactions with other low frequency modes originating by other equipment in the system typically not included in high frequency equivalents. Another application of high frequency equivalents is high frequency studies (e.g., switching). In such studies, a high frequency equivalent is required when the study area is complex and consists of a lot of transmission lines that contribute to the high frequency behavior of the study area, making the simulation very slow.

2.4. Wide-band Frequency Model Equivalents

Wide-band models are used when a single model is required to study the dynamic system's behavior in the whole frequency spectrum. Wide-band equivalents are the most challenging type of equivalents since running a model with accurate response in the whole frequency spectrum require forming an equivalent model that is either non-physical (i.e., frequency response transfer function), or in the form of a coupled low and high frequency models running with different time steps (i.e., hybrid equivalents or two-time-step equivalents).

2.4.1. Techniques for Developing Wide-band Frequency Model Equivalents

Wide-band equivalents can be developed in the form of physical, non-physical models or a combination of the two. For wide-band equivalents, specifically equivalents that run on real time simulators, the structure of the equivalent, i.e., physical or non-physical, is not relevant as in low frequency equivalents. The reason is because the limitations of simulation software do not apply.

2.4.1.1. User Developed Wide-band Equivalents

User developed equivalents are manual equivalents that are produced based on the user's knowledge and experience of the system. Typically, these equivalents would consist of a study area, a buffer zone and an external area. The combination of static and dynamic equivalents of the different areas of the system will depend on the phenomena under study. These equivalents mostly depend on the level of detail retained in the reduced model and its ability to reproduce the response of interest accurately. They also depend on the compromise between simulation time and accuracy that the study engineer is prepared to make based on the study in hand. With computation power today, this technique would suffice for a lot of studies that run on computers, but certainly will not in the field of RTDS unless the reduced system is very small (e.g., a couple of generators and an HVDC converter).

2.4.1.2. Hybrid Equivalents

Hybrid or two time-scale equivalents are currently under significant research and are showing promising results for large systems, especially for RTDS applications [17–20, 25, 34, 35]. With large power systems containing power electronics, it is very essential that simulations are carried out with detailed models in an electromagnetic environment and with a small simulation time constant. Hybrid equivalents can also be applied in a computer environment to overcome long simulation time of large systems in electromagnetic environment. The structure of a hybrid equivalent is detailed below:

- Internal System: This is the study area implemented in full detail.
- High frequency equivalent block: This is the high frequency frequency-based equivalent of the external system, developed in accordance with Section 2.3 and can be an FDNE, TLNE or M-TLNE. This block can be customized in accordance to the frequencies of interest based on the study.
- Low frequency equivalent block: This is the low frequency equivalent of the external system. This can be the full system running in the transient electromechanical environment (e.g., PSS®E) or a reduced low frequency model using the techniques shown in Section 2.2.
- Communication block: This block is essential in communicating the responses between the high frequency equivalent and the low frequency equivalent. Few available communication protocols have been investigated which include a serial [25], parallel [25] and a combined protocol [20] using the same machine or multiple machines via network sockets [20].

2.4.1.3. Non-physical Equivalents

Non-physical equivalents are developed by fitting the frequency response of the system to a frequency-dependent transfer function that produces the same dynamic behavior as the original detailed system. Methods similar to the ones detailed for high frequency model equivalents can be used to produce non-physical wide-band dynamic model equivalents.

2.4.2. Applications of Wide-band Frequency Model Equivalents

Wide-band equivalents are less needed on regular basis and are generally used in RTDS applications. Wide-band equivalents are subject to extensive ongoing research due to hardware limitations (i.e., number of racks available to simulate a large system). Wide-band equivalents are needed when studying phenomena that impacts frequencies in both the low and high frequency range such as sub-synchronous oscillations, control system design and relay testing.

2.5. Summary

This chapter presented some background on the different types of dynamic model equivalents, available techniques to develop each type and the applications associated with each type of equivalents. Low frequency models are used to study electromechanical dynamics. Techniques to develop low frequency model equivalents include: coherency methods, modal methods and measurement methods. Applications of low frequency model equivalents include: DSA applications, representing interconnected grids in dynamic simulations and studying impact of modern grid components such as power-electronics converters on low-frequency oscillations. High frequency models are used in simulating high frequency power system phenomena such as lightning and switching. Techniques to develop high frequency model equivalents include: user-developed high frequency equivalents, frequency dependent network equivalents, two-layer network equivalents, modified two-layer network equivalents and time domain methods. Applications of high frequency model equivalents include: lightning and switching studies in complex networks and when wide-band frequency equivalents require a high frequency equivalent to reduce the size of study area. Wide-band frequency model equivalent are used to study the dynamics of power systems in the whole frequency spectrum. Techniques to develop wide-band frequency equivalents include: user-developed wide-band equivalents, hybrid equivalents and non-physical equivalents. The main application for wide-band equivalents is in real time simulators.

This chapter covered the different types of model equivalents in general. The next Chapter provides a focused review of low frequency model equivalents using coherency techniques.

Chapter 3

Development of Low Frequency Model Equivalents Using Coherency Techniques

3.1. Introduction

Low frequency dynamic models equivalent techniques are categorized based on two distinct features. The first one is what information is used to develop the equivalent. The second is whether the equivalent is represented by conventional network components (e.g., machines, loads, transmission lines,etc.) or not.

The first feature distinguishes the source of model data which is used to develop the equivalent. Data sources could be either offline network model data, or alternatively, measurements from real systems such as those obtained using PMUs. Equivalents that use offline network model data use the full dynamic model data and reduce the order of it while preserving required dynamic behavior that will be of interest when conducting the studies. Coherency methods and modal methods are examples of techniques that use offline model data. On the other hand, equivalents that use real measurement as a source of data to develop the equivalent use estimation techniques to reconstruct the model based on measurements obtained during a system disturbance.

The second feature distinguishes how the equivalent model is represented after the required information is compiled together to form it. Representing the model by non-linear physical network components models results in a conventional reduced model that can be used directly in traditional power system simulation tools. On the other hand, producing an equivalent that is non-physical typically takes the form of a frequency equivalent transfer function and would require custom simulation tools to use it for simulation studies.

A technique that meets one feature does not necessarily have to meet a specific other feature.

That is, a technique that uses offline model data, typically in the form of dynamic models of physical network components, to construct the equivalent do not necessarily lead to a physical equivalent model. Table 3.1 include available techniques in literature and the category they fall into based on their features.

Table 3.1: Categories of low frequency model equivalent techniques

Technique	**Source**	**Equivalent Model**
Coherency	Offline + Online	Physical
Modal	Offline	Non-physical
Estimation	Offline + Online	Physical + Non-physical

3.2. Coherency Techniques

As mentioned in Chapter 2, coherency methods groups machines based on their responses to a certain disturbance. That is, generators are deemed coherent when they respond similarity to the disturbance and are consequently grouped together. As described in the previous section, coherency methods use offline machine and network models to derive a physical model that can be used directly in simulation tools commercially available. Moreover, coherency methods are most widely used in developing low frequency model currently and are available in commercial dynamic model equivalencing software such as DYNRED[36].

Coherency methods derive equivalents by following three steps, namely:

1. Identification of equivalent groups
2. Aggregation of equivalent groups
3. Static network reduction

Each step and available techniques will be discussed in the following subsections.

3.2.1. Stage One: Identification of Equivalent Groups

The purpose of this stage is to find which external machines swing together for a disturbance and group them together. Several techniques are available for identification of equivalent groups, to name a few [5]:

- Two-time-scale methods (slow coherency)
- Weak links
- Non-linear time simulation
- Linear time simulation
- User supplied grouping

In two-time-scale techniques (slow coherency) [5, 14], the system model is linearized and transformed using singular perturbation theory to separate the fast and slow modes. Slow modes are associated with inter-area oscillations which are separated and used to group machines based on their response to each mode. Fast modes are associated intra-area oscillations and are disturbance-dependent. As mentioned in Chapter 2, the main benefit of separating slow and fast modes is that the equivalent groups developed become disturbance-independent. This feature makes the two-time-scale technique superior in developing generic equivalents that can be used for multiple studies. Due to time separation, identification of groups can be done using eigenvalue and eigenvector analysis of the linearized system. Based on J. Chow's work [14], this can be done by linearizing the system model using the second order swing equation and then calculating the modes of the linearized system. Then, the machines are grouped into a predetermined number of groups based on their mode shapes for each slow mode. Tolerance-based slow coherency is based on the slow coherency techniques except that instead of using a predetermined number of groups as an input into the identification of groups stage. The input would be the number of slowest modes to preserve and a tolerance. The tolerance would add a measure of how perfectly coherent a certain group of machines are to the slow modes selected. This helps ensuring that in large systems, electrically far machines are not grouped together. More details about the slow coherency and tolerance-based slow coherency can be found in [14].

In Weak links method [23], coherency of generators is determined based on calculating a coupling factor between generators in the state matrix. Chow's slow coherency method discussed grouping machines based on eigenvalue and eigenvector analysis, then showed how the slow coherent groups are weakly coupled. Weak links method uses the fact that slow coherent groups will be weakly coupled and calculates the coupling factor between generators to group machines into slow coherent groups. Weak links method has the same advantage as slow coherency in that it yields equivalent groups that are disturbance independent making the equivalent useful for multiple studies. Weak Links method also skips the computational burden of calculating the eigenvalues of the linearized system required for the slow coherency method.

Another technique to determine coherency is by dynamic time simulation. In time simulation methods, a disturbance is introduced in the study area and machine rotor angles in the external system are analyzed to find generators that swing together [3]. The analysis consists of identifying generators with rotor angle responses that swing together within a predefined tolerance. The main advantage of this method is that less computations are required to identify the equivalent groups. In addition, as shown in [6], coherency can be calculated using non-linear time simulation. The main disadvantage of time simulation methods they show dependence on the type of disturbance used to group the generators and using the same equivalent to study different disturbance may yield inaccurate results [5].

3.2.2. Stage Two: Aggregation of Equivalent Groups

In this step, equivalent generators of each identified coherent group are formed and connected to represent each group. Generator aggregation methods are split into two major categories [4, 5, 14, 21]: classical and detailed. In classical methods, only the generator units are aggregated and all unit control systems (i.e., exciter, governors and power system stabilizers) are ignored. In detailed methods, all generator control systems are aggregated. Generally speaking, generator control systems provide damping to maintain stability of generators and as such, they are only required when the study duration is long or when the generator is in close proximity to the disturbance. Low frequency dynamic model equivalent are typically developed for systems far enough from the study system such as interconnections to external grids. As such, control systems will unlikely be required to be represented. Classical methods will produce acceptable aggregates for majority of cases [4]. Furthermore, detailed aggregation techniques are extremely limited and will always result in a larger equivalent model because different type of control systems within the same equivalent group cannot be aggregated into one type. Presence of different type of control systems within the same coherent group will require partitioning the equivalent group further. This is required to keep the machines that have the same type of control systems together resulting in a larger equivalent model.

3.2.2.1. Classical Aggregation Techniques

Classical methods include terminal bus aggregation, inertial aggregation and impedance compensated aggregation [5, 14, 37]. In terminal bus aggregation, the generator terminal buses are connected to a common bus with infinite admittances. The voltage at the common bus can be set to the average of the voltages at the generator terminal buses or a weighted average with respect to the active and reactive power generation. To maintain the same power flow, transformers with no leakage reactance and complex turns ratios (i.e., ideal phase shifters) connect the generator terminal buses to the newly created common bus. The phase shifters ensure that power flow from each aggregate machine matches the original power flow from the individual machines. Then, both machines are aggregated into a single machine by aggregating all machine parameters. In inertial aggregation, the aggregation is conducted similar to the terminal bus aggregation method mentioned before except that the equivalent generators are connected at their internal nodes instead of the terminal bus. This results in a more accurate aggregate model [5, 14, 37]. Impedance compensated aggregation is inertial aggregation with impedance correction to compensate for the assumption that internal generator nodes are connected via infinite admittance. It is also known as slow coherency aggregation. This aggregation technique requires linearizing the system and separating slow and fast modes by singular perturbation theory similar to slow coherency identification technique discussed in the previous section. Details on all three methods discussed are in [5, 14, 37].

All aggregation techniques discussed above deal with modeling how to connect the equivalent to the network. What is common between all techniques is how the individual parameters of the aggregate machine are developed. The impedance parameters of the equivalent machine are derived by considering all machines within the same group in parallel and adding their impedances in parallel. The inertia of the equivalent machine is the summation of the individual machines inertia. Time constants are inertia-weighted-summations of individual time constants. Table 3.2 includes the detailed equations for developing the aggregate machine parameters.

Table 3.2: Equivalent machine parameters

Parameter	**Formula**
Impedance (Z)	$Z_{eq} = \left(\sum_{i=1}^{n} \frac{1}{Z_i}\right)^{-1}$
Inertia (H)	$H_{eq} = \sum_{i=1}^{n} H_i$
Time Constant (τ)	$\tau_{eq} = \sum_{i=1}^{n} \tau_i \frac{H_i}{H_{eq}}$

In the above table, impedance would refer to all machine reactances and resistances. Similarly, time constant formula applies to all time constant parameters. All parameters should be scaled appropriately based on the individual machines base MVA to a common MVA base used for the aggregate machine.

3.2.2.2. Detailed Aggregation Techniques

As mentioned previously, detailed aggregation includes classical aggregation and aggregating all control systems. In [38], aggregated models of governors, exciters and power system stabilizers are computed by the least square fitting of the aggregated transfer function. Other methods available are MVA-weighted method [14] and the trajectory sensitivity method [14, 39]. Another detailed aggregation technique is Structure Preserving Technique shown in [40]. The proposed aggregation method uses a non-iterative procedure to determine the parameters of the equivalent generating unit, including associated control devices. It is based on the preservation of the coefficient matrices of the generator, excitation, and turbine governor models represented in time domain. Detailed aggregation is limited by the type of control systems within a coherent group and may require splitting a coherent group into smaller groups in order to aggregate similar types of control systems. Furthermore, the parameters derived through detailed aggregation are disturbance-dependent. As such, the parameters will be required to be recomputed for each disturbance that will be studied.

3.2.3. Stage Three: Static Network Reduction

In this step, the rest of static network is reduced. This includes transmission lines, transformers, loads and shunt devices. Static network reduction is conducted by reducing the size of the network admittance matrix in the external area while preserving the boundary buses of the study area and all connections to the external area. Available techniques include the Ward [1] and REI (Radial, Equivalent, Independent) [41] methods. The Ward method uses Gaussian elimination to recalculate the admittance matrix of the network such that injection currents at the boundary buses remain the same. In REI, power injections from external network are aggregated to a radial equivalent independent nodes through a lossless network. Then all non-essential buses are reduced using Gaussian elimination. The main difference between the two methods is that the Ward method replaces all injections with artificial injections by processing the admittance matrix of the external system. That results in losing the physical relationship of the external system such as load and generation. Consequently, the reduced static network will need to be recalculated for each system load and generation condition. By contrast, the REI method can aggregate the external system into physical regions preserving the physical load and generation of each region and allowing system conditions to be changed without changing the equivalent network. One of the available tools to perform static network reduction is the EEQV tool in PSS®E, which is based on the Ward technique discussed.

3.3. Tools to Develop Low Frequency Model Equivalents

This section documents the different tools used in developing low frequency model equivalents in this thesis. The purpose of this section is to document the tools for future researchers to be able to expand on this work further as the topic of dynamic model equivalents is ever expanding with dynamically changing electrical grids around the world.

The tools used in this thesis are split into three main parts:

- Coherency Toolbox- this toolbox was adapted from J. Chow's work in [14] with authorization. The original toolbox was written in MATLAB.
- Python Interface to PSS®E
- Simulation and Automation Tools- these are the tools used for dynamic simulation and plotting the results

The next few subsections will provide details on the three main parts described above. More detailed description of main modules and the source code are available in Appendix C.

3.3.1. Coherency Toolbox

This toolbox is the tolerance-based slow coherency algorithm that is part of the PST Toolbox developed by J. Chow [14] and others. This toolbox is publicly available and authorization was obtained from Professor J. Chow to use it for this thesis (see Appendix B). The toolbox also includes the three aggregation techniques discussed in this thesis, terminal aggregation, inertial aggregation and slow coherency aggregation.

The main addition introduced by the work of this thesis is the translation of the code to Python language. This is to simplify the interface to the software used for this work, PSS®E. PSS®has Python modules that can be used to run power flow studies, extract network data seamlessly, and is a software that is widely used in North America for power system studies. The details of the different functions in each module are available in the PST user manual that can be downloaded with the toolbox.

3.3.2. Python Interface to PSS®E

This part of the toolbox was developed as part of this thesis. This part interfaces with PSS®E to extract network data and device model parameters needed for the coherency toolbox. It also includes a Python wrapper for the coherency toolbox to allow running multiple equivalents with a set of input parameters such as the number of slow modes to preserve.

3.3.3. Simulation and Automation Tools

The simulation and automation tools developed as part of this thesis comprise of a dynamic simulation automation tool and an automated plotting tool to plot all results and prepare them for analysis.

3.4. Summary

This chapter presented a focused review of developing low frequency model equivalents using coherency techniques. Coherency techniques are the most common techniques used by the industry today and are available in commercially available solutions. The main tools used in developing low frequency model equivalents in this thesis were presented as well. The next chapter presents the main contribution of this thesis, a general procedure and guidelines to develop low frequency model equivalents.

Chapter 4

Guidelines to Develop Low Frequency Model Equivalents

4.1. Introduction

This chapter presents a general procedure and a set of guidelines aimed at developing low frequency model equivalents in a consistent manner. The procedure consists of four main stages that are closely tied to the three stages associated with developing low frequency model equivalents discussed in Chapter 3. The guidelines will be presented as principles sequentially. The principles will be supported by dynamic study demonstrations to show the validity of the guideline principles. The New England IEEE-39 Bus benchmark system will be used for the dynamic studies. Further demonstrations on a real system are presented in Chapter 5.

4.2. General Procedure to Develop Low Frequency Model Equivalents

Developing low frequency model equivalents should be conducted in four stages. The preliminary stage is stage one where the study cases are prepared and the preliminary requirements for the equivalent model are defined. In stage two, the equivalent groups are identified using a grouping algorithm. Stage three takes the equivalent groups and forms the aggregate models of the equivalent groups. Finally, stage four is the static network reduction stage, and in this stage, the rest of the static components are reduced. Fig. 4.1 shows a flow chart of the proposed procedure and more details of each step are included below.

4.2.1. Stage One: Prepare and Define

In this stage, the study case is prepared and the general dynamic model equivalent requirements are defined.

4.2.1.1. Step: 1.1- Prepare Study Case

In this step, the study case is prepared based on the system scenarios to be studied such as load and generation dispatches. Then, the required steady state and dynamic models are included and checked for accuracy.

4.2.1.2. Step: 1.2- Define General Equivalent Model Requirements

In this stage, high level requirements are defined to direct and benchmark the equivalent model against. This would include: target reduction percentages for machines, buses and lines, steady state and dynamic accuracy requirements and the type and location of disturbances to be studied.

4.2.2. Stage Two: Identification of Equivalent Groups

As shown in Chapter 3, several techniques are available for identification of equivalent groups using coherency. The goal of this stage is to find which machines swing together for a disturbance and group them to an equivalent machine.

Techniques that use frequency in selecting the equivalent are more superior than techniques that use the number of equivalent groups (i.e., the size of the equivalent) as an input. This is because using frequency will ensure that slow modes are selected and preserved in the equivalent system more accurately. In addition, techniques that show less dependence on disturbance and work for variety of disturbances are superior because they do not require continuous benchmarking against the full model. One of the techniques that satisfy both previously mentioned features is the tolerance-based slow coherency identification technique. This method, or a similar method that meet the same requirements, is recommended to be used for the identification of equivalent group. The main input parameters to the tolerance-based slow coherency identification algorithm are the number of slowest modes N and tolerance t.

4.2.2.1. Step 2.1- Define Frequencies of Interest

In this step, the frequencies of interest are defined based on the study scope and the knowledge and experience on the system to be studied. This is important if specific frequencies are of interest and need to be retained. When no specific frequencies are of interest, the equivalent model should preserve as many slow frequencies as possible without exceeding the requirement of the size of equivalent.

4.2.2.2. Step 2.2- Define Study Area and External Area

At this step, the study area needs to be defined and a demarcation is made for which buses would be considered the boundary of the study area.

4.2.2.3. Step 2.3- Prepare Models for Identification of Equivalent Groups

In this step, the study case is prepared for identification of equivalent groups. This would include removing specific generators that do not have dynamic models and replacing dynamic devices such as HVDC converter, renewable generators and loads with appropriate models required for the identification stage.

4.2.2.4. Step 2.4- Perform Identification of Equivalent Groups

This is where the main work happens in stage two. The equivalent groups are identified using one of the available techniques. The output of this step is the output of stage two which is a list of equivalent groups defined by specific generators.

4.2.3. Stage Three: Aggregation of Equivalent Groups

In this stage, equivalent generators of each identified coherent group are formed and connected to represent each group. As discussed in Chapter 3, generator aggregation methods are split into two major categories, classical methods that ignore control systems and detailed methods that include control systems of the equivalent generator (i.e., governors, exciters, and power system stabilizers).

4.2.3.1. Step 3.1- Define Equivalent Groups Based on Study Area and Geographical Requirements

In this step, the equivalent groups found in stage two are further split as some machines will have to be preserved in details inside the study area and others will have to also be preserved to maintain the geographical features of the system. Also, groups are split for aggregation requirements such as technology (i.e., hydro, thermal) and types of control systems. Finally, the specific methodology to aggregate the equivalent groups is selected at this step.

4.2.3.2. Step 3.2- Perform Aggregation of Equivalent Groups

The selected methodology and equivalent groups are aggregated and the parameters of the equivalent generators are computed and added to the study case.

4.2.3.3. Step 3.3- Test Dynamic and Steady State Accuracy Against Technical Performance Requirements

In this step, the performance of the equivalent model is tested against specified accuracy requirements defined in stage two. If the performance is not satisfactory, the equivalent groups are modified by either defining a bigger study area, or more appropriately, by recomputing the equivalent groups with more frequencies included in the identification stage. Also, at this stage, the various disturbances that this model will be used to study are tested to confirm accuracy requirement and whether the equivalent needs to be developed for each type or classification of disturbances. If the technical performance requirements are met, then the development of the equivalent model can proceed to static network reduction. The main reason for doing technical performance check at this stage is to separate performance issues caused by aggregation of equivalent groups from the ones cased by static network reduction.

4.2.4. Stage Four: Static Network Reduction

In this stage, the rest of static network is reduced to eliminate unessential buses and branches outside the study area. Static network reduction is required to reduce simulation time and complexity of the admittance matrix of the network. Depending on the scope of the study, various degrees of network reduction would be required further away from the study area.

4.2.4.1. Step 4.1-Define Static Network Reduction Requirements

In this stage, the requirements for static network reduction are defined. This would include the amount of reduction and which components are required to be kept in detail. Some of these requirements may dictate using a specific static network reduction technique over the other to facilitate using the same model for multiple studies. More details about static network reduction requirements is presented in the guideline principles.

4.2.4.2. Step 4.2- Perform Static Network Reduction

The requirements defined in the previous step along with the technique selected are used to perform static network reduction.

4.2.4.3. Step 4.3- Test Dynamic and Steady State Accuracy Against Technical Performance Requirements

In this step, the equivalent model after static network reduction applied is tested against the same performance requirements defined in stage one. In addition, the requirement of static network reduction are also confirmed to ensure the model is robust and ready for the studies. If the model does not meet the performance requirements, then the requirements of static network reduction should be modified by increasing the number of buses in the static network equivalent or by relaxing the performance requirements to ensure the reduced model meets the required performance requirements.

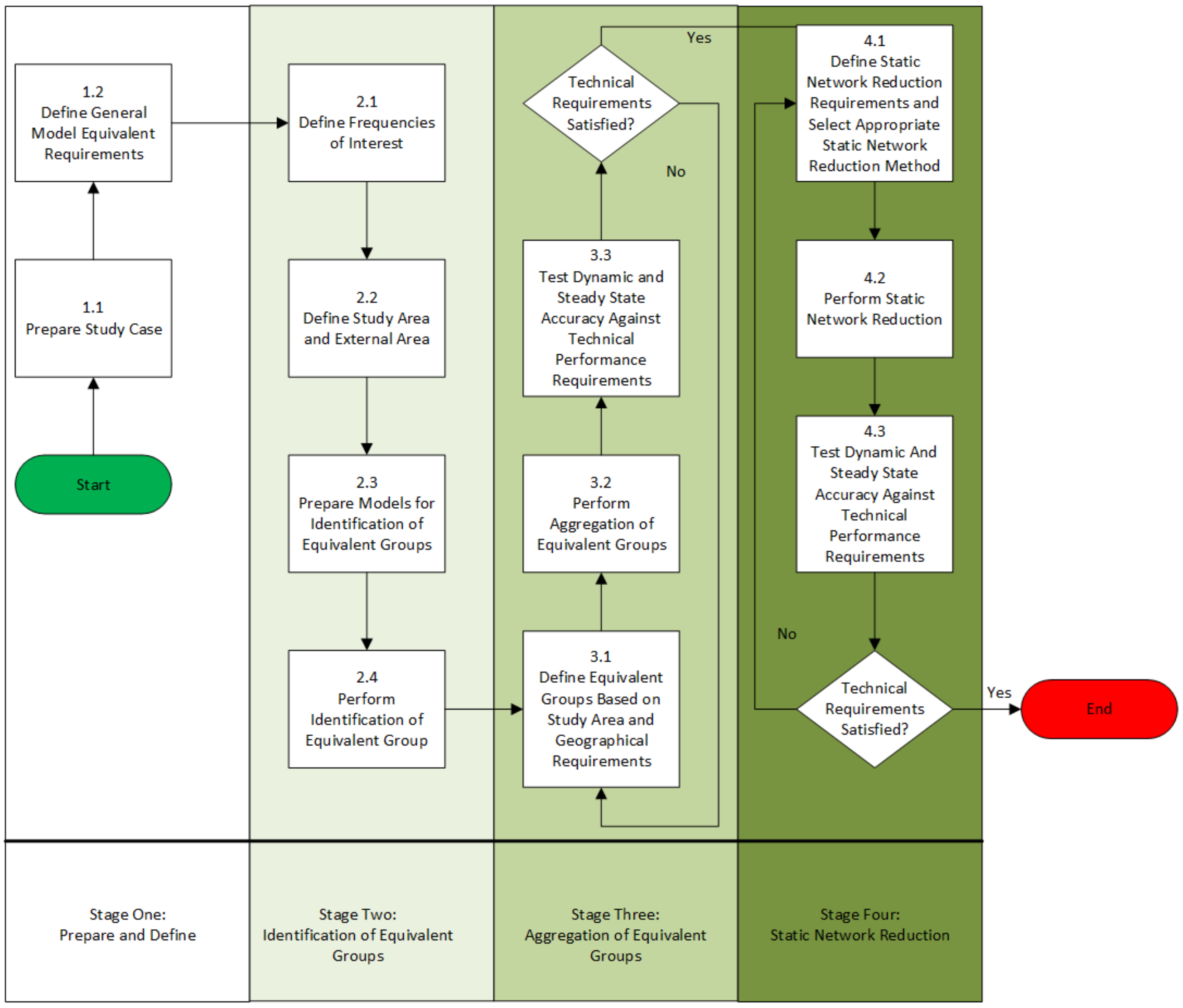

1.2
Define General Model Equivalent Requirements
1.1
Prepare Study Case
Start
2.1
Define Frequencies of Interest
2.2
Define Study Area and External Area
2.3
Prepare Models for Identification of Equivalent Groups
2.4
Perform Identification of Equivalent Group
Technical Requirements Satisfied?
Yes
No
3.3
Test Dynamic and Steady State Accuracy Against Technical Performance Requirements
3.2
Perform Aggregation of Equivalent Groups
3.1
Define Equivalent Groups Based on Study Area and Geographical Requirements
4.1
Define Static Network Reduction Requirements and Select Appropriate Static Network Reduction Method
4.2
Perform Static Network Reduction
4.3
Test Dynamic And Steady State Accuracy Against Technical Performance Requirements
No
Technical Requirements Satisfied?
Yes
End
Stage One: Prepare and Define
Stage Two: Identification of Equivalent Groups
Stage Three: Aggregation of Equivalent Groups
Stage Four: Static Network Reduction

Figure 4.1: General procedure to develop low frequency model equivalents

4.3. Description of Test System

The New England IEEE 39-Bus benchmark system was used to validate the guideline principles. It consists of the following components:

- 39 buses
- 34 branches and 12 transformers

- 10 generators with total dispatch of 6,141 MW. All generators are represented by GENROU models with IEEET1 model exciters except for machine connected to bus 39 which is an aggregate machine representing external system with no exciter.
- 19 loads totaling 6,098 MW
- no shunt devices

It was adopted from [42]. The following modifications are done in order to study the impact of power electronics on equivalents:

- The transmission line between buses 1 and 2 is replaced with a 500 kV HVDC line with a power order that matches MW flow on the original AC line. Shunt capacitor banks were added at both terminals to match original MVar flow at the two terminal buses. HVDC converters use CDC6T dynamic models
- Four type-4 wind machines totaling 80 MW are added to buses: 15, 21, 24, and 27. WT4G1 and WT4E1 dynamic models were used for these generators

The steady state and dynamic model parameters are included in Appendix A. In the studies of this chapter, all machines will be referenced by their high voltage terminal buses. The single line diagram of the IEEE 39-Bus system is shown in Fig. 4.2.

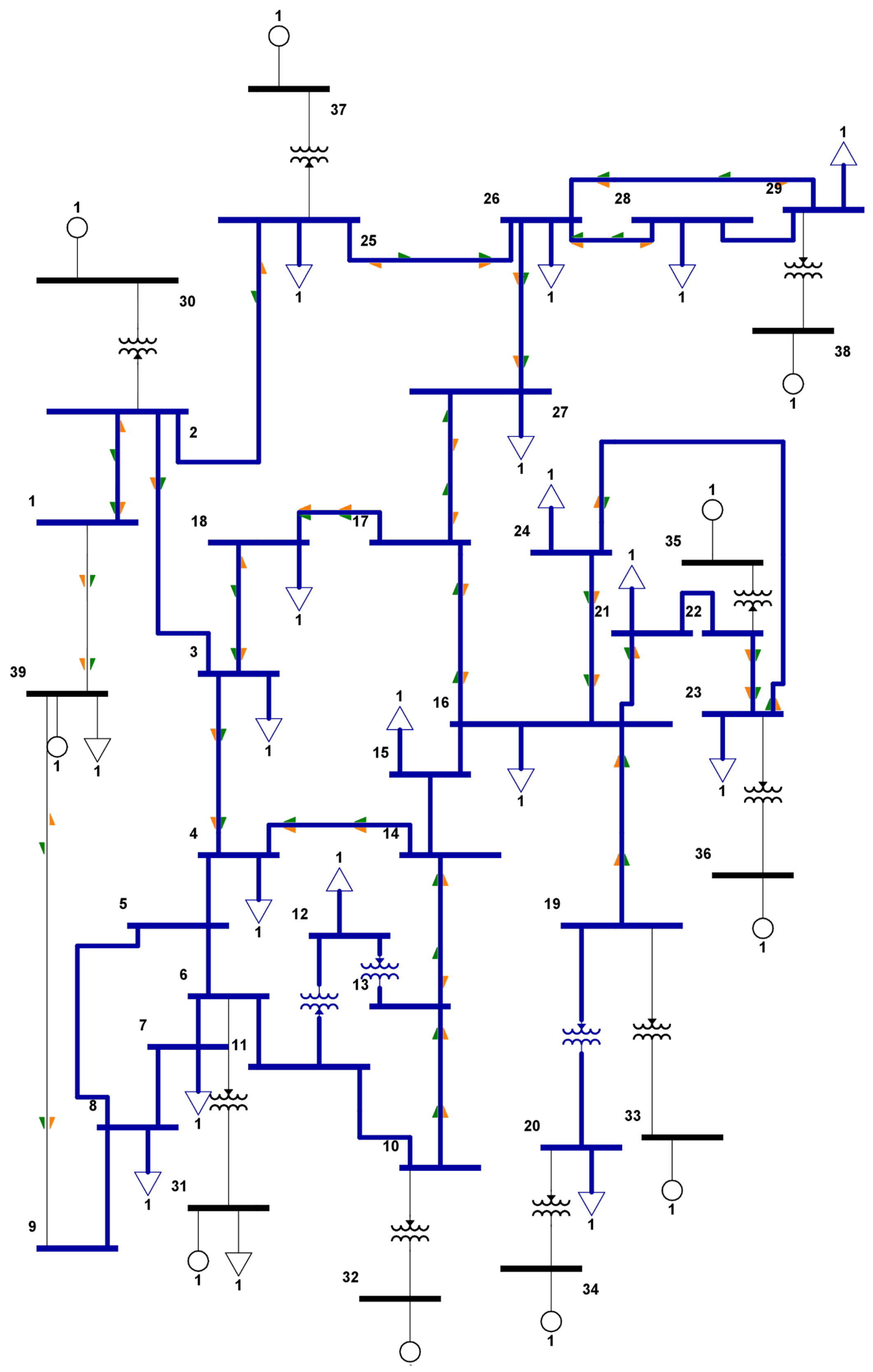


Figure 4.2: New England IEEE 39-Bus System

4.4. Guideline Principles to Develop Low Frequency Model Equivalents

The proposed guidelines to develop low frequency model equivalents consist of ten principles. The ten principles follow the general procedure defined in the previous section and are detailed in the following subsections.

- Principle one belongs to Stage One.
- Principles two to five belong to Stage Two.
- Principles six to eight belong to Stage Three.
- Principles nine and ten belong to Stage Four.

4.4.1. Principle One: Define Equivalent Model Requirements

The start of the development of a low frequency model equivalent should be a defining stage where main requirements are defined and specified. This ensures that the remaining stages of model development are based on a clear common goal. Some of the requirement that must be defined at this stage are:

- Percentage reduction for the number of machines
- Percentage reduction for the number of buses
- Mismatch tolerance for short circuit fault level and power flow from the study area
- Dynamic accuracy requirements
- Scenarios to be included in the study

4.4.2. Principle Two: Preserve Modes in the Range 0.2 Hz - 2 Hz

Low frequency oscillations are those associated with electro-mechanical interactions in power systems. When developing low frequency dynamic model equivalents, it is important to capture the appropriate oscillation modes. Low frequency oscillations associated with power systems are in the range of 0.2-2 Hz. Regardless of the technique used to identify equivalent groups, low frequency modes in the range specified should be preserved. This is the second principle of this guideline. In more specific cases when only a subset of frequency in the frequency range mentioned is of interest, a smaller range can be selected accordingly. The more modes preserved in the equivalent grouping, the higher is the accuracy of the equivalent. The consequence of preserving more modes is a larger number of equivalent groups leading to larger and slower equivalent.

Using New England IEEE 39-Bus test system and slow coherency identification technique, Table 4.1 shows various equivalent machine groups based on selecting different number of slow modes to include in the equivalent and the tolerance within each group. In these equivalents, only machine 37 is preserved in detail and forms the study area. Fig. 4.3 show the study area. The table shows the impact on accuracy by measuring mean square error (MSE) between the full model and the equivalent model for rotor angle, power swing and voltage for a sample disturbance inside the study area. The disturbance consists of a three-phase-to-ground fault at bus 2 which is cleared after 0.06 seconds. The rotor angle and power swing measurements are for machine 37. The rotor angle is plotted against machine 30 and when machine 30 is not retained, the rotor angle plot is not meaningful and MSE is large. The voltage plot is for the voltage at the faulted bus 2.

Based on MSE presented in Table 4.1, it is evident that including more slow modes in the equivalent results in more accurate model. In real systems with hundreds of generators, hundreds of slow modes will be present, and the number of modes selected to preserve in the equivalent model will impact the size of the equivalent and consequently the accuracy of the equivalent. Fig. 4.4 and 4.5 show the rotor angle, power swing and voltages for two modes (N=2,t=0.97) and four modes (N=4,t=0.92), respectively.

Table 4.1: Impact of selected number of modes on the accuracy of the equivalent

			Mean Square Error (MSE)		
# of Modes (N)	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
4	0.92	7	6.67E-01	6.20E-04	1.78E-05
3	0.97	7	1.44E+05	2.01E-02	2.39E-05
2	0.97	6	3.05E+05	2.17E-02	6.69E-05
2	0.87	5	1.01E+06	2.30E-02	1.17E-04
1	0.99	3	1.88E+05	5.70E-02	1.28E-04

Note: Not retaining the reference machine results in large MSE for rotor angles

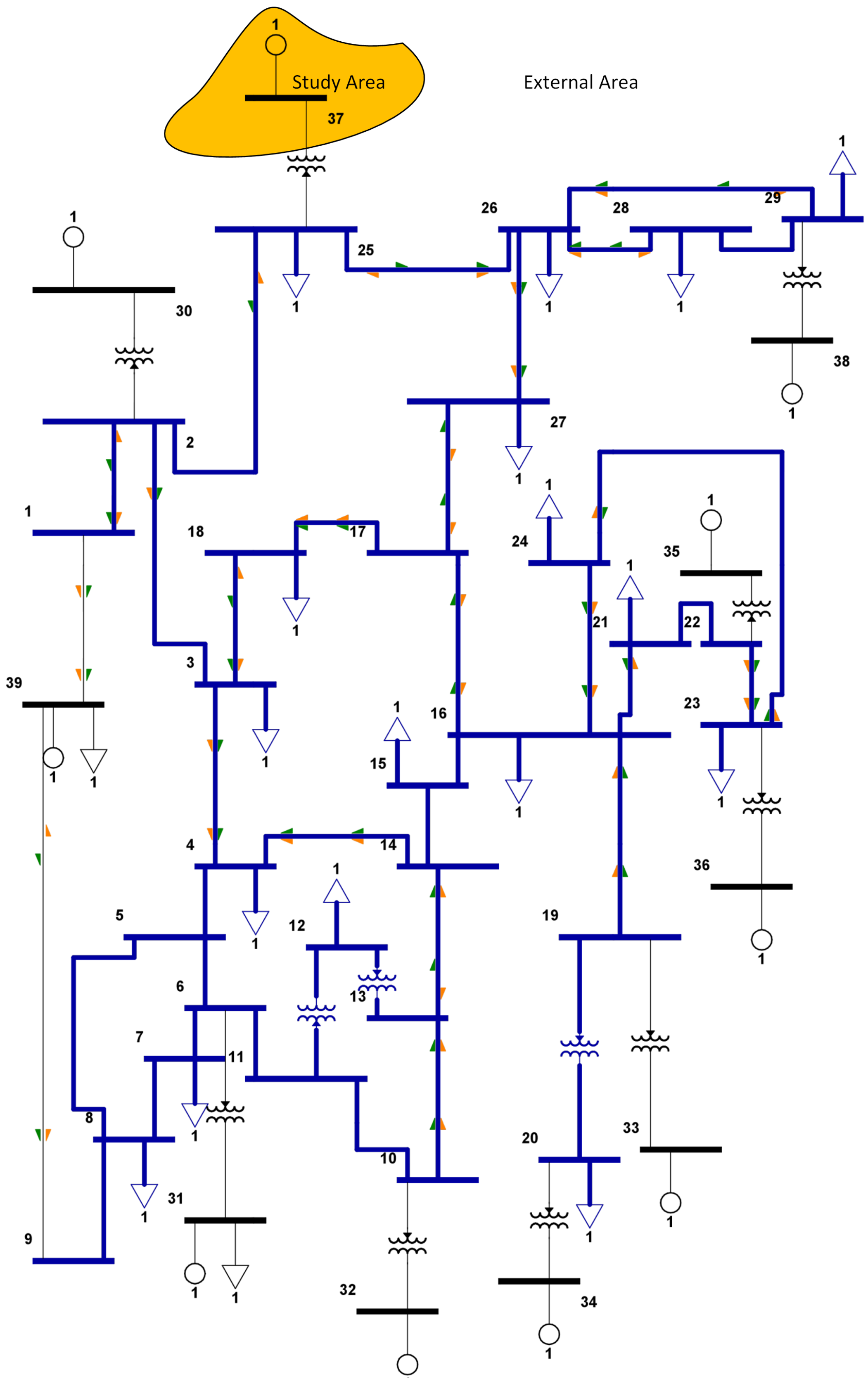


Figure 4.3: New England IEEE 39-Bus System - Small Study Area

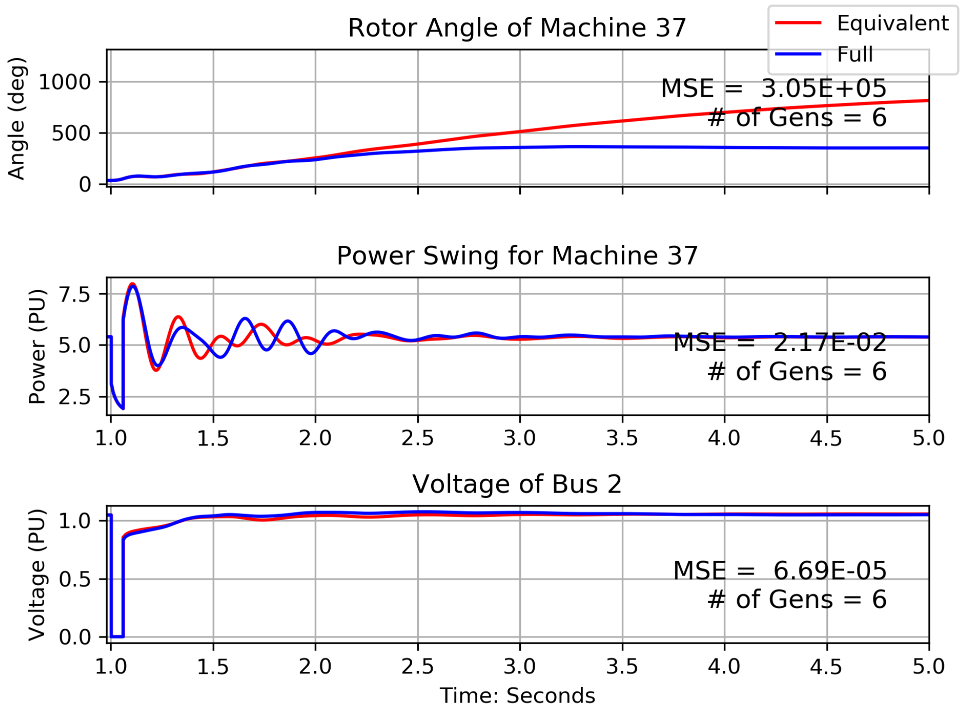


Figure 4.4: Equivalent machines = 6, preserved modes = 2 (N=2, t=0.97)

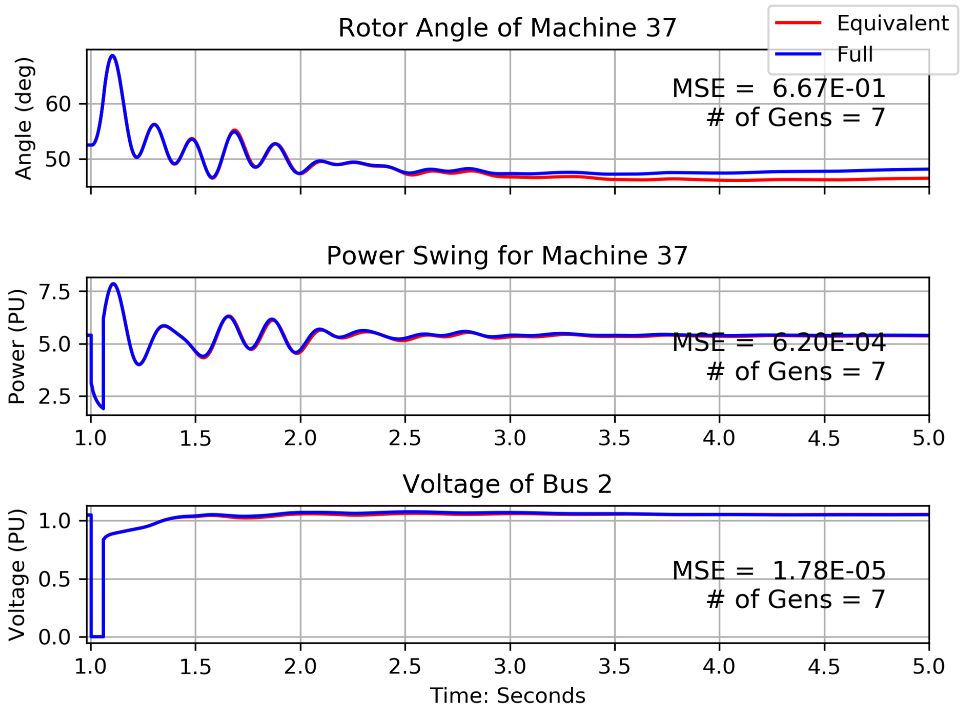


Figure 4.5: Equivalent machines = 7, preserved modes = 4 (N=4, t=0.92)

User supplied grouping can be a simple and effective method in forming equivalent groups, but it requires immense knowledge about the power system and its low frequency modes. While this is not recommended as part of this principle, if the user have enough knowledge about which machines swing in a coherent manner after disturbances, then these machines can be grouped together to form an equivalent group.

4.4.3. Principle Three: Define Study Area Boundary

The previous section showed equivalent groups based on specifying a study area that compromises of just one machine, machine 37. In practical applications, the study area needs to be defined more specifically. The main advantage of defining a bigger study area especially in large power systems is that including certain machines in the study area will result in an overall smaller equivalent. This is because machines that are electrically close to the study area have more impact on the dynamic behavior of the machines in the study area. This is the third principle of this guideline.

Using IEEE 39-Bus system, a larger study area comprising of machines 37, 30 and 39 is defined to show the impact of more appropriately defined study area. Fig. 4.6 shows a detailed view of the study area. Table 4.2 shows equivalent groups and MSE when machines 39 and 30 are also retained in addition to machine 37. The table shows that for the grouping developed from selecting just one mode (N=1 and t=0.99), the equivalent number of machines becomes five instead of the three obtained previously (see Table 4.1). This grouping results in 50% reduction in the number of machines with reasonable accuracy. Fig. 4.7 shows the difference in accuracy for machine 37 before retaining machines 30 and 39 and after retaining them and it also shows the benefit of including machines electrically close to the study area. In fact, when comparing Table 4.2 with Table 4.1, the MSE for the power swing at machine 37 with the five-machine equivalent resulting from preserving one mode is more accurate than the six-machines equivalent resulting from selecting two modes when machine 37 is only retained in the study area. This further confirms the importance of keeping enough modeling details in the study area.

Table 4.2: Impact of selecting larger study area

			Mean Square Error (MSE)		
# of Modes (N)	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
3	0.97	8	2.19E-01	2.19E-04	6.31E-06
4	0.92	7	6.67E-01	6.20E-04	1.78E-05
2	0.97	8	2.98E-01	1.10E-03	2.38E-05
1	0.99	5	1.88E+00	1.75E-03	6.48E-05
2	0.87	6	2.16E+00	2.34E-03	6.80E-05

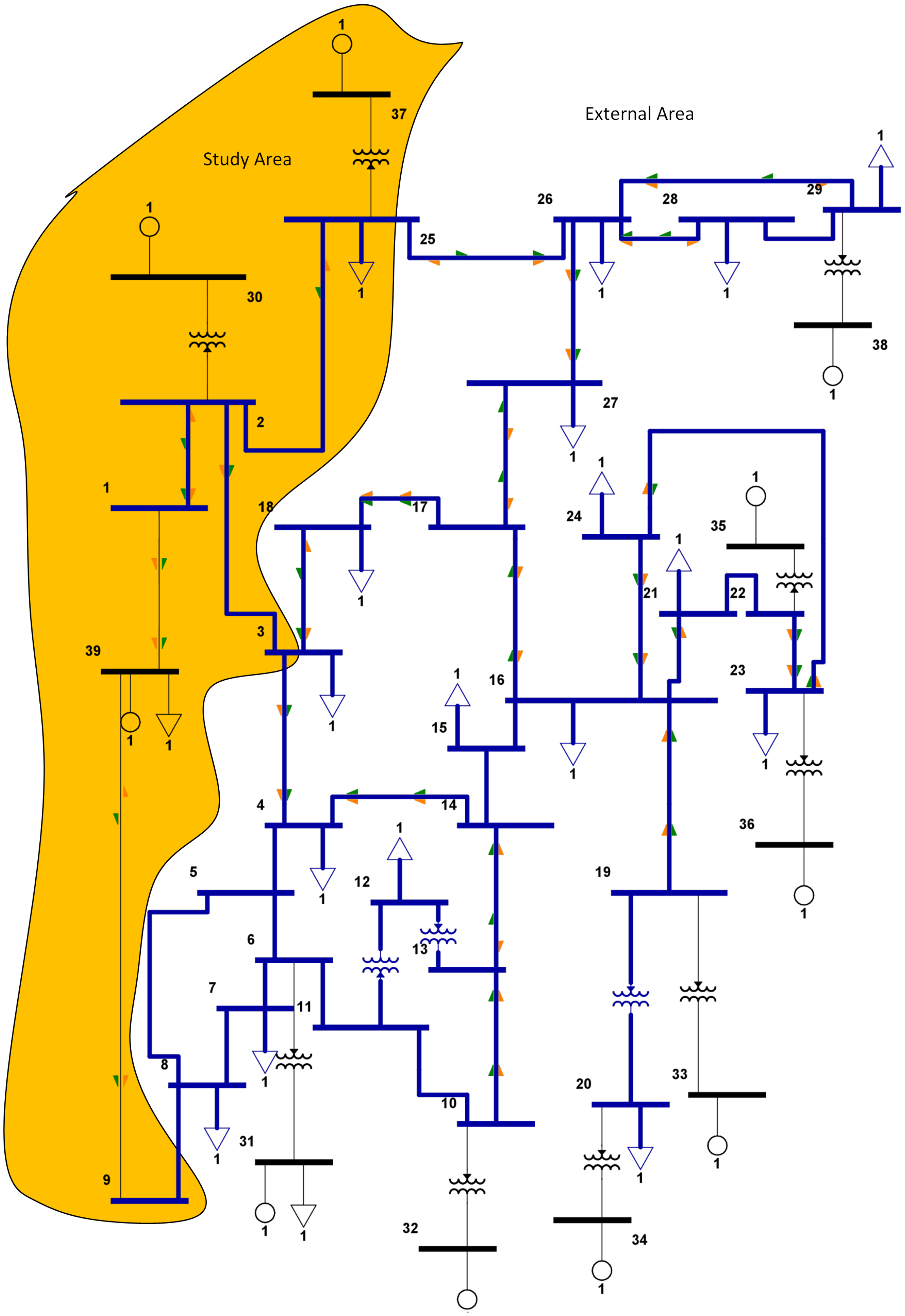
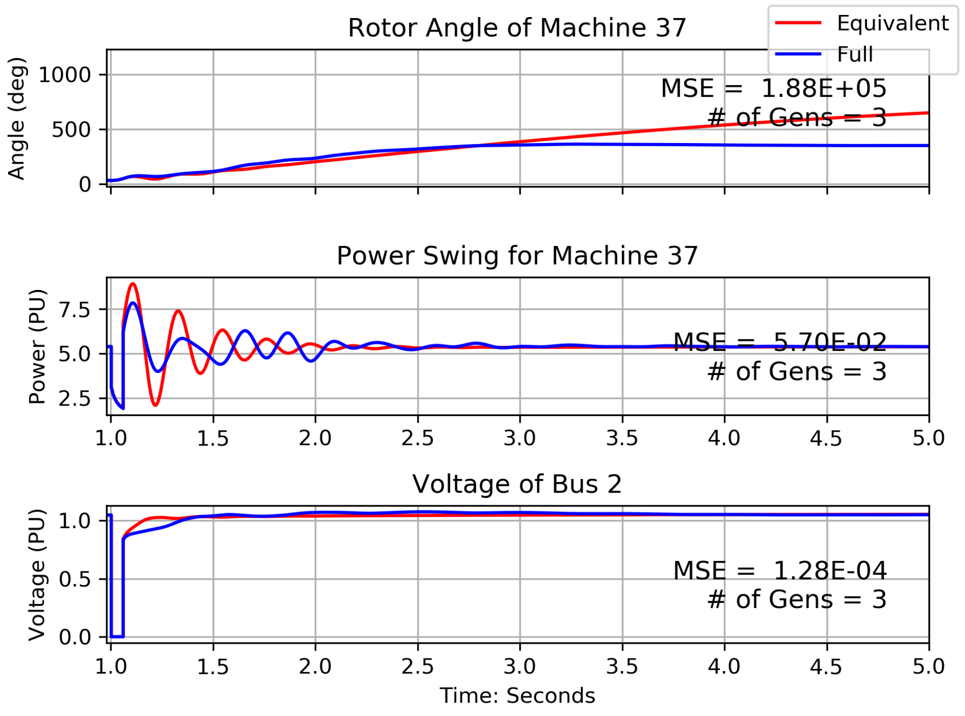
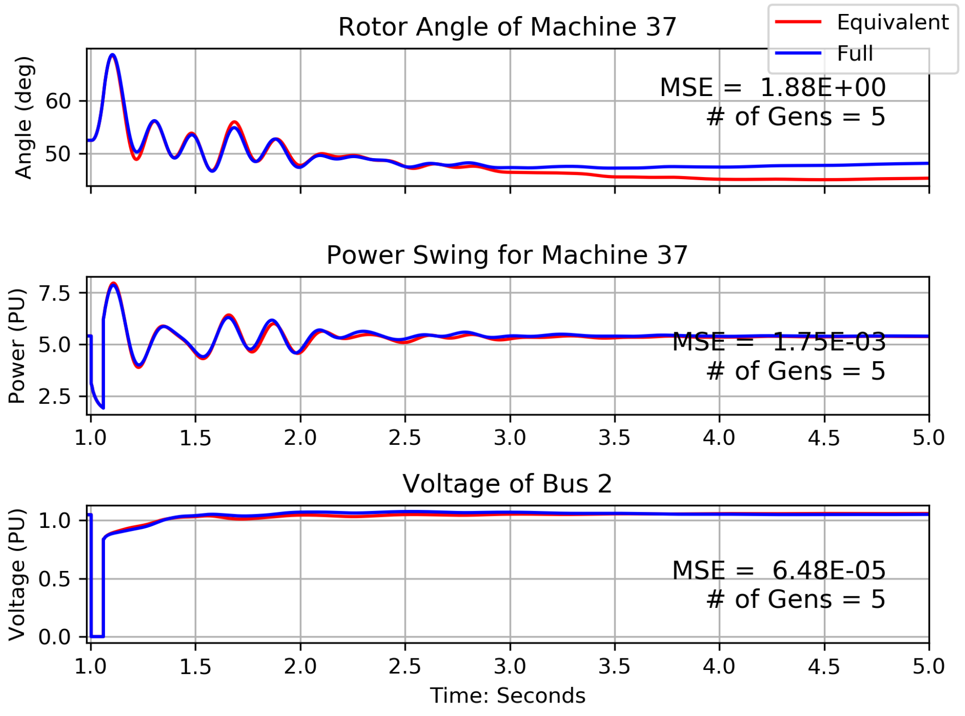


Figure 4.6: New England IEEE 39-Bus System - Large Study Area



(a) Fig a-Without retaining machines 30 and 39



(b) Fig b-With retaining machines 30 and 39

Figure 4.7: Preserved modes: 1 (N=1,t=0.99), impact of retaining machines 30 and 39

4.4.4. Principle Four: Include the Impact of Non-passive Devices on Group Identification

Today's power grids contain a lot of non-passive devices such as converters used in HVDC transmission and in a lot of renewable energy resources. As mentioned previously, when developing low frequency equivalents, electromechanical transients are required to be preserved to a level that yields accurate dynamic behavior. Low frequency oscillations are mainly associated with rotating machines and although non-passive devices may interact with these modes leading to over-damping or under-damping these modes, non-passive machines will not impact the identification of equivalent generators. However, inclusion of the non-passive devices in the dynamic simulations is required unless their behavior is either not the interest of the study or they are far enough that they will not impact the dynamics of the study area. The fourth guideline principle is to include the impact of non-passive devices when developing the equivalent by replacing them with constant loads preserving the power flow of the system and then putting them back for dynamic simulations.

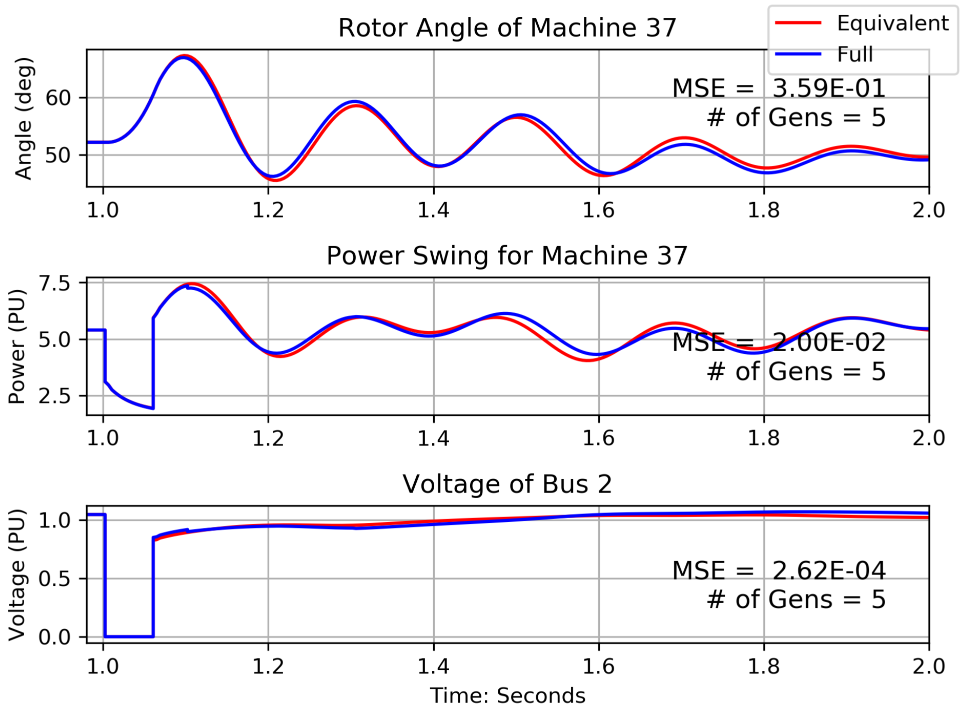
Using IEEE 39-Bus system, an HVDC link is added between buses 1 and 2 to replace an existing AC line maintaining the same power flow. Two cases were attempted to develop equivalent groups and show the impact of non-passive devices on the equivalent as follows:

- All Constant (AC): Replace HVDC by constant load and keep it as constant load for dynamic simulation
- Identification Constant (IC): Replace HVDC by constant load in the identification stage and put it back for dynamic simulation

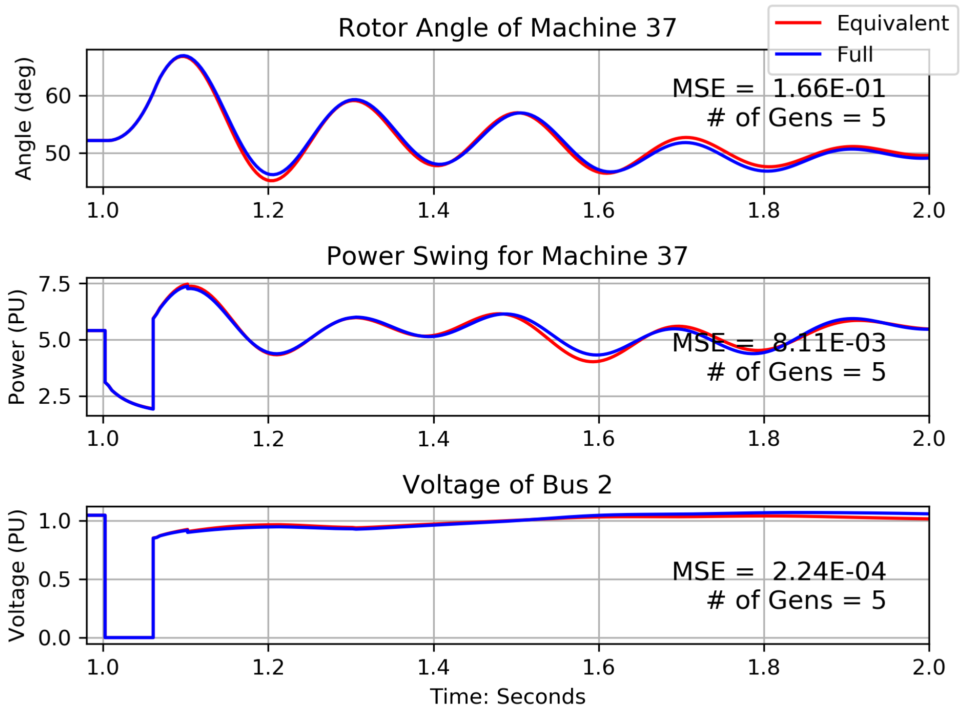
Table 4.3 shows the MSE associated with each of the two cases above when developing equivalent groups that preserve the same number of modes. It is evident that the dynamics of the HVDC converters do not need to be represented in identification but are required for dynamic simulations. So, they cannot be ignored completely. This is indicated in higher MSE for all cases denoted by All Constant. Fig. 4.8 shows both cases as an example for when one mode is preserved.

Table 4.3: Impact of HVDC

				Mean Square Error (MSE)		
Case	**# of Modes (N)**	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
IC	1	0.99	5	1.66E-01	8.11E-03	2.24E-04
AC	1	0.99	5	3.59E-01	2.00E-02	2.62E-04
IC	2	0.97	7	8.67E-02	7.16E-03	1.06E-04
AC	2	0.97	7	2.60E-01	2.14E-02	1.37E-04
IC	3	0.97	9	3.56E-03	5.92E-04	3.64E-06
AC	3	0.97	9	1.48E-01	1.12E-02	8.87E-05
IC	4	0.92	7	1.07E-02	1.54E-03	5.04E-05
AC	4	0.92	7	1.13E-01	7.65E-03	9.45E-05
IC	6	0.90	8	8.01E-03	1.13E-03	2.40E-05
AC	6	0.90	8	1.18E-01	8.20E-03	8.26E-05



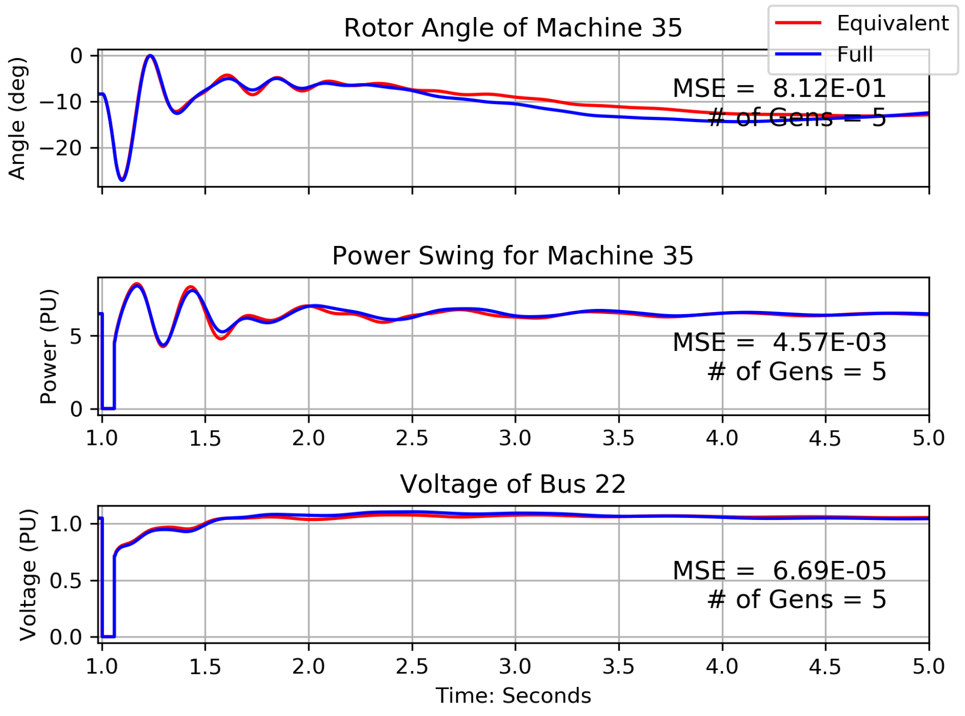
(a) Fig a-All Constant



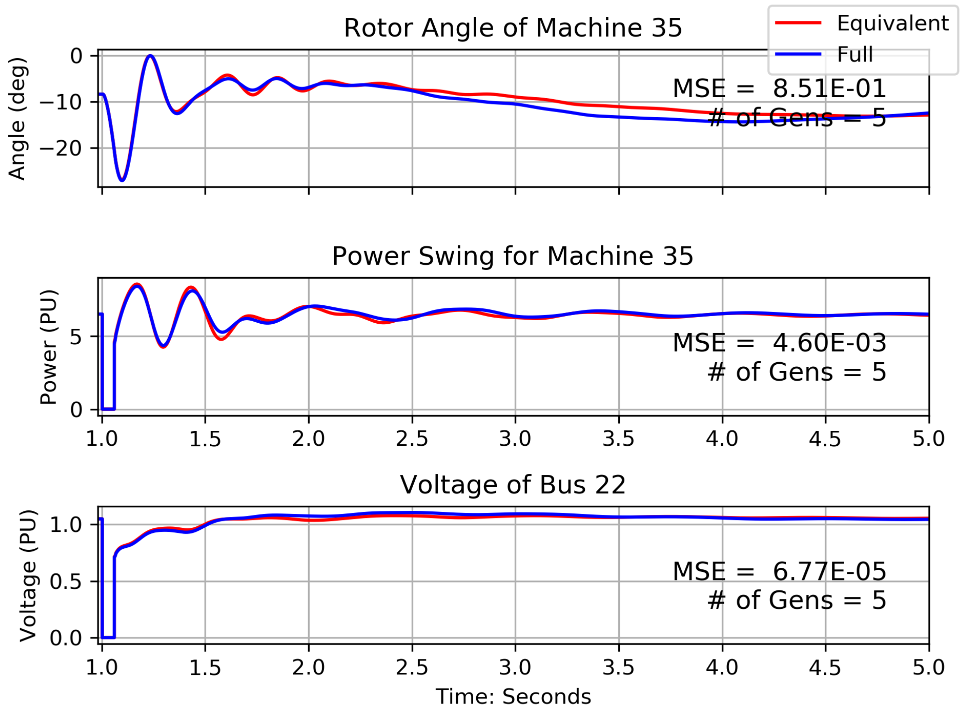
(b) Fig b-Identification Constant

Figure 4.8: Preserved modes = 1 (N=1, t=0.99) for system with HVDC close to study area. (a) is for all constant and (b) is for identification constant

When the HVDC is outside and far from the study area, the impact of using the full model during dynamic simulations becomes less important. To show that, a three-phase-to-ground fault is simulated far away from the HVDC link (at bus 22) and machine 35 is monitored. Fig. 4.9 shows both cases (i.e., constant load for identification and constant load for identification and dynamic simulation) for when four modes are preserved (N = 4 and t = 0.8) and the MSE is similar between the two.



(a) Fig a-All Constant



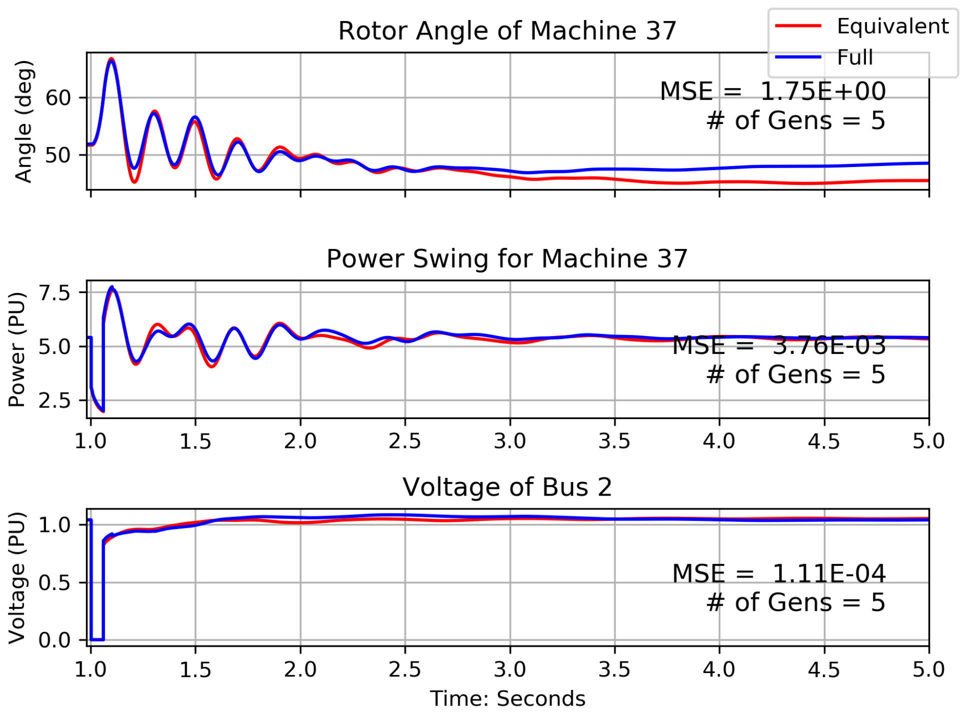
(b) Fig b-Identification Constant

Figure 4.9: Preserved modes = 4 (N=4, t=0.8) for system with HVDC far from study area. Top is for all constant and bottom is for identification constant

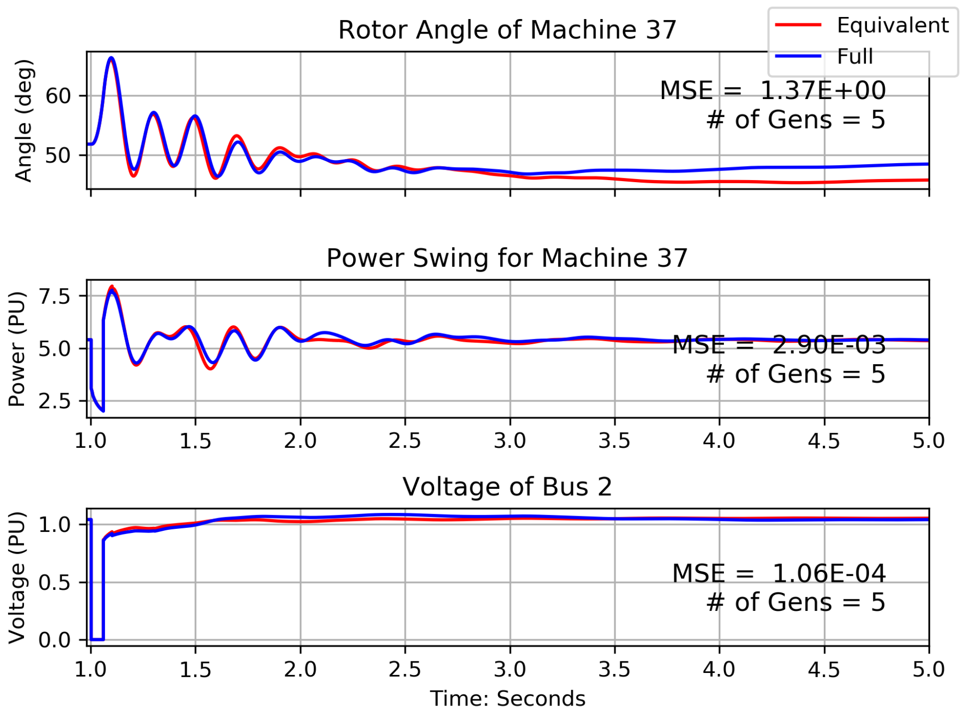
The behavior is the same for converter-based wind. When inside the study area, wind farms can be set to constant negative loads for identification and then full models should be used for simulations. Four type-4 wind generators totaling 80 MW are added to at buses 15,21,24,27. Table 4.4 shows a group of equivalents and the impact of using dynamic models for simulations. The MSE will increase if constant load is used for simulations. However, like HVDC, using constant loads for identification is acceptable. Fig. 4.10 shows both cases (i.e., constant load for identification and constant load for identification and dynamic simulation) for when one mode is preserved (N = 1, t = 0.99).

Table 4.4: Impact of wind generation

				Mean Square Error (MSE)		
Case	**# of Modes (N)**	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
IC	1	0.99	5	1.37E+00	2.90E-03	1.06E-04
AC	1	0.99	5	1.75E+00	3.76E-03	1.11E-04
IC	3	0.97	7	6.36E-01	5.10E-04	2.82E-05
AC	3	0.97	7	1.02E+00	2.13E-03	4.10E-05
IC	4	0.92	8	1.79E-01	2.20E-04	1.67E-05
AC	4	0.92	8	3.99E-01	1.85E-03	2.85E-05
IC	6	0.9	9	4.04E-02	1.27E-04	7.75E-06
AC	6	0.9	9	1.56E-01	1.92E-03	1.95E-05



(a) Fig a-All Constant



(b) Fig b-Identification Constant

Figure 4.10: Preserved modes= 1 (N=1, t=0.99) for system with four wind machines. Top is for all constant and bottom is for identification constant

Inclusion of power electronic-based devices in identification of equivalent groups has significant impact on the accuracy of the equivalent as shown in the previous analysis and results. In addition, power electronic-based devices impact the total number of equivalent groups and the machines included in each group. To conclude, it is important to show the overall impact of power electronics on the equivalent groups. Tables 4.5 and 4.6 show the equivalents for when preserving two modes (N=2, t=0.75) and one mode (N=1, t=0.99). The tables show that the equivalent groups will be impacted by the presence of non-passive devices. The tables also show that even for the same number of equivalent groups, the combination of machines in each group will be different. So it is important to represent the study system with all components when developing the equivalent.

Table 4.5: Impact of non-Passive devices on accuracy

				Mean Square Error (MSE)		
Case	**# of Modes (N)**	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
No H+W	1	0.99	5	1.88E+00	1.75E-03	6.48E-05
H	1	0.99	5	2.46E+00	2.38E-03	1.16E-04
H+W	1	0.99	5	1.37E+00	2.90E-03	1.06E-04
No H+W	2	0.75	6	2.16E+00	2.34E-03	6.80E-05
H	2	0.75	6	1.68E+00	1.95E-03	1.00E-04
H+W	2	0.75	6	6.29E-01	5.13E-04	2.80E-05

No H+W: Case without HVDC and Wind

H: Case with HVDC only

H+W: Case with both HVDC and Wind

Table 4.6: Impact of none passive devices on equivalent groups

Case	# of Groups (g)	Equivalent Groups
No H+W	5	30,37,39,41(equivalent for 31,32,33,38),43(equivalent for 34,35,36)
H	5	30,37,39,41(equivalent for 31,32,38),43(equivalent for 33,34,35,36)
H+W	5	30,37,39,106(equivalent for 31,32,33,38),108(equivalent for 34,35,36)
No H+W	6	30,37,39,41(equivalent for 31,32),43(equivalent for 33,34,38),45(equivalent for 35,36)
H	6	30,31,37,39,41(equivalent for 32,33,34,38),43(equivalent for 35,36)
H+W	6	30,37,38,39,106(equivalent for 31,32),108(equivalent for 33,34,35,36)

No H+W: Case without HVDC and Wind

H: Case with HVDC only

H+W: Case with both HVDC and Wind

4.4.5. Principle Five: Include the Impact of the Type of Disturbance on Group Identification

When developing an equivalent, the equivalent should work for a variety of disturbances and should be tested and compared against the full model for each type of disturbance to be studied. When the type of disturbance is not specific, and the equivalent model will be used for variety of studies, a sample set of disturbances should be tested to ensure that the equivalent is accurate. As discussed, slow coherency is a technique that is known to be disturbance-independent and can be affective in developing an equivalent for a variety of studies. However, if the disturbance is large enough to start impacting the external area significantly, then even techniques that are not disturbance independent could start to yield inaccurate results. So the fifth principle is that the type of disturbance needs to be taken into account when developing equivalent groups.

Using IEEE 39-Bus system, Table 4.7 shows errors with different disturbances inside the study area for the five-machine equivalent obtained when one mode (N = 1, t = 0.99) and the seven-machine equivalent obtained when three modes (N=3, t = 0.97) are preserved. Machines 30, 39, and 37 are retained and the system used includes HVDC and wind machines. For the simulations shown in the table, no specific information was used to form the equivalent groups based on the type of disturbance. The MSE shows that regardless of disturbance, the equivalent is still acceptable. Also, as expected, with more modes captured MSE will be smaller. Fig. 4.10 shows four disturbances when three modes are preserved (N=3 and t = 0.97).

Table 4.7: Impact of different type of disturbances on the accuracy of the equivalent model

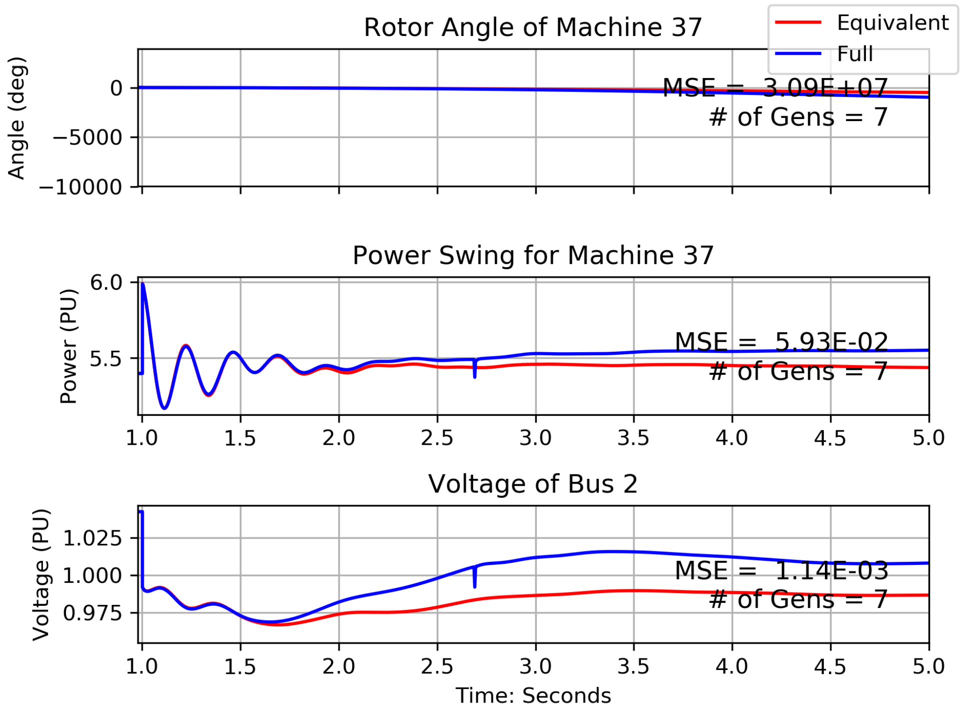
				Mean Square Error (MSE)		
Case	**# of Modes (N)**	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
BF	1	0.99	5	1.37E+00	2.90E-03	1.06E-04
BFBT	1	0.99	5	1.55E+01	1.62E-02	2.19E-04
TM	1	0.99	5	6.17E+07	9.93E-02	2.36E-03
TB	1	0.99	5	2.61E+00	1.71E-03	9.26E-06
BF	3	0.97	7	6.36E-01	5.10E-04	2.82E-05
BFBT	3	0.97	7	1.03E+00	1.81E-03	6.92E-05
TM	3	0.97	7	3.09E+07	5.93E-02	1.14E-03
TB	3	0.97	7	1.27E-01	4.37E-04	8.96E-06

BF: Bus Fault

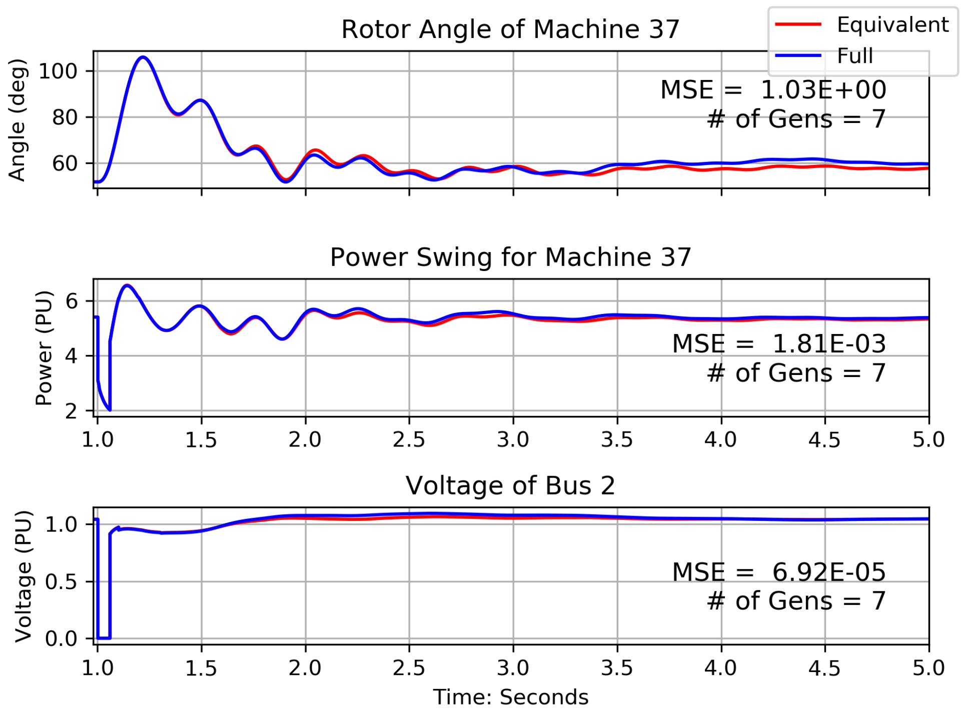
TM: Trip Machine 30

BFBT: Bus Fault and Branch Trip

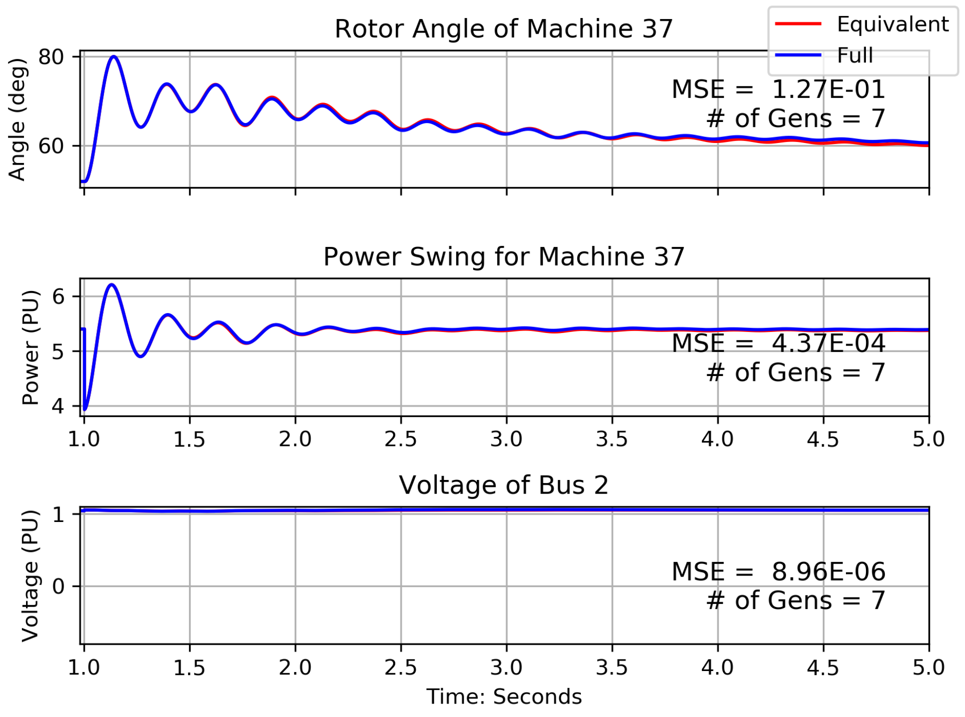
BT: Trip Branch



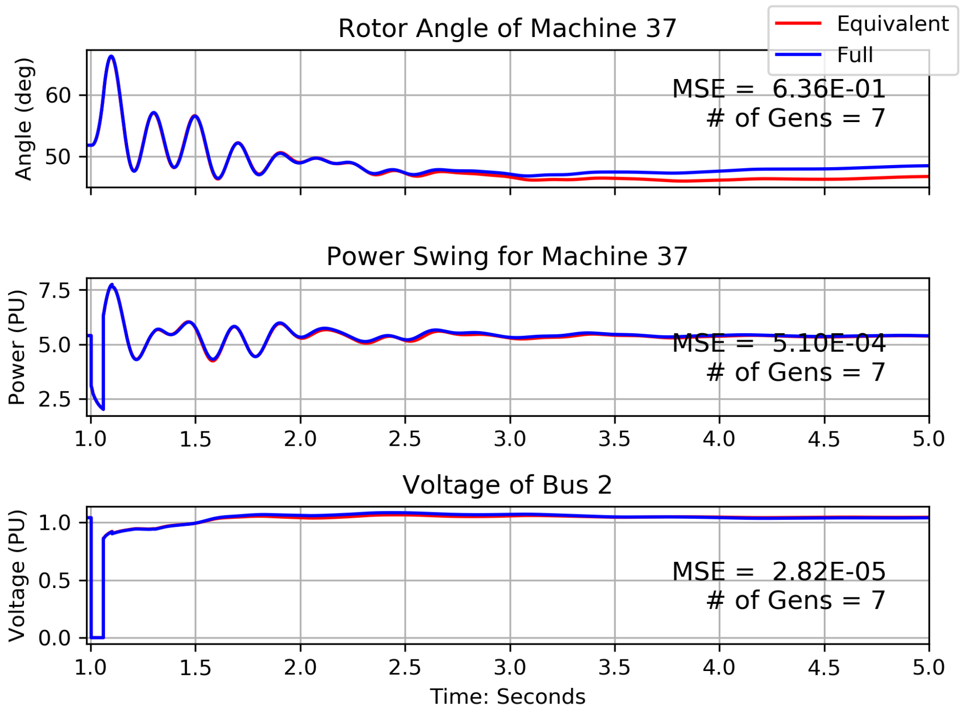
(a) Fig a-TM



(b) Fig b-BFBT



(c) Fig C-TB



(d) Fig D-BF

Figure 4.10: Preserved modes = 3 (N=3, t= 0.97) - impact of different disturbances

4.4.6. Principle Six: Represent the Equivalent Groups Accurately

The aggregate of each equivalent group is a generator with the aggregate parameters that cumulatively represent the dynamic response of all machines within that equivalent group. It is also required to maintain power flow out of each machine. That is, the equivalent model should preserve power flow out of each machine to the rest of the network. The equivalent should be developed by connecting all equivalent machines to a common bus and forming an aggregate model with parameters from the machines included in the equivalent group. The parameters are derived by doing an inertia-weighted-sum of all parameters to ensure that stronger machines have bigger impact on the dynamics of the aggregate machine. As shown in Section 3.2.2.1 and associated Table 3.2, the impedance parameters of the equivalent machine are derived by adding the impedances in parallel. The inertia of the equivalent machine is the summation of the individual machines inertia. Time constants are inertia-weighted-summations of individual time constants.

The objective of identifying equivalent groups is to replace each group with an aggregate model that produces an aggregate dynamic behavior that is reasonably accurate. When developing dynamic model equivalents, performance requirements should be specified in advance with maximum mismatch tolerance for each system condition under study. Representing each equivalent group of machines with an equivalent machine has the following performance requirements. These performance requirements will be covered in detail in the next sub-sections.

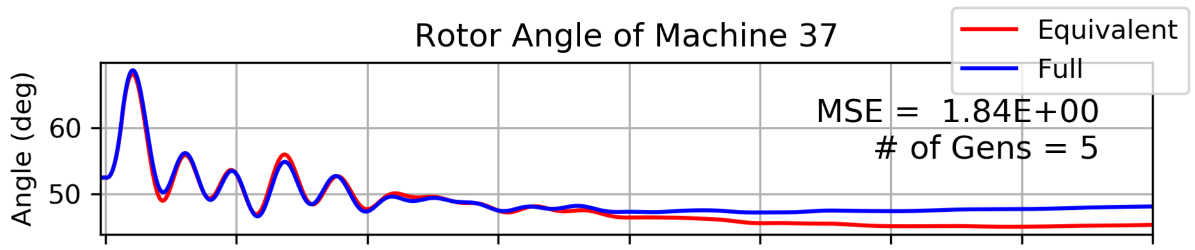
- Dynamic behavior accuracy
- Steady state behavior accuracy (i.e., power flow mismatch and short circuit fault level mismatch)

4.4.6.1. Dynamic Behavior Accuracy

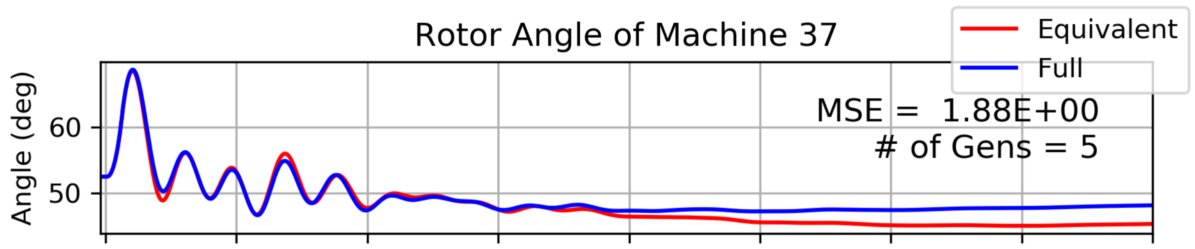
Accuracy can be measured by various error calculation methods. This thesis used the mean square error method to measure the dynamic accuracy. Dynamic accuracy is a crucial requirement in developing the aggregate and is influenced throughout the whole development of the equivalent model. Specifically, it is impacted by the following:

- Number of equivalent generators (i.e., percentage of reduction)
- How each equivalent group is modeled and the level of modeling detail used for each equivalent generator
- Static network reduction (i.e., percentage of reduction in static components)

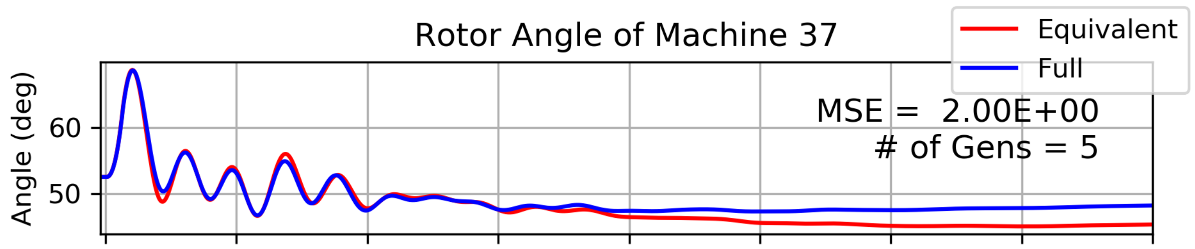
Hence, it is important to specify and monitor the dynamic behavior accuracy throughout the equivalent model development process. Although equivalent groups are identified in the identification step, the grouping can be changed by separating machines or defining a bigger study area which, as shown previously, have a significant impact on the accuracy of the equivalent. The key is to use a good compromise between error and the size of the equivalent. When the system is complex, and a study area cannot be defined easily, a series of trial and error would be required to reach a good compromise between accuracy and size of the equivalent. Accuracy can also be impacted by how the aggregate machine is added to the model. As mentioned in Section 3.2.2.1, aggregating each equivalent group individual machines can be done by connecting their terminal buses (i.e., terminal aggregation) or their internal buses (i.e., inertial aggregation and impedance-compensated inertial aggregation). This impacts the accuracy of the resultant slow modes of the equivalent model in comparison to the full model. Connecting the terminal buses of machines is the least complex, but it generally results in lower accuracy. The accuracy is improved when connecting internal buses. Fig. 4.11 shows rotor angles using terminal aggregation, inertial aggregation and impedance-compensated inertial aggregation. It is important to select the right mix of aggregation techniques that results in highest overall accuracy. It is recommended, however, to start with the terminal aggregation and then use inertial aggregation only if the slow modes are significantly impacted resulting in inaccurate dynamic performance. In [37], it is suggested to calculate the impedance-compensated aggregate model and check if the load introduced to correct power flow mismatch is of significant size which consequently leads to this technique being less accurate than the terminal and inertial aggregation techniques. This requires doing all the complex computations to develop the aggregate in order to judge whether it will be accurate or not to use. So, contrary to the recommendation in the referenced paper, it is recommended to start with the least complex aggregation technique instead. This is proven in Fig. 4.11 where the difference in accuracy did not warrant the extra computations associated with the impedance-compensated inertial aggregation technique.



(a) Fig a-Terminal Aggregation



(b) Fig b-Inertial Aggregation



(c) Fig c-Impedance-corrected intertial aggregation

Figure 4.11: Preserved mode = 1 (N=1,tol=0.99)- different aggregation techniques

Long simulation studies can show adverse effects of equivalent on the specific modes. When developing equivalents for long duration studies, benchmarking of the equivalent must be done using long time simulations to ensure that the accuracy of the equivalent holds for the whole simulation time.

4.4.6.2. Short Circuit Level Mismatch

Short circuit analysis should be used as a measure to test the impact of the aggregate on the equivalent external network impedance seen from the study area. Short circuit levels between the full model and equivalent at the boundary of study area should be unchanged [43]. Table 4.8 shows the short circuit mismatch between the full model and equivalent model at the boundary for the five-machine equivalent of original IEEE 39-Bus system, the modified system with HVDC line and the modified system with both HVDC + wind generators.

Table 4.8: Short circuit mismatch

Case	**Bus Number**	**3-Phase Short Circuit (kA) Full Model**	**3-Phase Short Circuit (kA) Equivalent Model**	**Difference (%)**
No HVDC +Wind	1	10.68	10.67	-0.05
	3	13.93	13.89	-0.31
	9	10.30	10.27	-0.18
	26	9.57	9.57	0.02
HVDC Only	1	7.18	7.18	-0.01
	3	13.21	13.14	-0.53
	9	10.31	10.28	-0.27
	26	9.42	9.40	-0.26
HVDC +Wind	1	7.18	7.18	-0.01
	3	13.32	13.23	-0.63
	9	10.30	10.26	-0.37
	26	9.53	9.50	-0.37

4.4.6.3. Power Flow Mismatch

Power flow at the boundary of the study area must match the full model. That is because any error in power flow means that the equivalent has resulted in changing the topology of the external system to the level that impacts power flow into the study area. Aggregation methods that use means to preserve power flow (e.g., by using ideal phase shifters) seamlessly result in no power flow mismatch.

Table 4.9 shows power flow mismatch at boundary buses between original full case and the equivalent case with 5 machines.

Table 4.9: Power flow mismatch

Line	Power Flow Mismatch (MW)	Power Flow Mismatch (MVar)
1-2	0	0
1-39	0	0
3-2	+0.3	-0.1
3-4	-0.4	-0.5
3-18	+0.1	-0.2
9-8	0	+0.2
9-39	0	-0.2
26-25	-0.3	-0.3
26-27	-0.5	-0.7
26-28	+0.4	+0.5
26-29	+0.4	+0.5

4.4.7. Principle Seven: Preserve Geographical Network Features

Experience on the power system under study is key in developing equivalents. It can be used to specify a more detailed study area as explained in Section 4.4.3. In some instances, aggregation needs to consider specific geographical network features that are ignored when the groups of equivalents are created based on coherency of swing modes. In that case, aggregation can be customized to keep certain machines separate or in two or more subgroups to preserve the geographical network features.

4.4.8. Principle Eight: Model Generator Controls for the Study

The eighth principle is to ensure that control systems of the equivalent generators are modeled appropriately for the study need. Generally, in applications requiring low frequency model equivalents, aggregated external generators do not need to have detailed controls modeled. This is because controls provide damping which should not impact the accuracy in the study area if the study area is selected appropriately. Aggregating generator control systems require detailed analysis to ensure parameters of the aggregated models represent the overall response of the individual generators. In addition, modeling detailed controls would require splitting equivalent groups based on the type and models of controls. Finally, as stated previously, when controls are aggregated and tuned for a specific disturbance, they will need to be re-tuned for any other disturbance as they will not be disturbance-independent.

All studies conducted in this chapter and next chapter did not model any controls for the equivalent groups and showed acceptable performance. This could be different if a significant reduction is required where the study area would be limited to one machine. In that case, controls may need to be modeled to a certain degree.

4.4.9. Principle Nine: Select the Static Network Reduction Methodology

Amongst the available techniques mentioned in Chapter 3, the reduction technique should be selected appropriately for the purpose of the study. For example, studies that would required modeling different load conditions (e.g., seasonal) would require using REI method to aggregate regional loads separately to allow changing the load conditions per region without performing static network reduction for each load conditions.

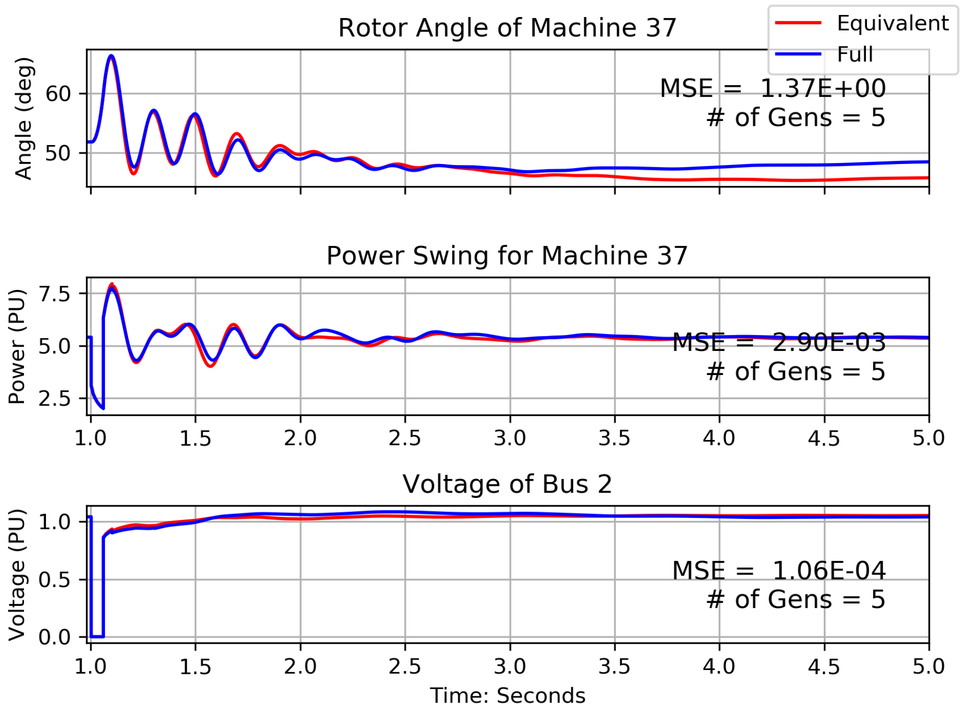
4.4.10. Principle Ten: Reduce Static Network Maintaining the Accuracy of Equivalent Model

Static network reduction should not have a significant effect on accuracy in transient (i.e., phasor type) dynamic simulations unless the voltage and current injections through the boundary buses to the external system change significantly by the disturbance being studied. In this case, the boundary buses should be extended further to ensure that accuracy is not affected. It is not recommended to include any dynamic devices (e.g., generators) in the static network reduction unless the device dynamic behavior can be ignored completely. The main objective of static network reduction is to reduce the size of the network and achieve faster simulations. In the case of the IEEE 39-Bus system, which is a relatively small model to begin with, the simulation time savings between the aggregate five-machine equivalent model with no static network reduction and the model with the static network reduced was found to be 20%. In larger systems, it is expected that the simulation time saving will be more significant. Choosing to perform static network reduction should be based on the size of the network and the impact on simulation time. When the equivalent model is to be used in electromagnetic transient studies, then reducing the static network size exhibits a significant impact on the simulation time depending on the frequency models used in the study. Fig. 4.12 shows the 5-machine aggregate model shown in previous sections with static network reduced in accordance to Table 4.10 shown below. The network reduction shown was obtained by retaining all machines and selecting the following buses as boundary buses: 1,2,3,9,25,26,30,37,39. The reduction in the number of buses was 68%.

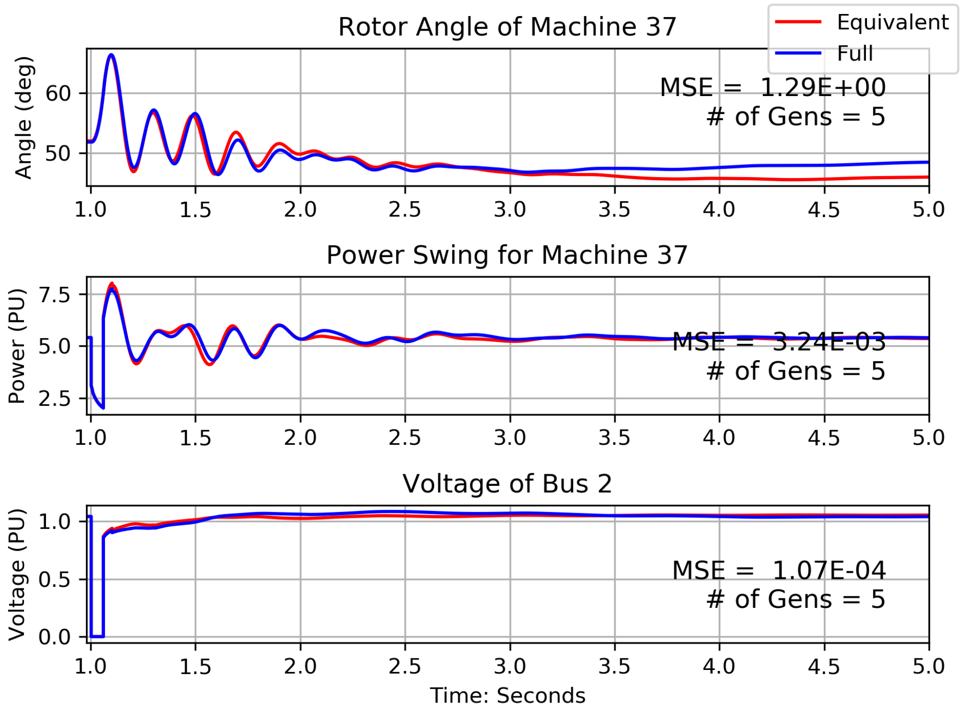
Table 4.10: Static network reduction of IEEE 39-bus Five-machine equivalent model

Component	**Full Model**	**Reduced Model**	**% Reduction**
# of Buses	47	15	68%
# of Lines	35	40	114%(increase[1])
# of Transformers	23	2	91%

[1]# of branches was not reduced but instead increased because of the number of connections to the specified boundary buses that need to be retained to ensure power flow from and to boundary buses is unchanged.



(a) Fig a-Before Static Network Reduction



(b) Fig b-After Static Network Reduction

Figure 4.12: Five-machine equivalent with and without static network reduction

4.4.10.1. MW/MVar mismatch

Similar to the aggregation stage, the MW/MVar mismatch should be minimized. Steady state MW/MVar flow between the study area and external area should not be affected by static network reduction. Likewise, unless dynamic devices are included in the static network reduction, as discussed previously, dynamic power flow between the reduced model and full model should not be affected by static network reduction. Table 4.11 shows the impact of static network reduction discussed previously on MW/MVar mismatch.

Table 4.11: Power flow mismatch - IEEE 39-bus five-machine equivalent

Line	**Power Flow Mismatch (MW)**	**Power Flow Mismatch (MVar)**
3-2	0	-0.6
9-39	0	-0.5
26-25	0	+0.9

4.4.10.2. Short circuit mismatch

Short circuit mismatch after static network reduction should match the full model. Table 4.12 shows the difference in short circuit levels before and after static network reduction for bus number 3. The accepted mismatch could be relaxed taking into account the consequence of short circuit mismatch on the study, but it should be generally minimized.

Table 4.12: Impact of static network reduction on short circuit mismatch

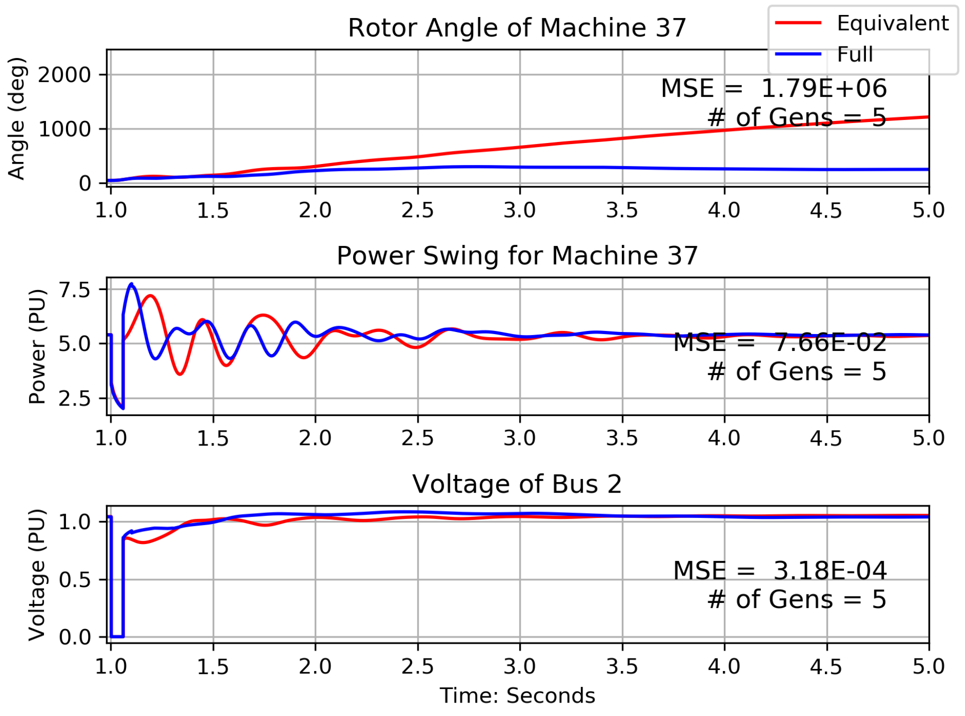
Case	**Bus Number**	**3-P (kA)**	**Difference (%)**
Full Model	3	13.3	
Five Machine Equivalent	3	13.2	-0.6%
Five Machine Equivalent (Static Network Reduced)	3	13.4	0.8%

4.5. Benefits of Following Consistent Guidelines

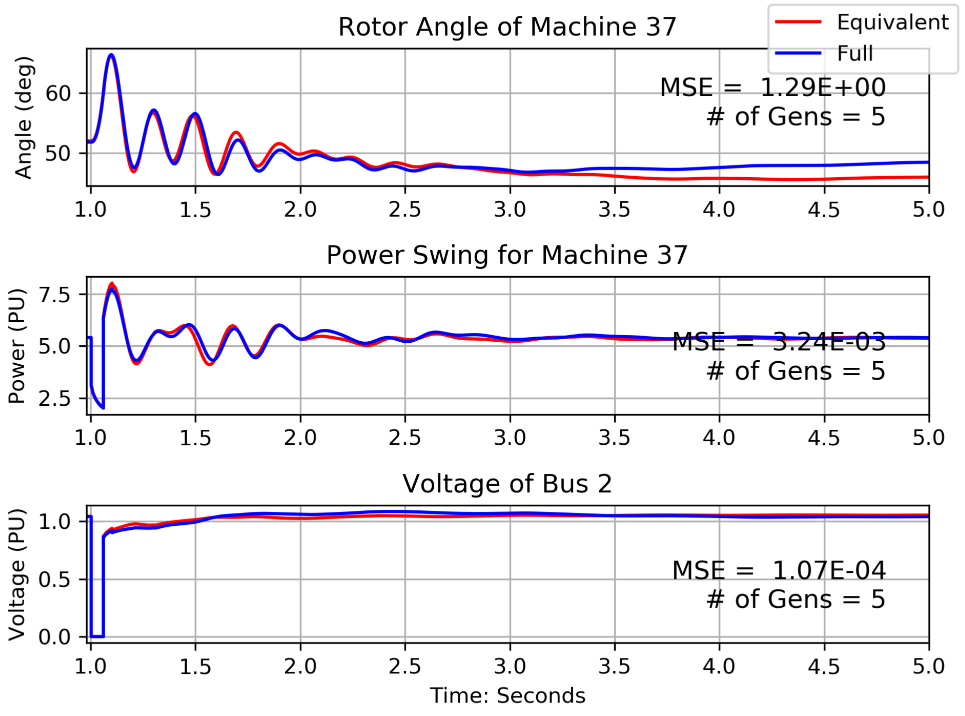
The impact of developing low frequency model equivalents without a consistent set of guidelines was demonstrated implicitly throughout this chapter by comparing the impact of following each guideline principle in the dynamic and steady state accuracy of the equivalent model. This section will attempt to compare the overall cumulative impact on the accuracy of the equivalent when following the guidelines in this chapter against not following any guidelines at all. Two equivalent models were developed as follows:

- Model #1: This model is developed without following any of the guidelines in this chapter. Model features:
    - Two slow modes are preserved, randomly selected as the two slowest modes, and only machine 37 is selected and included in the study area.
    - All non-passive devices are represented as constant sources in the coherency identification and dynamic simulation steps.
    - Static network components are reduced significantly
- Model #2: This model is developed following all the guidelines proposed in this chapter. Model features:
    - One slow mode is preserved, but more detailed study area is selected.
    - All non-passive devices are represented as constant sources in the coherency identification stage, but full models are used for dynamic simulations.
    - Static network components are reduced monitoring the dynamic model accuracy.

Fig. 4.13 shows the performance of Model #1 and Model #2. Although the resulting number of generators is five in both models (reduction of 50%), MSE difference between the two models clearly confirms the importance of following consistent guidelines to develop dynamic model equivalents.



(a) Fig a-Model #1 resulting from not following guidelines



(b) Fig b-Model #2 resulting from following guidelines

Figure 4.13: Impact of following consistent guidelines

4.6. Summary

This chapter presented a general procedure and a set of guidelines to develop low frequency model equivalents. The procedure and guidelines were demonstrated using the New England IEEE 39-bus benchmark system. The procedure consists of four main stages and the guidelines were presented as ten principles. The procedure and guidelines are purposed to achieve consistent and accurate model equivalents regardless of the method used or the entity in charge of developing the equivalent. The guidelines demonstrated clearly the impact of not following consistent methodology and guidelines when developing equivalent systems and the importance of defining performance requirements as part of the development of model equivalents.

Next chapter includes an application of the same procedure and guideline presented in this chapter on Alberta Interconnected Electric System to show the effectiveness of the general procedure and guidelines in acheiving consistent equivalents on real system models.

Chapter 5

Application on Alberta Interconnected Electric System

5.1. Introduction

This chapter covers the application of the procedure and guidelines described in Chapter 4 on the Alberta Interconnected Electric System (AIES). The AIES was used as an example to demonstrate the application of the proposed procedure and guidelines on a real system model. The purpose of the demonstrations in this chapter is to further confirm the validity of the guideline and show the importance of following consistent guidelines and principles when developing low frequency model equivalents. In addition, it brings forward some of the challenges that arise when developing low frequency model equivalents on real system models.

The proposed procedure in Chapter 4 was followed to develop the equivalent for the AIES. However, this chapter is focused on the guideline principles and the discussion will be based on confirming the guideline principles.

5.2. Description of the AIES System Model

The AIES system model used in this chapter is a true model of the AIES. It is a summer peak study case representing the system in 2016. The model contains the following features:

- Two HVDC links connecting the north of the province to the south of the province
- 155 rotating machines with full dynamic models of which 133 are synchronous generators and 22 are wind generators. All machine models have various control system models included as well.
- Seven static var compensators

- Two FACT devices
- Three interconnections to neighboring systems
- Total generation: 10,662 MW
- Total load: 10,340 MW

The AIES is a reasonable sample of an electric system anywhere in the world with classical and modern network components included. The study case and all models are those that would be used for real studies conducted on Alberta system.

The studies shown in the following section will be for a specific area in the system and will be named the Study Area. The Study Area is a major generation and load center and is close to the south terminals of both HVDC converter stations. Unless otherwise indicated, the disturbance used in these studies is a three-phase-to-ground fault at machine A. Machine A is a combined cycle generator in the Study Area.

5.3. Application of Guidelines to Alberta Interconnected Electric System

The following subsections details the application of the guideline principles on Alberta Interconnected Electric System.

5.3.1. Principle One: Define Equivalent Model Requirements

The study case was prepared by checking dynamic models. Ensuring that models initialize correctly and that the study case contains no errors.

The following modifications were done:

- All machines that do not have dynamic models were replaced by negative loads
- All machines with classical dynamic models were replaced by negative loads
- All machines with non-generic dynamic models were replaced by negative loads

After the above modifications were completed, one more check was conducted to ensure that the case data extracted by the model equivalencing tool is correct. One error found was that transformers were not defined consistently in the case. The parameter that defines which winding is the from and two winding was not consistent and resulted in extracting the wrong ratio. This was corrected by defining the transformer windings consistently in the study case.

The following requirements were defined for the dynamic equivalent:

- Reduce the number of machines by approximately 50%
- Reduce the number of buses by approximately 50%
- Steady state MW/MVar power flow at boundary buses matches the full model
- Short circuit fault level at all boundary buses matches the full model
- Dynamic accuracy error should be reasonably small
- Equivalent model should be accurate for a three-phase-to-ground at the study machine terminals, line trip, machine trip and three-phase-to-ground fault at HVDC terminal disturbances.

5.3.2. Principle Two: Preserve Modes in the Range 0.2 Hz - 2 Hz

No specific range of frequencies is of interest in this study. The requirement is to preserve as many as possible without exceeding the target reduction ratio of 50%.

5.3.3. Principle Three: Define Study Area Boundary

The guideline principle for study area boundary is to retain enough detail in the study area in order to select fewer modes to preserve in the equivalent resulting in more accurate and smaller equivalent model. Given the number of machines in close proximity to the study machine, the appropriate definition of the Study Area would be to include all machines and buses in the whole planning area. However, to confirm this guideline principle, two equivalent models were developed:

- Equivalent Model #1: Retain all machines in the Study Area plus the machines representing the inter-tie to British Columbia; preserve 30 slow modes
- Equivalent Model #2: Retain Machine A only in the Study Area plus the machines representing the inter-tie to British Columbia; preserve 30 slow modes
- Equivalent Model #3: Retain Machine A only in the Study Area plus the machines representing the inter-tie to British Columbia; preserve 45 slow modes

In all equivalents, all HVDC converters were replaced by constant loads for identification stage and then dynamic models were used for the simulations. In addition, all renewable generators were replaced by constant negative loads for the identification stage and then dynamic models were used for the simulations.

Figures 5.1 and 5.2 show the result of Equivalent Model #1 and #2 respectively. Table 5.1 shows a summary of the findings. It is evident that selecting a reasonably detailed study area resulted in a more accurate model. However, in this case the model resultant from the detailed study area has more equivalent machines (57 vs. 53). This is not the reason why it is more accurate. In fact, looking at Equivalent #3, which has 45 modes selected with the same study area as Equivalent #2, it results in bigger (64 machines) and less accurate equivalent. Equivalent #3 is shown in Fig. 5.3.

Table 5.1: Impact of study area boundary - AIES

				Mean Square Error (MSE)		
Case	**# of Modes (N)**	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
Equivalent #1	30	0.95	57	6.07E-01	7.85E-05	2.14E-07
Equivalent #2	30	0.95	53	9.48E-01	3.66E-04	9.59E-07
Equivalent #3	45	0.90	64	5.73E-01	1.48E-04	9.85E-07

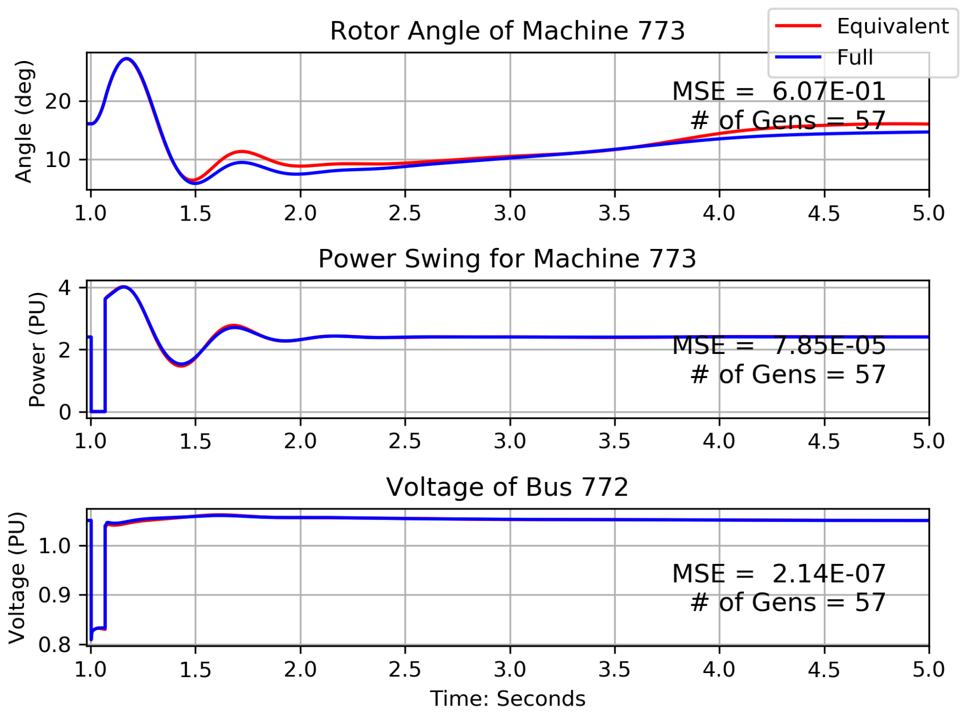


Figure 5.1: AIES - detailed study area

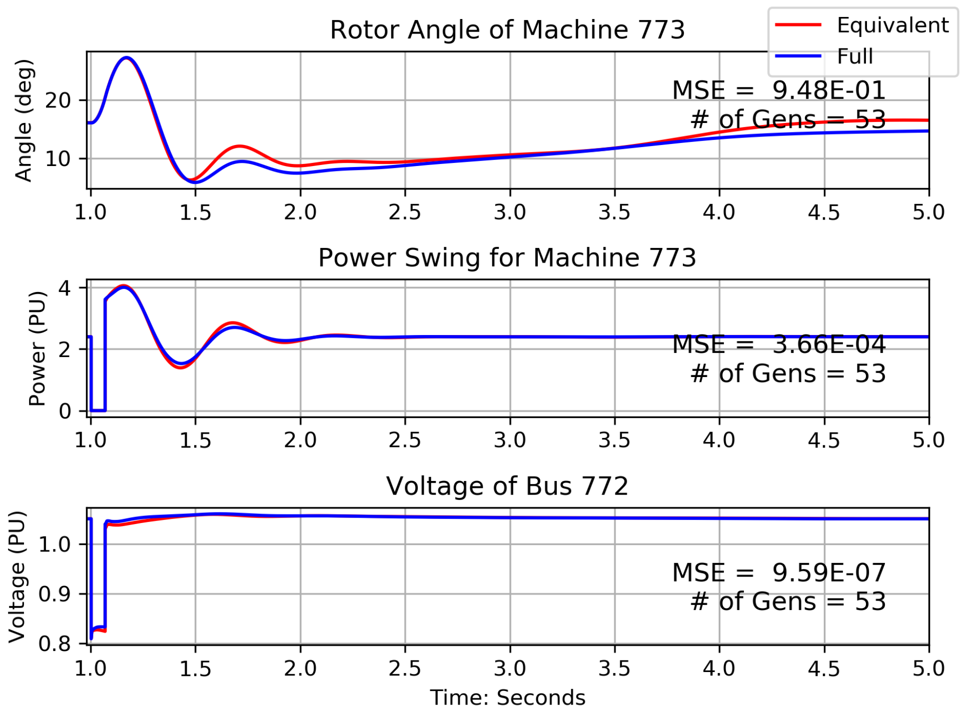


Figure 5.2: AIES - small study area

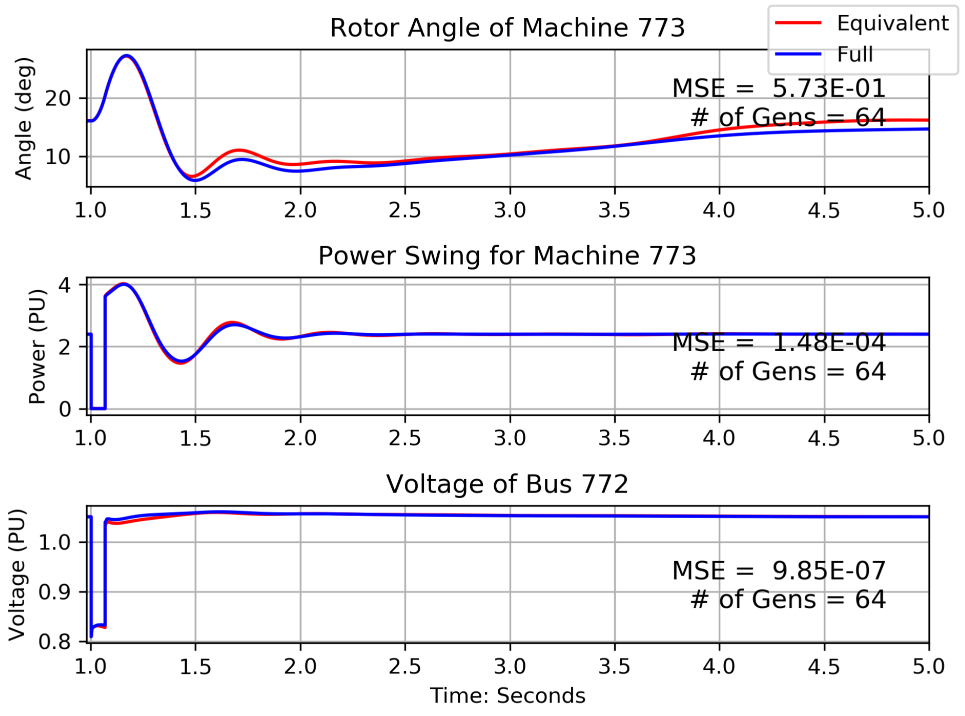


Figure 5.3: AIES - small study area with 45 modes preserved

5.3.4. Principle Four: Include the Impact of Non-passive Devices on Group Identification

As shown in Chapter 4, non-passive devices (i.e., HVDC converters and full converter renewable generation) need to be included for identification in the form of constant loads. However, the full models need to be used for dynamic simulations unless the non-passive device are far away from the study area. To confirm this guideline principle, two equivalent models were developed:

- Equivalent Model #1: Replace non-passive devices with constant loads for equivalent groups identification, then use full dynamic models for simulation; preserve 30 modes

- Equivalent Model #2: Replace non-passive devices with constant loads for equivalent groups identification, then use the same constant loads for simulation; preserve 30 modes

Table 5.2 shows the error difference between the two equivalents. The MSE difference is more significant for voltage (approximately 6 times). The difference will be even more pronounced if the disturbance was close to the terminals of the HVDC converters. Equivalent #1 was previously shown in Fig. 5.1. Fig. 5.4, shows Equivalent #2.

Table 5.2: Impact of non-passive devices - AIES

				Mean Square Error (MSE)		
Case	**# of Modes (N)**	**Tolerance (t)**	**# of Groups (g)**	**Rotor Angle**	**Power Swing**	**Voltage**
Equivalent #1	30	0.95	57	6.07E-01	7.85E-05	2.14E-07
Equivalent #2	30	0.95	57	5.89E-01	9.23E-05	1.30E-06

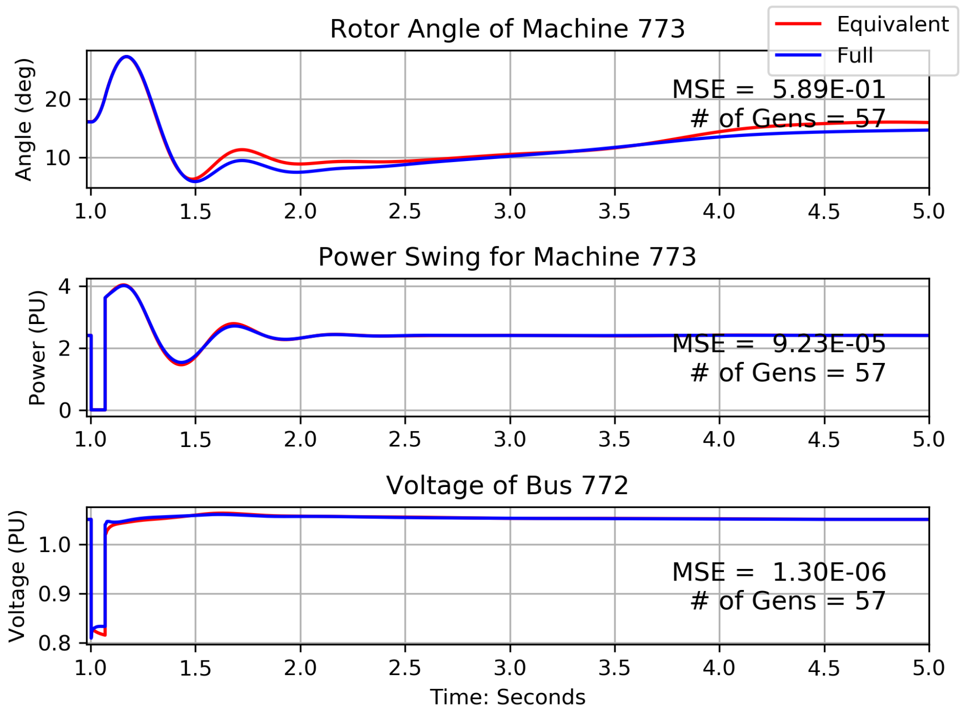


Figure 5.4: AIES - HVDC constant

5.3.5. Principle Five: Include the Impact of the Type of Disturbance on Group Identification

Developed equivalent should work for the disturbance being studied. It is possible to develop an equivalent for a specific study area that works for multiple disturbances. As explained in Chapter 4, equivalents derived by separating slow modes, which are typically associated with inter-area oscillations, is effective in developing equivalent models that work for multiple disturbances. As specified in stage zero, the three disturbances, in addition to the bus fault disturbance, used in the previous subsections were attempted to test this guideline principle. They are described below:

- Study #1: Line connecting machine A high voltage terminal to next substation is tripped, shown in Fig. 5.5
- Study #2: A 240 MW machine in the same generation plant is tripped, shown in Fig. 5.6
- Study #3: Three-phase-to-ground fault at the HVDC converter terminal closest to Machine A, shown in Fig. 5.7

The equivalent derived previously performed well for studies #1 and #2. Study #3 showed worse overall accuracy. This is mainly because of the severity of the disturbance and the fact that the other side of the HVDC link is not included in the study area. Study #3 shows the importance of defining good study area on the accuracy of the equivalent. To confirm that, the study area was extended to include all machines on the other side of the HVDC link while preserving the same number of modes. This is shown in Fig. 5.8. Despite being a bigger model, which is expected after extending the study area, the improvement in accuracy is significant. So, if this disturbance was crucial for this equivalent, then the equivalent must be extended to include the machines on the other terminal of the HVDC line in the study area.

Table 5.3 includes a summary of all three studies and the alternative study done for study #3.

Table 5.3: Impact of disturbance type- AIES

Case	**# of Modes (N)**	**Tolerance (t)**	**# of Groups (g)**	**Mean Square Error (MSE)** **Rotor Angle**	**Power Swing**	**Voltage**
Study #1	30	0.95	57	6.32E-02	5.44E-09	1.54E-09
Study #2	30	0.95	57	3.25E+01	8.85E-06	1.99E-07
Study #3	30	0.95	57	8.90E+01	6.37E-04	3.89E-06
Study #3A	30	0.95	66	1.18E+01	1.33E-04	6.49E-07

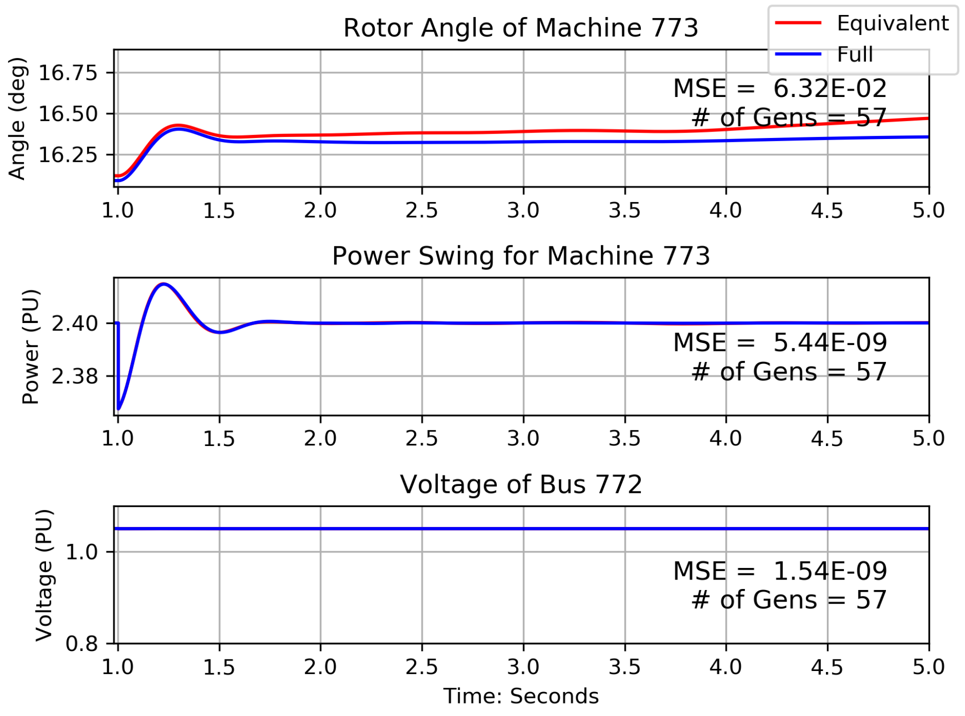


Figure 5.5: AIES - Impact of disturbance - line trip

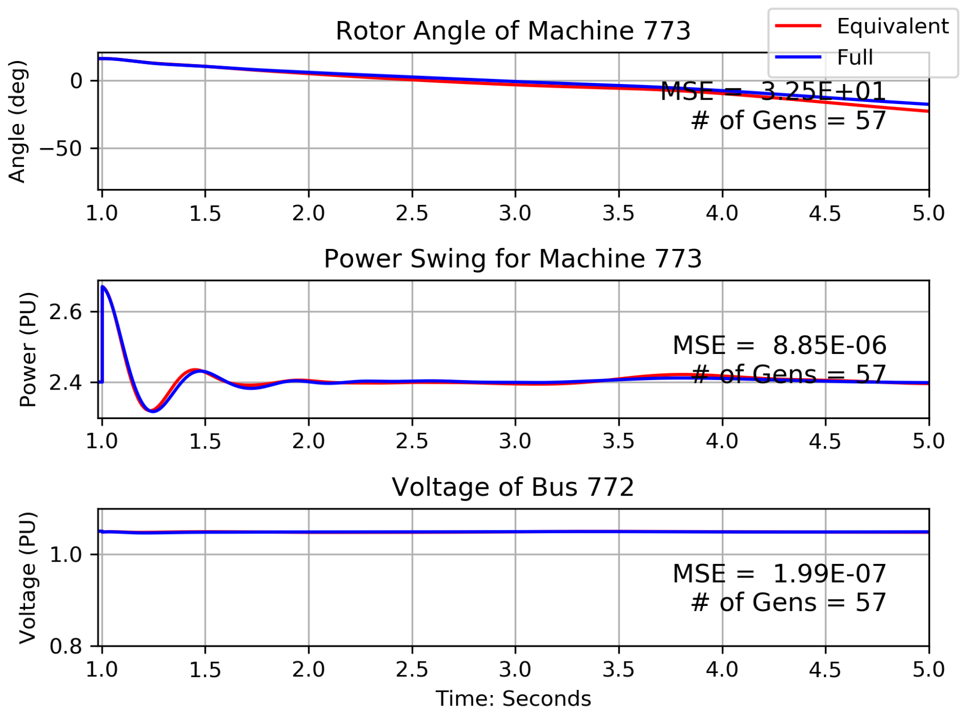


Figure 5.6: AIES - Impact of disturbance - machine trip

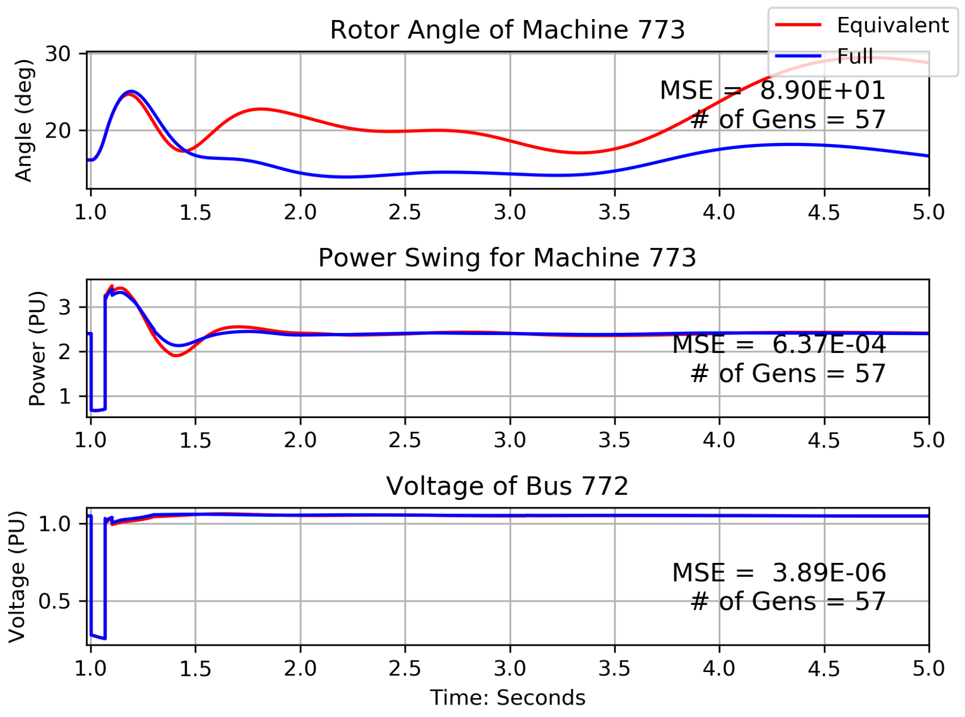


Figure 5.7: AIES - Impact of disturbance - HVDC terminal fault

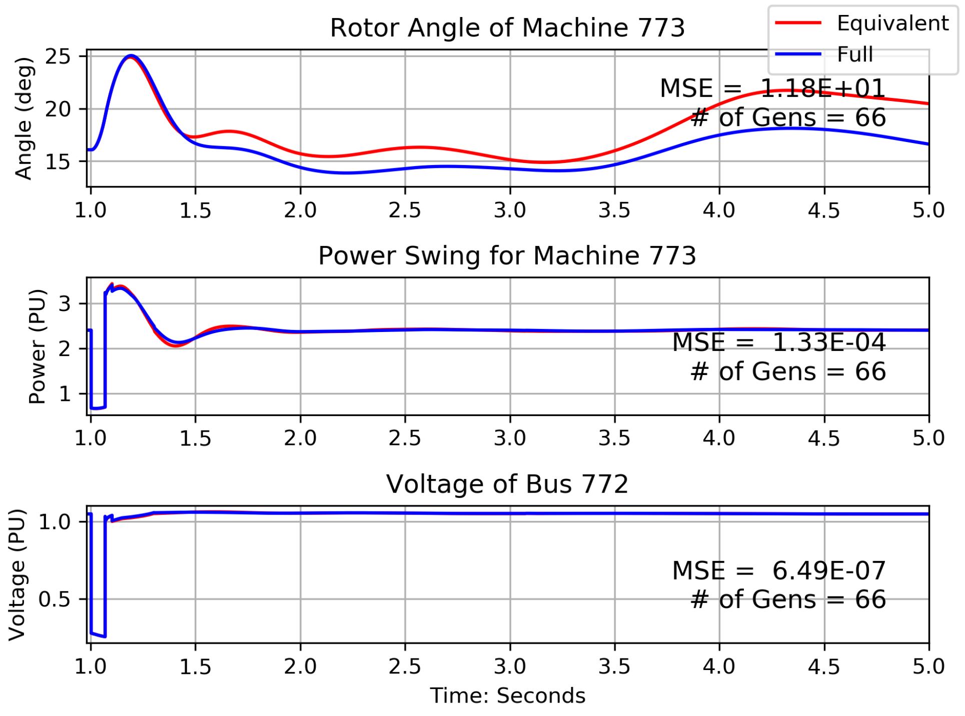


Figure 5.8: AIES - Impact of disturbance - HVDC terminal fault with extended study area

5.3.6. Principle Six: Represent the Equivalent Group Accurately

5.3.6.1. Dynamic Behavior Accuracy

Dynamic behavior accuracy was demonstrated throughout the previous subsections and used as a measure to compare the accuracy of dynamic model equivalents produced. The study area defined, the number of slow modes selected, and how non-passive devices were modeled resulted in a reasonably accurate model which satisfied the dynamic behavior accuracy requirement.

Inertial aggregation was used to form the equivalent generators. It resulted in an accurate model and no further aggregation techniques were attempted.

5.3.6.2. Short Circuit Mismatch

As explained in Chapter 4, short circuit mismatch is an important performance measurement that need to be taken into account when forming the equivalent. Table 5.4 shows the short circuit mismatch error at the boundary of the Study Area used throughout this chapter. The short circuit mismatch is insignificant and the reduced model meets the requirement of the short circuit level mismatch specified in principle one.

Table 5.4: Short circuit mismatch - AIES

Bus Number	**3-Phase Short Circuit (kA) Full Model**	**3-Phase Short Circuit (kA) Equivalent Model**	**Difference (%)**
617	12.95	12.96	+0.03
160	26.29	26.30	+0.05
159	22.54	22.55	+0.04
161	15.86	15.87	+0.05

5.3.6.3. Power Flow Mismatch

As explained in Chapter 4, power flow mismatch is an important performance measurement that need to be taken into account when forming the equivalent. Table 5.5 shows the power flow mismatch at the boundary of the Study Area used through out this chapter. The mismatch is insignificant and the reduced model meets the requirement specified in principle one.

Table 5.5: power flow mismatch - AIES

Line	**Power Flow Mismatch (MW)**	**Power Flow Mismatch (MVar)**
159-480	0	0
159-485	0	-0.1
159-943	0	+0.1
617-643	0	0
617-653	0	0
160-152	0	0
161-155	0	0

5.3.7. Principle Seven: Preserve Geographical Network Features

To preserve geographical and electrical network features, a study was conducted to separate two machines from an equivalent group that was produced by selecting to preserve 30 modes. The two machines belong to one distinct generation plant and are connected to 240 kV system. They were originally grouped along with four other machines connected to 500 kV system. While all six machines are coherent, it would be beneficial to separate the machines connected to 240 kV to preserve the network features. This change preserves an important geographical feature in the AIES. Fig. 5.9 shows the result of this equivalent. The equivalent accuracy is comparable to the original equivalent shown in Fig. 5.1 which proves that preserving geographical and electrical network features in the equivalent should not result in less accurate equivalent. However, it may result in a larger equivalent depending on which features need to be preserved (in this case it did not).

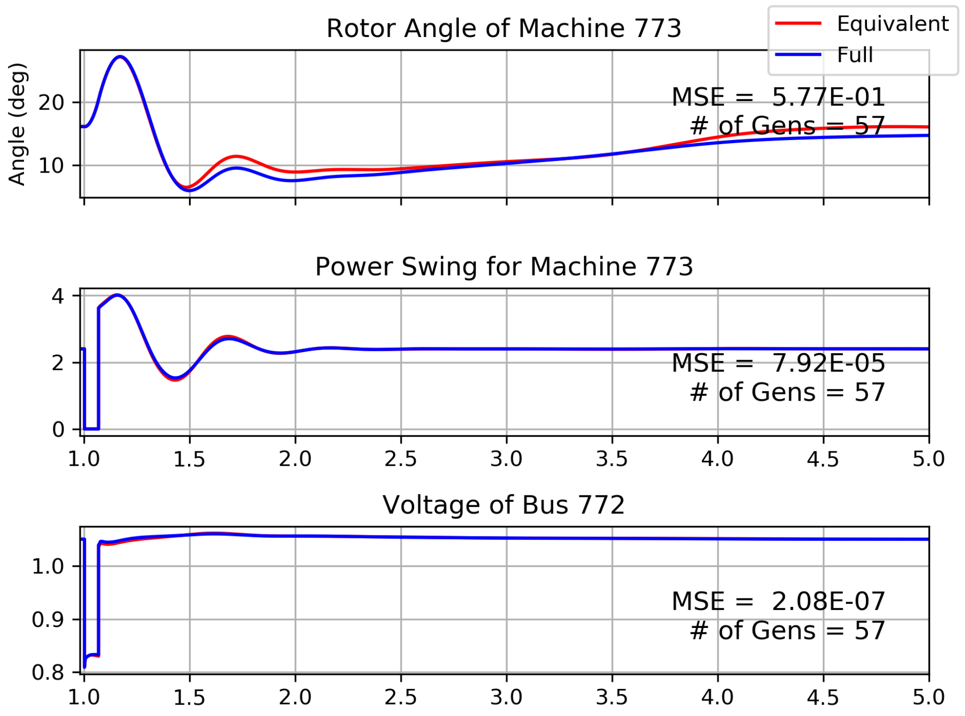


Figure 5.9: AIES - preserving geographical network features

5.3.8. Principle Eight: Model Generator Controls for the Study

All studies conducted did not necessitate developing any controls for the equivalent machines. All equivalent machines were represented by standard PSS®E GENROU and GENSAL generator models. As mentioned in Chapter 4, low frequency model equivalents generally do not require representing equivalent generators with control systems unless an equivalent consisting of just a few machines is required due to extremely limited hardware resources such as in real time simulators.

5.3.9. Principle Nine: Select the Static Network Reduction Methodology

The requirement of this equivalent does not require keeping physical loads in the external area retained in the equivalent. Hence, the equivalent of the static network will be conducted to target 50% reduction in the external area as specified. Static network reduction was conducted on the AIES model using PSS®E EEQV tool. The EEQV tool uses a form of Ward technique to reduce the static network. Two different levels of static network reductions were conducted to ensure that the impact of static network reduction is clearly demonstrated in line with the guidelines presented in Chapter 4. The first level is denoted "Moderate" to show static network reduction level that meets the mentioned guideline and to target 50% reduction. The second level is denoted "High" to show the impact of high static network reduction. Both levels retain all buses in the Study Area. So the difference is in the number of components reduced in the external area. Table 5.6 shows the results. Specifically, the table shows the % reduction in bus and branch static network components and the impact on simulation time. The static network reduction did not result in significant improvement in simulation time for a 20 second simulation when moderate static network reduction is conducted. However, when high static network reduction is conducted, the improvement in simulation time is 44%. This difference can be more significant if the model is used for longer duration studies. The studies in this chapter are not conducted on electromagnetic environment, but if that was the case, then the static network reduction would make a significant difference in simulation time in that type of study. The target reduction of approximately 50% was met.

Table 5.6: Impact of static network reduction on equivalent

Case	# of Machines	# of Buses	# of Branches	Simulation Time
Full Model	133	2482	7395	3.80s
Moderate Static Network Reduction	57	1380	5008	4.44s
High Static Network Reduction	57	351	2210	2.50s
% Reduction	57%	44%	32%	14%
	57%	86%	70%	44%

5.3.10. Principle Ten: Reduce Static Network Maintaining the Accuracy of Equivalent Model

5.3.10.1. Dynamic Behavior Accuracy

As described in Chapter 4, static network reduction should not result in impact on dynamic accuracy. To confirm that, a dynamic study was conducted to shows the difference between the performance of the equivalent model without static network reduction and with the two levels of static network reduction obtained in the previous section. The performance of the equivalents is shown in Table 5.7. As expected and included in the guideline, static network reduction resulted in insignificant impact on the accuracy for the moderate case. The high case results in higher impact. The high static network reduction case would not be a recommended level of reduction. And if followed, can result in inconsistent dynamic results. Fig. 5.10 shows the dynamic behavior of the moderate case. Fig. 5.11 shows the dynamic behavior of the high static network reduction case. The case without static network reduction is the same case shown in Fig. 5.1.

Table 5.7: Impact of static network reduction on dynamic behavior accuracy - AIES

	Mean Square Error (MSE)		
Case	**Rotor Angle**	**Power Swing**	**Voltage**
No Static Network Reduction	6.07E-01	7.85E-05	2.14E-07
Moderate Static Network Reduction	6.63E-01	7.68E-05	2.43E-07
High Static Network Reduction	7.68E+01	9.91E-04	4.28E-05

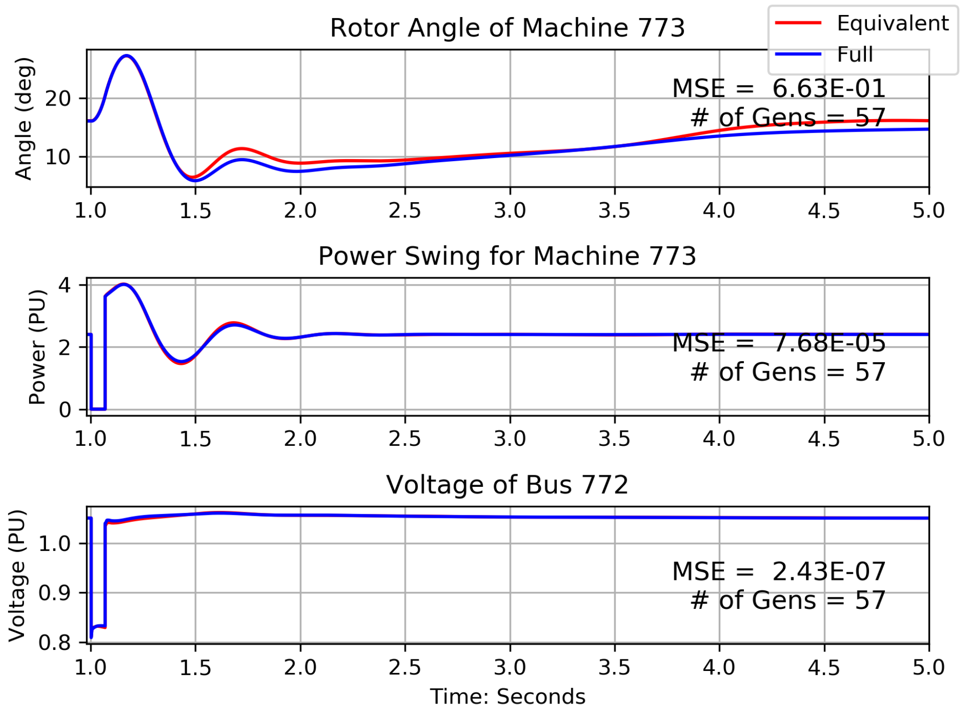


Figure 5.10: AIES - impact of moderate static network reduction on dynamic behavior accuracy

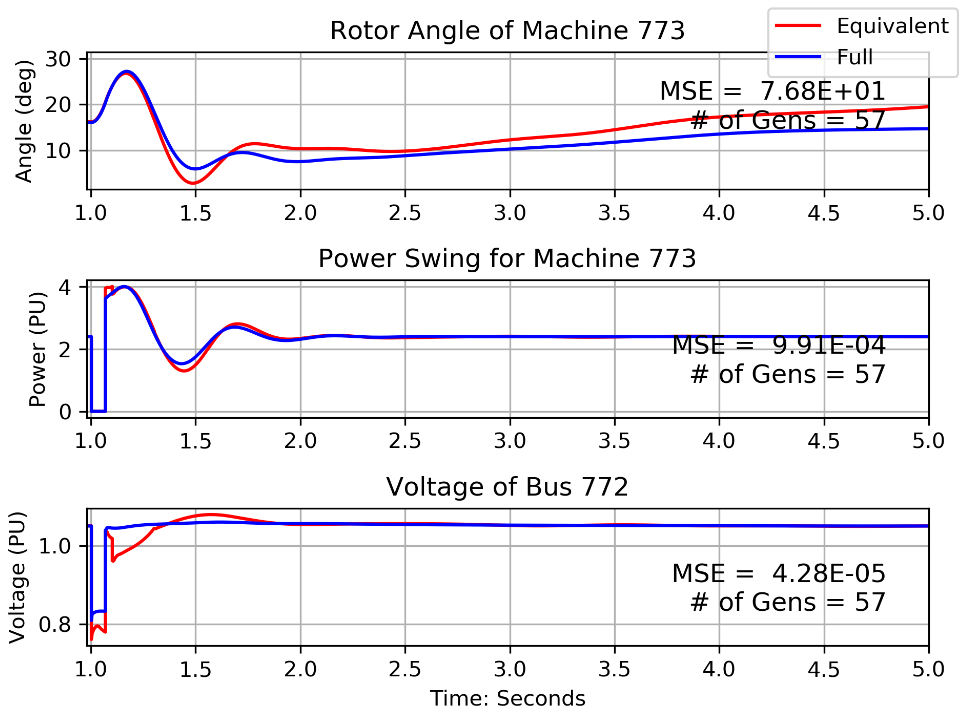


Figure 5.11: AIES - impact of high static network reduction on dynamic behavior accuracy

5.3.10.2. MW/MVar mismatch

Table 5.8 shows the power flow mismatch after static network reduction. For the moderate reduction case, all mismatches are minimal and meet the specified requirement for this equivalent model. For the high reduction case, mismatches of up to 1.6 MW and 4 MVar were found, which is in line with the deteriorated dynamic behavior accuracy. This further confirms that smaller mismatch in steady state power flow is required in order to produce an accurate equivalent.

Table 5.8: Power flow mismatch - AIES

	Moderate Case		**High Case**	
Line	**MW Mismatch**	**MVar Mismatch**	**MW Mismatch**	**MVar Mismatch**
159-480	+0.4	-0.1	-1.6	+0.1
159-485	0	-0.1	0	0
159-943	-0.1	+0.1	-0.1	0
617-643	0	0	0	+0.1
617-653	0	0	-0.1	+0.2
160-152	-0.1	-0.5	+0.2	-1.6
161-155	0	-0.6	+0.7	-4

5.3.10.3. Short Circuit mismatch

Table 5.9 shows the result of the short circuit mismatch after static network reduction for both the moderate and the high reduction case. As laid down in the guidelines of Chapter 4, static network reduction should not result in significant short circuit mismatch. For the moderate reduction case, the errors shown in the table are from the aggregation stage and when compared with Table 5.4, no further error was introduced by static network reduction. For the high reduction case, the error is more significant and in line with the dynamic behavior accuracy impact shown in the previous subsection.

Table 5.9: Short circuit mismatch - AIES

Bus Number	**3-Phase Short Circuit (kA) Full Model**	**Difference (%) Moderate Case**	**Difference (%) High Case**
617	12.95	+0.03	+0.92
160	26.29	+0.05	+0.84
159	22.54	+0.04	+1.14
161	15.86	+0.05	+0.52

5.4. Summary

This section presented an application of the general procedure and guidelines of Chapter 4 on the Alberta Interconnected Electric System. The procedure and guideline, when followed, resulted in an accurate low frequency equivalent that met all the requirements. The application of the procedure and guideline on this real system showed that they are valid regardless of the system used.

Chapter 6

Conclusion

6.1. Chapter Summary

Dynamic model equivalents is an ongoing requirement that did not diminish with the abundance of today's computing resources. Equivalents must be developed in a consistent and reliable manner. This thesis began by reviewing the literature available for developing dynamic model equivalents. The literature is enriched with lots of journal papers on this topic. However, the majority of literature available is specific on the evolution of different dynamic model equivalencing techniques. So the literature spans over the techniques and various testing that was conducted to prove their validity and accuracy. However, the literature lacked a comprehensive guideline on how to consistently develop dynamic model equivalents for different type of studies.

In Chapter 2, the thesis presented a high level overview of the different types of dynamic model equivalents and their associated applications. It also included a review on the different techniques available in literature to develop the different type of equivalents. The chapter highlighted how the topic of dynamic model equivalents is a broad topic with lots of opportunities for research and development.

In Chapter 3, the thesis covered low frequency model equivalent techniques in more detail with some focus on coherency methods. Coherency methods are one of the most popular methods in the industry and are used in commercially available software today. It is also the method that was used in this thesis to demonstrate the proposed guidelines. The chapter provided more details about the different techniques available for the different stages of low frequency model equivalent development using coherency. This chapter also includes a summary of the tools adapted and developed to produce the dynamic equivalents used in this thesis. The tools consist of three main parts: the coherency algorithm, the interfaces to PSS®E, and the dynamic simulation and plotting modules.

In Chapter 4, a general procedure is proposed to develop low frequency model equivalents followed by a set of guidelines aiming at the development of consistent and reliable low frequency model equivalents. The guidelines are split into ten principles over the four stages of low frequency model equivalent development. The ten principles are demonstrated and validated using New England IEEE 39-bus test system. The demonstrations both validate the guideline principles proposed and show the impact of not using consistent guidelines in developing low frequency model equivalents.

In Chapter 5, the procedure and guidelines proposed in Chapter 4 are applied on Alberta Interconnected Electric System as a demonstration on a real system model to validate the ten principles mentioned in Chapter 4. In addition, the application shows the impact of following a consistent methodology in developing equivalents on the accuracy and reliability of the equivalent. Moreover, this chapter showed the challenges that arise when applying the proposed procedure and guideline on a real system.

The proposed procedure and guidelines were developed generically to work on any system. However, some applications are unique and may require work beyond what is proposed in this thesis. For instance, developing equivalents for systems with dynamic loads. The proposed procedure and guidelines in this thesis did not cover model equivalents of dynamic loads. More generally, the application of the proposed guidelines on systems with components that were not investigated in this work need to consider the specific limitation of the proposed guidelines.

6.2. Contributions

The contribution of this thesis is a proposed and validated general procedure and guidelines to develop low frequency model equivalents. The procedure is sequential and is aimed to set the steps required for each stage in the development of the equivalent model. The guidelines were presented in the form of ten principles split based on the four main stages of the general procedure.

The secondary contribution is to explicitly show the impact of inconsistency in developing the equivalent model. These inconsistencies were demonstrated when the principles of the guideline were validated. Using consistent and reliable model equivalents play a key role in ensuring that today's complicated and ever expanding grids are designed and operated reliably.

6.3. Future Work

This thesis was only a piece of the tremendous opportunities available for further work in the topic of dynamic model equivalents. An opportunity exists to develop comprehensive guidelines for the other two types of dynamic model equivalents, namely high frequency model equivalents and wide-band equivalents. Together, the three guidelines will form a much needed comprehensive guideline to develop dynamic model equivalents consistently and reliably.

Bibliography

[1] J. B. Ward. “Equivalent Circuits for Power-Flow Studies”. In: *Transactions of the American Institute of Electrical Engineers* 68.1 (July 1949), pp. 373–382. ISSN: 0096-3860. DOI: 10.1109/T-AIEE.1949.5059947.

[2] W. T. Brown and W. J. Cloues. “Combination Load-Flow and Stability Equivalent for Power System Representation on A-C Network Analyzers [includes discussion]”. In: *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems* 74.3 (Jan. 1955). ISSN: 0097-2460. DOI: 10.1109/AIEEPAS.1955.4499148.

[3] R. Podmore. “Identification of Coherent Generators for Dynamic Equivalents”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-97.4 (July 1978), pp. 1344–1354. ISSN: 0018-9510. DOI: 10.1109/TPAS.1978.354620.

[4] L. Wang et al. “Dynamic reduction of large power systems for stability studies”. In: *IEEE Transactions on Power Systems* 12.2 (May 1997), pp. 889–895. ISSN: 0885-8950. DOI: 10.1109/59.589749.

[5] W. W. Price et al. “Large-scale system testing of a power system dynamic equivalencing program”. In: *IEEE Transactions on Power Systems* 13.3 (Aug. 1998), pp. 768–774. ISSN: 0885-8950. DOI: 10.1109/59.708595.

[6] M. Matar, N. Fernandopulle, and A. Maria. “Dynamic Model Reduction of Large Power Systems Based On Coherency Aggregation Techniques and Black-Box Optimization”. In: *International Conference on Power Systems Transients (IPST)*. July 2013, pp. 1–6.

[7] J. P. Yang, G. H. Cheng, and Z. Xu. “Dynamic Reduction of Large Power System in PSS/E”. In: *2005 IEEE/PES Transmission Distribution Conference Exposition: Asia and Pacific*. Aug. 2005, pp. 1–4. DOI: 10.1109/TDC.2005.1546815.

[8] Arash Vahidnia et al. “Dynamic equivalent state estimation for multi-area power systems with synchronized phasor measurement units”. In: *Electric Power Systems Research* 96 (2013), pp. 170–176. ISSN: 0378-7796. DOI: `http://dx.doi.org/10.1016/j.epsr.2012.11.006`. URL: `http://www.sciencedirect.com/science/article/pii/S0378779612003392`.

[9] P. Ju, L. Q. Ni, and F. Wu. “Dynamic equivalents of power systems with online measurements. Part 1: Theory”. In: *IEE Proceedings - Generation, Transmission and Distribution* 151.2 (Mar. 2004), pp. 175–178. ISSN: 1350-2360. DOI: `10.1049/ip-gtd:20040095`.

[10] P. Ju et al. “Dynamic equivalents of power systems with online measurements Part 2: Applications”. In: *IEE Proceedings - Generation, Transmission and Distribution* 151.2 (Mar. 2004), pp. 179–182. ISSN: 1350-2360. DOI: `10.1049/ip-gtd:20040075`.

[11] Ning Zhou et al. “Calibration of reduced dynamic models of power systems using phasor measurement unit (PMU) Data”. In: *North American Power Symposium (NAPS), 2011*. Aug. 2011, pp. 1–7. DOI: `10.1109/NAPS.2011.6024873`.

[12] A. Chakrabortty, J. H. Chow, and A. Salazar. “A Measurement-Based Framework for Dynamic Equivalencing of Large Power Systems Using Wide-Area Phasor Measurements”. In: *IEEE Transactions on Smart Grid* 2.1 (Mar. 2011), pp. 68–81. ISSN: 1949-3053. DOI: `10.1109/TSG.2010.2093586`.

[13] S. Wang et al. “Measurement-based coherency identification and aggregation for power systems”. In: *2012 IEEE Power and Energy Society General Meeting*. July 2012, pp. 1–7. DOI: `10.1109/PESGM.2012.6345407`.

[14] Joe H. Chow, ed. *Power System Coherency and Model Reduction*. 1st ed. Vol. 94. Power Electronics and Power Systems. Springer-Verlag New York, 2013. ISBN: 978-1-4614-1802-3. DOI: `10.1007/978-1-4614-1803-0`.

[15] M. Abdel-Rahman, A. Semlyen, and M. Reza Iravani. "Two-layer network equivalent for electromagnetic transients". In: *IEEE Transactions on Power Delivery* 18.4 (Oct. 2003), pp. 1328–1335. ISSN: 0885-8977. DOI: 10.1109/TPWRD.2003.817749.

[16] M. Matar and R. Iravani. "A Modified Multiport Two-Layer Network Equivalent for the Analysis of Electromagnetic Transients". In: *IEEE Transactions on Power Delivery* 25.1 (Jan. 2010), pp. 434–441. ISSN: 0885-8977. DOI: 10.1109/TPWRD.2009.2034785.

[17] X. Lin, A. M. Gole, and M. Yu. "A Wide-Band Multi-Port System Equivalent for Real-Time Digital Power System Simulators". In: *IEEE Transactions on Power Systems* 24.1 (Feb. 2009), pp. 237–249. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2008.2007000.

[18] Y. Liang et al. "Improved Coherency-Based Wide-Band Equivalents for Real-Time Digital Simulators". In: *IEEE Transactions on Power Systems* 26.3 (Aug. 2011), pp. 1410–1417. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2010.2085456.

[19] Y. Zhang et al. "Development and Analysis of Applicability of a Hybrid Transient Simulation Platform Combining TSA and EMT Elements". In: *IEEE Transactions on Power Systems* 28.1 (Feb. 2013), pp. 357–366. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2012.2196450.

[20] Q. Huang and V. Vittal. "Application of Electromagnetic Transient-Transient Stability Hybrid Simulation to FIDVR Study". In: *IEEE Transactions on Power Systems* 31.4 (July 2016), pp. 2634–2646. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2015.2479588.

[21] U. D. Annakkage et al. "Dynamic System Equivalents: A Survey of Available Techniques". In: *IEEE Transactions on Power Delivery* 27.1 (Jan. 2012), pp. 411–420. ISSN: 0885-8977. DOI: 10.1109/TPWRD.2011.2167351.

[22] *Alberta Climate Leadership Plan*. URL: https://www.alberta.ca/climate-leadership-plan.aspx.

[23] R. Nath, S. S. Lamba, and K. s. P. Rao. “Coherency Based System Decomposition into Study and External Areas Using Weak Coupling”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-104.6 (June 1985), pp. 1443–1449. ISSN: 0018-9510. DOI: 10.1109/TPAS.1985.319158.

[24] S. Nabavi and A. Chakrabortty. “Topology identification for dynamic equivalent models of large power system networks”. In: *2013 American Control Conference*. June 2013, pp. 1138–1143. DOI: 10.1109/ACC.2013.6579989.

[25] V. Jalili-Marandi et al. “Interfacing Techniques for Transient Stability and Electromagnetic Transient Programs IEEE Task Force on Interfacing Techniques for Simulation Tools”. In: *IEEE Transactions on Power Delivery* 24.4 (Oct. 2009), pp. 2385–2395. ISSN: 0885-8977. DOI: 10.1109/TPWRD.2008.2002889.

[26] B. Gustavsen and A. Semlyen. “Rational approximation of frequency domain responses by vector fitting”. In: *IEEE Transactions on Power Delivery* 14.3 (July 1999), pp. 1052–1061. ISSN: 0885-8977. DOI: 10.1109/61.772353.

[27] X. Nie, Y. Chen, and V. Dinavahi. “Real-Time Transient Simulation Based on a Robust Two-Layer Network Equivalent”. In: *IEEE Transactions on Power Systems* 22.4 (Nov. 2007), pp. 1771–1781. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2007.907963.

[28] M. Matar and R. Iravani. “FPGA Implementation of a Modified Two-Layer Network Equivalent for Real-Time Simulation of Electromagnetic Transients”. In: *International Conference on Power Systems IPST Transients*. Ed. by editor. 2009.

[29] Jun-Hee Hong and Jong-Keun Park. “A time-domain approach to transmission network equivalents via Prony analysis for electromagnetic transients analysis”. In: *IEEE Transactions on Power Systems* 10.4 (Nov. 1995), pp. 1789–1797. ISSN: 0885-8950. DOI: 10.1109/59.476042.

[30] Wallace do Couto Boaventura et al. “Robust sparse network equivalent for large systems: part I-methodology”. In: *IEEE Transactions on Power Systems* 19.1 (Feb. 2004), pp. 157–163. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2003.818603.

[31] W. do Couto Boaventura et al. “Robust sparse network equivalent for large systems: Part II-performance evaluation”. In: *IEEE Transactions on Power Systems* 19.1 (Feb. 2004), pp. 293–299. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2003.818592.

[32] A. Abur and H. Singh. “Time domain modeling of external systems for electromagnetic transients programs”. In: *IEEE Transactions on Power Systems* 8.2 (May 1993), pp. 671–679. ISSN: 0885-8950. DOI: 10.1109/59.260813.

[33] H. Singh and A. Abur. “Multi-port equivalencing of external systems for simulation of switching transients”. In: *IEEE Transactions on Power Delivery* 10.1 (Jan. 1995), pp. 374–382. ISSN: 0885-8977. DOI: 10.1109/61.368376.

[34] X. Guillaud et al. “Applications of Real-Time Simulation Technologies in Power and Energy Systems”. In: *IEEE Power and Energy Technology Systems Journal* 2.3 (Sept. 2015), pp. 103–115. DOI: 10.1109/JPETS.2015.2445296.

[35] Q. Huang and V. Vittal. “OpenHybridSim: An open source tool for EMT and phasor domain hybrid simulation”. In: *2016 IEEE Power and Energy Society General Meeting (PESGM)*. July 2016, pp. 1–5. DOI: 10.1109/PESGM.2016.7741233.

[36] *Improved Dynamic Equivalencing Software*. Tech. rep. TR-105919, EPRI Project: 2447-02. EPRI, 1995.

[37] J. H. Chow et al. “Inertial and slow coherency aggregation algorithms for power system dynamic model reduction”. In: *IEEE Transactions on Power Systems* 10.2 (May 1995), pp. 680–685. ISSN: 0885-8950. DOI: 10.1109/59.387903.

[38] A. J. Germond and R. Podmore. “Dynamic Aggregation of Generating Unit Models”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-97.4 (July 1978), pp. 1060–1069. ISSN: 0018-9510. DOI: `10.1109/TPAS.1978.354585`.

[39] S. M. Benchluch and J. H. Chow. “A trajectory sensitivity method for the identification of nonlinear excitation system models”. In: *IEEE Transactions on Energy Conversion* 8.2 (June 1993), pp. 159–164. ISSN: 0885-8969. DOI: `10.1109/60.222699`.

[40] M. L. Ourari, L. A. Dessaint, and Van-Que Do. “Dynamic equivalent modeling of large power systems using structure preservation technique”. In: *IEEE Transactions on Power Systems* 21.3 (Aug. 2006), pp. 1284–1295. ISSN: 0885-8950. DOI: `10.1109/TPWRS.2006.876699`.

[41] T. E. Dy Liacco, S. C. Savulescu, and K. A. Ramarao. “An On-Line Topological Equivalent of a Power System”. In: *IEEE Transactions on Power Apparatus and Systems* PAS-97.5 (Sept. 1978), pp. 1550–1563. ISSN: 0018-9510. DOI: `10.1109/TPAS.1978.354647`.

[42] Edwin B. Cano, ed. *New England 39 Bus Test System*. 2012. URL: `http://elektrisidadpilipinas.blogspot.com/2012/11/new-england-39-bus-test-system.html`.

[43] *Short-Circuit Modeling and System Strength*. Tech. rep. North American Electric Reliablity Corporation, 2018.

Appendix A

IEEE 39-Bus System Model Data

Table A.1: Transformer Data

From	To	R (pu)	X (pu)	Ratio (pu)
2	30	0	0.0181	1.025
6	31	0	0.025	1.07
10	32	0	0.02	1.07
11	12	0.0016	0.0435	1.006
12	13	0.0016	0.0435	1.006
19	20	0.0007	0.0138	1.06
19	33	0.0007	0.0142	1.07
20	34	0.0009	0.018	1.009
22	35	0	0.0143	1.025
23	36	0.0005	0.0272	1
25	37	0.0006	0.0232	1.025
29	38	0.0008	0.0156	1.025

Table A.2: Machine Data

Bus	PGen (MW)	QGen (Mvar)	R Source (pu)	X Source (pu)
30	250	146.2705	0.00014	0.0132
31	521.2275	197.6481	0.0027	0.0369
32	650	205.4528	0.000386	0.032
33	632	110.0967	0.000222	0.031
34	508	165.8515	0.00014	0.0568
35	650	212.6253	0.00615	0.0236
36	560	101.2961	0.000268	0.034
37	540	0.5233	0.000686	0.03
38	830	22.9159	0.0003	0.0314
39	1000	88.0607	0.0001	0.0006

Table A.3: Bus Data

Bus Number	Voltage (pu)	Angle (deg)
1	1.0474	-31.7
2	1.0487	-29.02
3	1.0301	-31.87
4	1.0038	-32.87
5	1.0054	-31.88
6	1.0079	-31.21
7	0.9972	-33.39
8	0.9962	-33.88
9	1.0283	-33.59
10	1.0171	-28.69
11	1.0131	-29.55
12	0.994	-29.51
13	1.0138	-29.36
14	1.0114	-30.92
15	1.0152	-31.01
16	1.0317	-29.46
17	1.0335	-30.57
18	1.0309	-31.49
19	1.0498	-24.29
20	0.9912	-25.28
21	1.0317	-27.05
22	1.0498	-22.6
23	1.0448	-22.8
24	1.0372	-29.34
25	1.0575	-27.63
26	1.052	-28.8
27	1.0377	-30.76
28	1.0501	-25.28
29	1.0499	-22.52
30	1.0475	-26.6
31	0.982	-23.26
32	0.9831	-20.7
33	0.9972	-19.08
34	1.0123	-20.09
35	1.0493	-17.64
36	1.0635	-14.95
37	1.0278	-20.85
38	1.0265	-15.46
39	1.03	-33.32

Table A.4: Branch Data

From	To	R (pu)	X (pu)	B (pu)
1	2	0.0035	0.0411	0.6987
1	39	0.001	0.025	0.75
2	3	0.0013	0.0151	0.2572
2	25	0.007	0.0086	0.146
3	4	0.0013	0.0213	0.2214
3	18	0.0011	0.0133	0.2138
4	5	0.0008	0.0128	0.1342
4	14	0.0008	0.0129	0.1382
5	6	0.0002	0.0026	0.0434
5	8	0.0008	0.0112	0.1476
6	7	0.0006	0.0092	0.113
6	11	0.0007	0.0082	0.1389
7	8	0.0004	0.0046	0.078
8	9	0.0023	0.0363	0.3804
9	39	0.001	0.025	1.2
10	11	0.0004	0.0043	0.0729
10	13	0.0004	0.0043	0.0729
13	14	0.0009	0.0101	0.1723
14	15	0.0018	0.0217	0.366
15	16	0.0009	0.0094	0.171
16	17	0.0007	0.0089	0.1342
16	19	0.0016	0.0195	0.304
16	21	0.0008	0.0135	0.2548
16	24	0.0003	0.0059	0.068
17	18	0.0007	0.0082	0.1319
17	27	0.0013	0.0173	0.3216
21	22	0.0008	0.014	0.2565
22	23	0.0006	0.0096	0.1846
23	24	0.0022	0.035	0.361
25	26	0.0032	0.0323	0.513
26	27	0.0014	0.0147	0.2396
26	28	0.0043	0.0474	0.7802
26	29	0.0057	0.0625	1.029
28	29	0.0014	0.0151	0.249

Table A.5: Load Data

Bus	Pload (MW)	Qload (Mvar)
3	322	2.4
4	500	184
7	233.8	84
8	522	176
12	7.5	88
15	320	153
16	329.4	32.3
18	158	30
20	628	103
21	274	115
23	247.5	84.6
24	308.6	-92.2
25	224	47.2
26	139	17
27	281	75.5
28	206	27.6
29	283.5	26.9
31	9.2	4.6
39	1104	250

Table A.6: Generator Dynamic Models - All GENROU

Bus	Xd	Xq	X'd	X'q	X"d = X"q	Xl	S(1.0)	S(1.2)
30	0.1	0.069	0.031	0.018	0.0132	0.0125	0	0
31	0.295	0.282	0.0697	0.17	0.0369	0.035	0	0
32	0.2495	0.237	0.0531	0.0876	0.032	0.0304	0	0
33	0.262	0.258	0.0436	0.166	0.031	0.0295	0	0
34	0.67	0.62	0.132	0.166	0.0568	0.054	0	0
35	0.254	0.241	0.05	0.0814	0.0236	0.0224	0	0
36	0.295	0.292	0.049	0.186	0.034	0.0322	0	0
37	0.29	0.28	0.057	0.0911	0.03	0.028	0	0
38	0.2106	0.205	0.057	0.0587	0.0314	0.0298	0	0
39	0.2	0.019	0.006	0.008	0.0006	0.003	0	0

Table A.7: Exciter Dynamic Models - ALL IEEET1

Bus	TR (sec)	KA	TA (sec)	VRMAX or zero	VRMIN	KE or zero	TE (>0)(sec)
30	0	5	0.06	5	-5	-0.0485	0.25
31	0	6.2	0.05	5	-5	-0.633	0.405
32	0	5	0.06	5	-5	-0.0198	0.5
33	0	5	0.06	5	-5	-0.0525	0.5
34	0	40	0.02	10	-10	1	0.785
35	0	5	0.02	5	-5	-0.0419	0.471
36	0	40	0.02	6.5	-6.5	1	0.73
37	0	5	0.02	5	-5	-0.047	0.528
38	0	40	0.02	10.5	-10.5	1	1.4

Bus	KF	TF(>0)(sec)	Switch=0	E1	SE(E1)	E2	SE(E2)
30	0.04	1	0	1.7	0.5	3	2
31	0.057	0.5	0	3	0.66	4	0.88
32	0.08	1	0	3	0.13	4	0.34
33	0.08	1	0	3	0.08	4	0.31
34	0.03	1	0	3	0.03	4	0.91
35	0.0754	1.246	0	3	0.08	4	0.25
36	0.03	1	0	3	0.03	4	0.74
37	0.0854	1.26	0	3	0.09	4	0.28
38	0.03	1	0	3	0.03	4	0.85

Table A.8: HVDC Dynamic Model - CDC6T

Parameter	Converter	Inverter
ALFDY (degrees)	5	5
GAMDY (degrees)	18	18
TVDC (sec)	0.02	0.02
TIDC (sec)	0.02	0.02
VBLOCK (pu)	0.6	0.6
VUNBL (pu)	0.65	0.65
TBLOCK (sec)	0.1	0.1
VBYPAS (kV)	0	0
VUNBY (pu)	0.9	0.9
TBYPAS (sec)	0.1	0.1
RSVOLT (kV)	0	0
RSCUR (amp)	0	0
VRAMP (pu/sec)	5	5
CRAMP (pu/sec)	5	5
C0 (amp)	200	200
V1 (kV)	150	150
C1 (amp)	800	800
V2 (kV)	200	200
C2 (amp)	1500	1500
V3 (kV)	225	225
C3 (amp)	2000	2000
TCMODE (sec)	0	0
VDEBLK (pu)	0	0
TDEBLK (sec)	0	0
TREBLK (sec)	0	0
VINBLK (pu)	0	0
TCOMB (sec)	0	0
VACBYP (pu)	0.8	0.8
TDEBYP (sec)	0	0
TINBLK (sec)	0	0
TINBYP (sec)	0	0
TVRDC (sec)	0.02	0.02

Table A.9: Wind Generators Dynamic Models - WT4G1

Parameter	Bus 101	Bus 102	Bus 103	Bus 104
TIQCmd, Converter time constant for IQcmd, second	0.01	0.01	0.01	0.01
TIpCmd, Converter time constant for IPcmd, second	0.01	0.01	0.01	0.01
VLVPL1 - Low Voltage power Logic (LVPL), voltage 1 (pu)	0.8	0.8	0.8	0.8
VLVPL2 - LVPL voltage 2 (pu)	0.81	0.81	0.81	0.81
GLVPL - LVPL gain	1.35	1.35	1.35	1.35
High Voltage reactive Current (HVRC) logic,voltage (pu)	1.35	1.35	1.35	1.35
CURHVRCR - HVRC logic, current (pu)	1.35	1.35	1.35	1.35
RIp_LVPL, Rate of active current change	5	5	5	5
T_LVPL, Voltage sensor for LVPL, second	0.02	0.02	0.02	0.02

Table A.10: Wind Generator Dynamic Models - WT4E1

Parameter	Bus 101	Bus 102	Bus 103	Bus 104
Tfv - V-regulator filter	0.01	0.01	0.01	0.01
Kpv - V-regulator proportional gain	0.01	0.01	0.01	0.01
Kiv - V-regulator integrator gain	0	0	0	0
Kpp - T-regulator proportional gain	5	5	5	5
Kip - T-regulator integrator gain	15	15	15	15
Kf - Rate feedback gain	0	0	0	0
Tf - Rate feedback time constant	0.08	0.08	0.08	0.08
QMX - V-regulator max limit	0.69	0.69	0.69	0.69
QMN - V-regulator min limit	-0.69	-0.69	-0.69	-0.69
IPMAX - Max active current limit	1.2	1.2	1.2	1.2
TRV - V-sensor	0.02	0.02	0.02	0.02
dPMX - Max limit in power PI controller (pu)	0.5	0.5	0.5	0.5
dPMN - Min limit in power PI controller (pu)	-0.5	-0.5	-0.5	-0.5
T_POWER - Power filter time constant	0.05	0.05	0.05	0.05
KQi - MVAR/Volt gain	2	2	2	2
VMINCL	0.8	0.8	0.8	0.8
VMAXCL	1.2	1.2	1.2	1.2
KVi - Volt/MVAR gain	200	200	200	200
Tv - Lag time constant in WindVar controller	40	40	40	40
Tp - Pelec filter in fast PF controller	0.05	0.05	0.05	0.05
ImaxTD - Converter current limit	1.35	1.35	1.35	1.35
Iphl - Hard active current limit	1.35	1.35	1.35	1.35
Iqhl - Hard reactive current limit	1.35	1.35	1.35	1.35

Appendix B

Permission to Use Slow Coherency Toolbox

From: Chow, Joe H
Sent: Tuesday, May 9, 2017 7:35 AM
To: sameh2001ye@gmail.com
Subject: RE: Permission to use--Power System Coherency Toolbox for the PST

You are welcome to use the source code, which is the purpose for a user without second guessing algorithms published in our papers.

I would appreciate receiving your code. I won't use it, but want to see how it is done.

Regards,

Joe H. Chow
Professor, Electrical, Computer, and Systems Engineering
Campus Director, NSF/DOE CURENT ERC
Rensselaer Polytechnic Institute, Troy, New York
voice: 518-276-6374 fax: 518-276-8715 email: chowj@rpi.edu
web: http://www.ecse.rpi.edu/homepages/chowj

From: sameh2001ye@gmail.com [mailto:sameh2001ye@gmail.com]
Sent: Monday, May 08, 2017 10:32 PM
To: Chow, Joe H <chowj@rpi.edu>
Subject: Permission to use--Power System Coherency Toolbox for the PST

Hi Prof. Chow:

First, I'd like express my gratitude. I have been using your book *"Power System Coherency and Model Reduction"* and I would like to thank you for such a great industry reference in this topic.

I'd like your permission to use the Power System Coherency Toolbox in my thesis. I plan to use it in several case studies on Alberta's system with modifications. The modifications will include porting the whole code to Python and building interface modules to PSS/E.

I will properly add references to your book and papers and indicate that this is the origin of the code. Other than your permission, do I need to do anything else to be able to use this code in my thesis?

Best Regards,
Sameh

Appendix C

Python Code for Tools to Develop Low Frequency Model Equivalents

C.1. Coherency Toolbox

Description of main modules:

- *coh_main.py*: main interface to the coherency identification module of coherency toolbox. Sets up main parameters and calls *coherent.py*
- *coherent.py*: this is the main module, it generates the $MK^{-}1$ matrix. Refer to [14] for more details.
- *V_slow.py*: this function calculates the slow modes of the $MK^{-}1$ matrix. Refer to [14] for more details.
- *coh_map.py*: this function calculates the coherency map based on the slow modes calculated by *V_slow.py*
- *ex_group.py*: this function computes and outputs the coherent groups
- *agg_main.py*: main interface to the aggregation module of the coherency toolbox
- *i_agg.py*: inertial aggregation module

coh_main.py

```python
def coh_main(bus,line,machine,ns,tol):
    import numpy as np
    import scipy.io as sio
    import cmath
    import math
    import glb
    import sys
    from coherent import coherent

    glb.mac_con = machine

    jay = cmath.sqrt(-1)

    glb.basrad = 2*math.pi*60

    glb.basmva = 100
    glb.syn_ref = 1


    area,nmach_a,areal,nmach_al,eig_all = coherent(bus,line,2,ns,tol)

    areabusnumbers = np.zeros((area.shape[0],area.shape[1]),dtype=int)
    for i in range(area.shape[0]):
        for j in range(area.shape[1]):
            if area[i,j]!=0:
                areabusnumbers[i,j] = glb.mac_con[area[i,j]-1,1]

    return area, nmach_a,areabusnumbers,eig_all
```

coherent.py

```
def coherent(bus,line,meth,ns,tol):
    from svm_em import svm_em
    from V_slow import V_slow
    from coh_map import coh_map
    from ex_group import ex_group
    import glb,math,cmath
    import numpy as np
    import scipy.io as sio

    MK = svm_em(bus,line,1)

    eig_s,V_s,eig_all = V_slow(MK,ns)

    grouping,c_map = coh_map(V_s,tol)

    area,nmach_a,areal,nmach_al = ex_group(grouping,ns)

    return area,nmach_a,areal,nmach_al,eig_all
```

V_slow.py

```
import math#,cmath
def V_slow(MK,n_s):
    import numpy as np
    lamda, eig_vec = np.linalg.eig(MK)

    k = np.argsort(np.abs(lamda))
    lamda2 = lamda[k]
    eig_s = lamda2[0:n_s]

    V_s = eig_vec[:, k[0:n_s]]

    return eig_s, V_s,lamda2
```

coh_map.py

```python
def coh_map(V_s,tol):
    import glb
    import numpy as np
    t1 = V_s.shape
    n = t1[0]
    narea = t1[1]

    for i in range(narea):
        V_s[:,i] = V_s[:,i]/np.linalg.norm(V_s[:,i])

    for i in range(n):
        V_s[i,:] = V_s[i,:]/np.linalg.norm(V_s[i,:])

    c_map = V_s.dot(V_s.T)
    grouping = c_map - tol*np.ones((n,n))
    c_map = np.maximum(grouping, np.zeros((n,n)))


    return grouping, c_map
```

ex_group.py

```python
def ex_group(grouping,ns):
    import glb
    import numpy as np
    from rem_area import rem_area
    from diagb import diagb
    from add_area import add_area
    from offdb import offdb
    t1 = grouping.shape
    nmach = t1[0]

    n_coh = np.ones((nmach,1),dtype=np.int)
    coh_area = np.zeros((nmach,nmach),dtype=np.int)
    coh_area[:,0] = np.arange(1,nmach+1)
    coh_label = np.arange(1,nmach+1,dtype=np.int)
    coh_label = np.resize(coh_label, (nmach, 1))

    for i in range(nmach):
        for j in range(i+1,nmach):
            if grouping[i,j] >0:
                if coh_label[i] != coh_label[j]:
                    k1 = coh_label[i]-1
                    k2 = coh_label[j]-1
                    k_small = min(k1,k2)
                    k_large = max(k1,k2)
                    coh_label[coh_area[k_large,0:n_coh[k_large].item()]-1]
                            =(k_small+1)*np.ones((n_coh[k_large].item(),1)
                              ,dtype=np.int)
                    t1 = coh_area[k_small,0:n_coh[k_small].item()]
                    t2 = coh_area[k_large,0:n_coh[k_large].item()]
                    temp = np.concatenate((t1,t2),axis=1).T

                    n_coh[k_small] = n_coh[k_small] + n_coh[k_large]
                    coh_area[k_small,0:n_coh[k_small].item()] =
                     ↪  np.resize(temp,(temp.shape[0],))
                    coh_area[k_large, 0:n_coh[k_large].item()] =
                     ↪  np.zeros((1,n_coh[k_large].item()))
                    n_coh[k_large] = 0

    select = np.array([])

    for i in range(nmach):
        if n_coh[i] != 0:
            select = np.append(select, i)

    t1 = select.shape
    nr = t1[0]
    arear = coh_area[select.astype(int),:]
    nmach_ar = n_coh[select.astype(int)]

    for i in range(nr):
        temp = arear[i,1:nmach_ar[i].item()]
        temp = np.sort(temp)
        arear[i,1:nmach_ar[i].item()] = temp

    n = -1

    area = []
    nmach_a = []

    for i in range(nr):
        if nmach_ar[i] == 1:#single machine area, no splitting
            #n = n+1
            area.append(np.array([arear[i,0]]))
            nmach_a.append(nmach_ar[i])
        else:
            temp = arear[i,0:nmach_ar[i].item()]
            t1 = (temp-1).T
            map = grouping[t1,:]
            map = map[:,t1]

            row_sum = np.sum(map-np.diag(np.diag(map)),axis=0)
            if np.amin(row_sum) > 0:
                area.append(temp)
                nmach_a.append(nmach_ar[i])
            else:
                iarea = 1
                nmach = nmach_ar[i]
                narea_tem = nmach
                area_tem = np.arange(1,nmach+1,dtype=np.int)
                area_tem = np.resize(area_tem,(iarea,nmach[0]))
                cost_old = 0
                term = 0

                while term ==0:
                    if iarea ==1:
                        map_diag = map
                    else:
                        map_diag = diagb(map,area_tem,narea_tem)
                    y = np.sum(map_diag,axis=0)
                    for k in range(iarea):
                        jk = area_tem[k,0:narea_tem[k]]
                        t1 = jk-1
                        y[t1.astype(int)] = y[t1.astype(int)]/narea_tem[k]

                    next_mac = np.argmin(y)+1
                    t1 = rem_area(area_tem,narea_tem,next_mac)
                    areay = t1[0]
                    nareay = t1[1]

                    cost = 0

                    for k in range(iarea+1):
                        t1 = add_area(areay,nareay,next_mac,k)
                        areax = t1[0]
                        nareax = t1[1]
                        map_off = offdb(map,areax,nareax)
                        new_cost = np.sum(np.sum(map_off))
                        if new_cost < cost:
                            cost = new_cost
                            area_tem = areax
                            narea_tem = nareax

                    t1 = narea_tem.shape
                    iarea = t1[0]
                    if cost < cost_old:
                        cost_old = cost
                    else:
                        term = 1
                        t1 = narea_tem.shape
                        n_add = t1[0]#double check
                        for k in range(n_add):
                            #n = n+1
                            t1 = narea_tem[k]
                            area.append(
                            temp[area_tem[k,0:t1].astype(int)-1])
                            nmach_a.append(narea_tem[k])

    area = np.asarray(area)
    nmach_a = np.asarray(nmach_a,dtype=np.int)
    #constract array from area and nmach_a
    maxx = np.amax(nmach_a)
    area2 = np.zeros((nmach_a.shape[0],maxx),dtype=np.int)
    for i in range(area.shape[0]):
        if area[i].shape[0] < maxx:
            area2[i] = np.append(area[i],
             ↪  np.zeros(maxx-area[i].shape[0],dtype=np.float))
        else:
            area2[i] = area[i]
    area = area2
    t1 = nmach_a.shape


    return area,nmach_a,arear,nmach_ar
```

agg_main.py

```
def agg_main(bus,line,mac_con, area, narea):
    # this function is used in coh_run to run aggregation of
    # reduced groups calculated simulataneously

    import cmath
    import math
    from reduce import reduce
    from i_agg import i_agg
    import numpy as np
    import scipy.io as sio
    import glb
    from svm_em import svm_em

    jay=cmath.sqrt(- 1)


    glb.mac_con = mac_con

    basrad=2 * math.pi * 60
    basmva=100

    rbus,rline,rmac=i_agg(bus,line,area,narea,100)

    return rbus, rline,rmac
```

i_agg.py

```
def i_agg(bus,line,area,nmach_a,basemva):
    import glb
    import numpy as np
    import cmath
    import math
    from line_pq import line_pq
    jay = cmath.sqrt(-1)
    from eqgen import eqgen

    n_bus = np.array(bus)
    n_line = np.array(line)
    n_mac = glb.mac_con
    add_bus = np.zeros((1,10),dtype=float)
    t1 = n_mac.shape
    if t1[1] <=21:
        t2 = np.ones((t1[0], 23))
        t2[:, 0:t1[1]] = n_mac
        n_mac = t2

    nbus = n_bus.shape[0]
    bus_int = np.zeros(((np.amax(n_bus[:,0].astype(int))),1),dtype=int)

    for i in range(nbus):
        bus_int[n_bus[i,0].astype(int)-1] = i

    tot_mac = n_mac.shape[0]
    mac_int = np.zeros(((np.amax(n_mac[:,0].astype(int))),1),dtype=int)
    for i in range(tot_mac):
        mac_int[n_mac[i,0].astype(int)-1] = i

    bus_vol = np.array(n_bus[:,1])
    bus_ang = np.array(n_bus[:,2])
    num_area = area.shape[0]

    nmac_con = []

    totalp = 0
    totalq = 0

    for r in range(num_area):
        if nmach_a[r]==1:
            nmac_con.append(n_mac[mac_int[area[r,0]-1],:])
            n_mac[mac_int[area[r,0]-1],0] = 0

        else:
            num_mach = nmach_a[r]
            com_bus = np.amax(n_bus[:,0])+1
            mac_list = mac_int[area[r,0:nmach_a[r]]-1].T
            nlines = n_line.shape[0]
            bus_list = bus_int[glb.mac_con[mac_list,1].astype(int)[0]-1]
            bus_type = 2
            for ii in range(num_mach):
                if n_bus[bus_list[ii,0],9]==1:
                    bus_type = 1

            m = glb.mac_con[mac_list,2]*glb.mac_con[mac_list,15]/basemva
            ma = np.sum(m)
            Pg = n_bus[bus_list,3]*glb.mac_con[mac_list,21].T
            Qg = n_bus[bus_list,4]*glb.mac_con[mac_list,22].T
            vb = n_bus[bus_list,1]
            angb = n_bus[bus_list,2]*math.pi/180
            vbx = vb*(np.cos(angb)+jay*np.sin(angb))
            xdp = np.zeros((num_mach,1))

            for i in range(num_mach):
                if np.count_nonzero(glb.mac_con[mac_list,7])!=0:
                    xdp[i] = basemva*glb.mac_con[mac_list[0,i],7]
                            /glb.mac_con[mac_list[0,i],2]
                else:
                    xdp[i] = basemva*glb.mac_con[mac_list[0,i],6]
                            /glb.mac_con[mac_list[0,i],2]

            int_cur = ((Pg+jay*Qg)/vbx).conjugate()
            int_vol = vbx + jay*xdp*int_cur

            mag_cbus = np.dot(np.absolute(int_vol).T,m.T/ma)
            ang_cbus = np.dot(np.angle(int_vol).T,m.T/ma)

            vg = np.absolute(int_vol)
            delta = np.angle(int_vol)
            phia = ang_cbus*np.ones(angb.shape)-delta
            tapa = mag_cbus/vg
            xdpa = xdp
            v1 = np.dot(mag_cbus*(np.cos(ang_cbus)+jay*np.sin(ang_cbus)),
                np.ones((vg.shape[1],vg.shape[0]),dtype=float))
            v2 = vb*(np.cos(angb)+jay*np.sin(angb))
            s1,s2 = line_pq(v1, v2, np.zeros((vg.shape[0],vg.shape[1])),
             ↪  xdpa,
                    np.zeros((vg.shape[0],vg.shape[1])),tapa, phia * 180 /
                     ↪  math.pi)


            add_bus[0,0] = com_bus
            add_bus[0,1] = mag_cbus
            add_bus[0,2] = ang_cbus*180/math.pi
            add_bus[0,3:8] = np.zeros((1,5))
            add_bus[0,9] = 3
            n_bus = np.append(n_bus,add_bus,axis=0)


            for j in range(num_mach):
                n_bus[bus_list[j],5:7]=
                 ↪  n_bus[bus_list[j],5:7]-np.array((Pg[j]+s2[j].real,
                                        Qg[j]+s2[j].imag),dtype=float).T

            r1 = num_mach
            temp = np.zeros((r1+1,n_line.shape[1]))


            temp[0:r1,0] = np.dot(
                            np.ones((bus_list.shape[0],bus_list.shape[1]))
                            ,com_bus).T
            temp[0:r1,1] = n_bus[bus_list,0].T
            temp[0:r1,3] = xdpa.T
            temp[0:r1,5] = tapa.T
            temp[0:r1,6] = (phia*180/math.pi).T

            xdeq = 1/np.sum(np.ones((num_mach,1))/xdp)

            term_bus = np.amax(n_bus[:,0])+1
            com_vol = mag_cbus*(np.cos(ang_cbus)+jay*np.sin(ang_cbus))
            new_cur =
             ↪  (np.dot(np.sum(s1),np.linalg.inv(com_vol))).conjugate()
            term_vol = com_vol -jay*np.dot(xdeq,new_cur)


            add_bus[0, 0] = term_bus
            add_bus[0, 1] = np.absolute(term_vol)
            add_bus[0, 2] = np.angle(term_vol) * 180 / math.pi
            add_bus[0, 3] = (term_vol * new_cur.conjugate()).real
            add_bus[0, 4] = (term_vol * new_cur.conjugate()).imag

            add_bus[0, 5:8] = np.zeros((1, 3))
            add_bus[0, 9] = bus_type
            n_bus = np.append(n_bus,add_bus,axis=0)

            temp[r1, 0] = com_bus
            temp[r1, 1] = term_bus
            temp[r1, 2] = 0.0
            temp[r1, 3] = -xdeq
            temp[r1, 4:6] = np.zeros((1, 2))

            n_line = np.append(n_line,temp,axis=0)

            agg_mac = eqgen(n_mac,mac_list,basemva,term_bus,area[r,0])

            nmac_con.append(agg_mac)

            n_mac[mac_list,0] = np.zeros((num_mach,1)).T
    machinesretained = 0

    for r in range(num_area):
        if nmach_a[r]>1:

            num_mach = nmach_a[r];    # no. of coherent machines
            mac_list = mac_int[area[r,0:nmach_a[r]]-1].T
            bus_list = bus_int[glb.mac_con[mac_list,1].astype(int)[0]-1]
            Pg = n_bus[bus_list,3]*glb.mac_con[mac_list,21].T
            Qg = n_bus[bus_list,4]*glb.mac_con[mac_list,22].T


            for i in range(bus_list.shape[0]):
                n_bus[bus_list[i, 0], 3] = n_bus[bus_list[i, 0], 3] -
                 ↪  Pg[i]
                n_bus[bus_list[i, 0], 4] = n_bus[bus_list[i, 0], 4] -
                 ↪  Qg[i]

                if (n_bus[bus_list[i, 0], 3] <0.01):
                    n_bus[bus_list[i, 0],9] = 3

    for i in range(tot_mac):
        if n_mac[i,0]!=0:
            nmac_con.append([n_mac[i,:]])
            machinesretained = machinesretained+1

    nmac_con = np.asarray(nmac_con,dtype=float)
    nmac_con = nmac_con[:,0,:]


    return n_bus,n_line,nmac_con
```

C.2. Python Interface to PSS®E

Description of main modules:

- *coh_run_loop.py*: this is the main module that allows producing multiple equivalents based on a list of input parameters such as number of modes to preserve.
- *getpssedata.py*: this module extracts all network data from a given PSS®E case. This includes bus, line, transformer, shunt and machine dynamic model data.
- *savepssedata.py*: this module saves the equivalent model to a PSS®E case ready for the study. It modifies the case to include the aggregated machines and remove original machines that have been included in the aggregate machine models. This module also inserts the new dynamic models of the aggregate machines.

coh_run_loop.py

```
#main program to run coherency equivalency program
# this program runs gets data using psse and then
# runs coherency equivalent program

import scipy.io as sio
import cmath,math
import numpy as np
import sys
from coh_main import coh_main
from getpssedata import getpssedata
from agg_main import agg_main
from savepssedata import savepssedata
from svm_em import svm_em
from V_slow import V_slow
import glb
import warnings
from ReadYMatrix import ReadYMatrix
import pandas as pd
import os.path as path
import sys

from SetupPSSE2 import *
import psspy
from datetime import datetime
from CleanFolder import CleanFolder
import os

t1=datetime.now()




if len(sys.argv)==6:#check that arguements are good
    ns = sys.argv[1]
    ns = int(ns)
    tol = sys.argv[2]
    tol = float(tol)
    savefile = sys.argv[3]
    dynfile = sys.argv[4]
    matfile = sys.argv[5]
else: #use default file names
    if len(sys.argv)==3:
        studyname = sys.argv[1]
        machinestokeep = np.array([int(sys.argv[2])])
    else:
        machinestokeep = np.array([36])


    if len(sys.argv)==4:
        machinestokeep = np.array([int(sys.argv[2]),int(sys.argv[3])])
        studyname = sys.argv[1]

    if len(sys.argv)>=5:
        machinestokeep = np.array([int(sys.argv[i]) for i in
         ↪  range(2,len(sys.argv))])
        studyname = sys.argv[1]


    #for Alberta
    #nslist = np.array([5,10,15,30,45,60,80])
    #tollist = np.array([0.9,0.95,0.99])
    #for IEEE 39-Bus
    nslist = np.array([1, 2, 3, 4, 6])
    tollist = np.array([0.75, 0.87, 0.9, 0.92, 0.97, 0.99])

    iter=1
    folderup = path.abspath(path.join(__file__ ,"../.."))
    if not os.path.exists(folderup+'\\cohcases\\'+'\\'+studyname+'\\'):
        os.makedirs(folderup+'\\cohcases\\'+'\\'+studyname+'\\')
    CleanFolder(folderup+'\\cohcases\\'+'\\'+studyname+'\\','.sav')
    CleanFolder(folderup+'\\cohcases\\'+'\\'+studyname+'\\','.dyr')
    if not os.path.exists(folderup+'\\output\\'+'\\'+studyname+'\\'):
        os.makedirs(folderup+'\\output\\'+'\\'+studyname+'\\')
    fh = open(folderup+'\\output\\'+'\\'+studyname+'\\'+'list.csv', 'w')
    PSSEObject = SetupPSSE()
    temppath = folderup+'\\temp\\'
    PSSEObject.UpdateOutputFile(Outputfile=temppath+studyname)

    NoDCCases = ['retain machine 37 New' , 'retain machine 37 30 and 39 New',
                'retain machine 37 New test2' ,
                'retain machine 37 30 and 39 New test2']
    DCCases = ['retain machine 30 37 39_HVDC bus fault and HVDC All constant New' ,
               'retain machine 35 36_HVDC bus fault and HVDC All constant New',
               'retain machine 30 37 39_HVDC bus fault and HVDC Iden constant New' ,
               'retain machine 35 36_HVDC bus fault and HVDC Iden constant New',
               'retain machine 35 36_HVDC bus fault and HVDC All constant']
    DCWindCases = ['retain machine 30 37 39_HVDC and Wind_bus fault and Both All constant New',
                   'retain machine 30 37 39_HVDC and Wind_bus fault and Both Iden constant New']
    Alberta = ['Alberta test', 'Alberta test2','Alberta HVDC and Wind','Alberta HVDC and Wind wsnd',
                'Alberta HVDC and Wind Small Study Area','Alberta HVDC and Wind Constant HVDC']
    if studyname in NoDCCases:
        savefile = folderup + '\\Full Cases\\IEEE 39-Bus Working Case_2.sav'
        dynfile =  folderup + '\\Full Cases\\IEEE 39-Bus Dynamic Data.dyr'
    elif studyname in DCCases:
        savefile = folderup + '\\Full Cases\\IEEE 39-Bus Working Case_2_HVDC_ConsLoad.sav'
        dynfile =  folderup + '\\Full Cases\\IEEE 39-Bus Dynamic Data_HVDC_ConsLoad.dyr'
    elif studyname in DCWindCases:
        savefile = folderup + '\\Full Cases\\IEEE 39-Bus Working Case_2_HVDC_Wind_ConsLoad.sav'
        dynfile =  folderup + '\\Full Cases\\IEEE 39-Bus Dynamic Data_HVDC_Wind_ConsLoad.dyr'
    elif studyname in Alberta:
        savefile = folderup + '\\Full Cases\\OPBC_2016SP_V33_R2_Latest_WindConstant_HVDCConstant.sav'
        dynfile =  folderup + '\\Full Cases\\OPBC_2016SP_Dynamic_V33_R2_WindConstant_HVDCConstant.dyr'
    glb.recalculateReducedYMatrix = 1

    for ii in range(nslist.shape[0]):
        for jj in range(tollist.shape[0]):
            ns = nslist[ii]
            tol = tollist[jj]
            matfile = folderup+'\\temp\\casedata_'+studyname

            psspy.case(savefile)
            getpssedata(savefile,dynfile,matfile,False)
            psspy.close_powerflow()

            mat = np.load(matfile+'.npz')
            bus = mat['bus']
            line = mat['line']
            machine = mat['Mparameter2']


            glb.GetYMatrixFromPSSE=False

            area,nmach_a,areabusnumbers,eig_allbefore = coh_main(bus,line,machine,ns,tol)
            MK = svm_em(bus,line,1)
            eig_s,V_s,eig_allbefore = V_slow(MK,ns)


            noofareasbeforeaggregation=area.shape[0]

            nomachinesfull = machine.shape[0]

            machinesretained = np.sum(nmach_a)
            for i in range(areabusnumbers.shape[0]):
                for j in range(areabusnumbers.shape[1]):
                    for k in range(machinestokeep.shape[0]):
                        if areabusnumbers[i,j] == machinestokeep[k]:
```

```
                        areabusnumbers[i,j] = 0
                        area[i,j] = 0
                        nmach_a[i] = nmach_a[i]-1
                sort_indices = np.argsort(areabusnumbers[i,:])[::-1]
                areabusnumbers[i,:] = areabusnumbers[i,sort_indices]
                area[i,:] = area[i,sort_indices]


            area = area[:,0:np.amax(nmach_a)]
            areabusnumbers = areabusnumbers[:,0:np.amax(nmach_a)]


            indices = np.argwhere(nmach_a==0)
            nmach_a = np.delete(nmach_a,indices)
            area = np.delete(area,indices,0)
            areabusnumbers = np.delete(areabusnumbers,indices,0)

            area = area.astype(int)
            nmach_a = nmach_a.astype(int)
            machinesretained = machinesretained-np.sum(nmach_a)



            rbus,rline,rmac = agg_main(bus,line,machine,area,nmach_a)
            matfile = folderup+'\\temp\\data_reduced_'+studyname
            np.savez(matfile,bus=rbus,line=rline,mac=rmac)

            glb.mac_con = rmac
            MK = svm_em(rbus,rline,1)
            eig_s,V_s,eig_allafter = V_slow(MK,ns)


            after = np.array([ cmath.sqrt(i)/(2*math.pi) for i in
             ↪  eig_allafter])
            before = np.array([ cmath.sqrt(i)/(2*math.pi) for i in
             ↪  eig_allbefore])

            if noofareasbeforeaggregation<99999:
                psspy.case(savefile)
                savepssedata(savefile,dynfile,[],[],[],True,
                            matfile,iter,folderup+'\\cohcases\\'+
                            '\\'+studyname+'\\',False)
                psspy.close_powerflow()
                df = pd.DataFrame.from_dict({'File':[iter],
                    'n':[ns],
                        'tol':[tol],
                        '#Gen Full':[nomachinesfull],
                         '#Gen Reduced':[rmac.shape[0]],
                        'Modes_Original': [before],
                        'Modes_After': [after]})
                    if iter==1:
                        df.to_csv(fh,mode='w',columns=['File','n', 'tol','#Gen
                         ↪  Full', '#Gen Reduced',
                        'Modes_Original','Modes_After'],index=False)
                    else:
                        df.to_csv(fh,mode='a',columns=['File','n', 'tol','#Gen
                         ↪  Full','#Gen Reduced',
                        'Modes_Original','Modes_After'],header=False
                        ,index=False)
                iter+=1

t2=datetime.now()
dt=t2-t1
print t1
print t2
s=dt.seconds+dt.microseconds*1.0e-6
h=int(s/3600)
m=int(s-h*3600)/60
s=(s-h*3600-m*60)
print 'time: %3i:%3i:%6.2f (hr:mi:ss)'%(h,m,s)
```

getpssedata.py

```
#This program does not convert zero sequence data and they need to be
# done seperately if fault analysis is to be performed.

import os
import sys
import psspy
import redirect
import numpy as np
import scipy.io as sio
import os.path as path
import sys
import SetupPSSE2



def getpssedata(savefile,dynfile,matfile,initiatepsse):

    folderup = path.abspath(path.join(__file__ ,"../../.."))

```

```python

    if initiatepsse==True:
        SetupPSSE2.setup_psse_from_python(output_to_file=True, append=False,CasePath=folderup+'\\temp\\')
        # open a case; or read a .raw file with psspy.read()
        psspy.case(savefile)


    # solve the case
    psspy.fnsl()
    subsystem = -1

    ierr = psspy.bsys(subsystem,0,[ 0.4, 500.],len(areano),areano,0,[],0,[],0,[])
    # read bus data
    ierr, bus1 = psspy.abusint(subsystem, 1, ['NUMBER', 'TYPE'])
    ierr, bus2 = psspy.abusreal(subsystem, 1, ['PU', 'ANGLED'])
    ierr, bus3 = psspy.agenbuscplx(subsystem, 3, ['PQGEN'])
    ierr, bus4 = psspy.alodbuscplx(subsystem, 3, ['TOTALNOM'])
    ierr, bus5 = psspy.afxshuntcplx(subsystem, 3, ['SHUNTACT', 'SHUNTNOM'])
    ierr, bus6= psspy.afxshuntint(subsystem, 3, ['NUMBER'])
    #load switched shunts
    ierr, bus7 = psspy.aswshint(subsystem, 3, ['NUMBER'])
    ierr, bus8 = psspy.aswshcplx(subsystem, 3, ['YSWACT'])


    bus = np.array([np.array(bus1[0]), np.array(bus2[0]), np.array(bus2[1]), np.array(bus3[0]).real, np.array(bus3[0]).imag, np.array(bus4[0]).real, np.array(bus4[0]).imag])

    t1 = np.zeros(bus.shape[1], dtype=float)
    bus = np.vstack((bus,t1))
    bus = np.vstack((bus,t1))
    bus5 = np.array(bus5)
    bus6 = np.array(bus6)
    bus7 = np.array(bus7)
    bus8 = np.array(bus8)



    if (bus6.size>0):
        for i in range(bus6.shape[1]):
            a = np.argwhere((bus[0,:] == bus6[0,i]))
            bus[7,a[0,0]] = bus5[1,i].real
            bus[8,a[0,0]] = bus5[1,i].imag
    if (bus7.size>0):
        for i in range(bus7.shape[1]):
            a = np.argwhere((bus[0,:] == bus7[0,i]))
            bus[7,a[0,0]] = bus[7,a[0,0]]+bus8[0,i].real
            bus[8,a[0,0]] = bus[8,a[0,0]]+bus8[0,i].imag
    # the following is used to test savnw case swich to 0 to disable and use IEEE 39 bus, 2 to use alberta base case
    test = 2
    bus = np.vstack((bus,np.array(bus1[1])))


    #loop and remove bus type 1 generation because PQ Generation pulled above includes type 1 buses.
    for i in range(bus.shape[1]):
        if bus[9,i] ==1:
            bus[3,i] = 0
            bus[4,i] = 0

    #convert to pu on 100 MVA base
    bus[3:9,:] = bus[3:9,:]/100
    bus = bus.T
    # change bus types 1 to 3 for load bus and 3 to 1 for swing bus because the algorithm bus type codes are different uses them opposite to PSSE

    for i in range(bus.shape[0]):
        if bus[i,9]==1:
            bus[i,9] = 3
        elif bus[i,9]==3:
            bus[i,9] = 1
    # load line data
    # ignores magnetizing admittance
    ierr, line1 = psspy.abrnint(subsystem, 1, 1, 3, 1, ['FROMNUMBER', 'TONUMBER'])
    ierr, line2 = psspy.abrncplx(subsystem, 1, 1, 3, 1, ['RX'])
    ierr, line3 = psspy.abrnreal(subsystem, 1, 1, 3, 1, ['CHARGING'])
    ierr, line4 = psspy.atrnint(subsystem, 1, 1, 1, 1, ['FROMNUMBER', 'TONUMBER','WIND1NUMBER'])
    ierr, line5 = psspy.atrnreal(subsystem, 1, 1, 1, 1, ['RATIO', 'ANGLE'])

    line = np.array([np.array(line1[0]),np.array(line1[1]),
                    np.array(line2[0]).real, np.array(line2[0]).imag,
                    np.array(line3[0])])

    t1 = np.zeros(line.shape[1], dtype=float)
    line = np.vstack((line,t1))
    line = np.vstack((line,t1))

    frm = np.array(line4[0])
    to = np.array(line4[1])
    wind1 = np.array(line4[2])
    ratio = np.array(line5[0])
    angle = np.array(line5[1])

    for i in range(frm.shape[0]):
        a = np.argwhere((line[0]==frm[i]) & (line[1]==to[i]))
        line[5,a[0][0]] = ratio[i] if wind1[i]==frm[i] else 1./ratio[i]
        line[6, a[0][0]] = angle[i] if wind1[i]==frm[i] else -1*angle[i]

    line = line.T

    #load machine parameters
    #todo double check machine data

    ierr = psspy.dyre_new([1,1,1,1],dynfile,folderup+'\\temp\\'+
                            matfile+'_conec.flx',folderup+'\\temp\\'+
                            matfile+'_conet.flx',folderup+'\\temp\\'+
                            matfile+'_compile1.bat')

    #uncomment if you want to load a snp file
    #ierr = psspy.rstr('IEEE 39-Bus Dynamic DataWGoverner.snp')

    ierr, Nmach=psspy.amachcount(subsystem, 1)            # get no of
     ↪  machines in the subsystem
    ierr, iMbus = psspy.amachint(subsystem, 1, 'NUMBER') # get machine bus
     ↪  numbers
    ierr, iWind = psspy.amachint(subsystem,1,'WMOD')# to check for wind
     ↪  machines and exclude them for now
    ierr, cMids = psspy.amachchar(subsystem, 1, 'ID')     # get machine IDs
    #get machines base MVA
    ierr, MBASE = psspy.amachreal(subsystem, 1, 'MBASE')
    ierr, PQGenM = psspy.amachcplx(subsystem,1,'PQGEN')
    PQGenM = np.array(PQGenM).T/100
    ierr, mr= psspy.amachcplx(subsystem, 1, 'ZSORCE') #get machine
     ↪  impedances
    ierr, mrs= psspy.amachreal(subsystem, 1, 'XSYNCH') #get machine
     ↪  synchrounous impedance
    ierr, mrt= psspy.amachreal(subsystem, 1, 'XTRANS') #get machine transient impedance
    ierr, mrst= psspy.amachreal(subsystem, 1, 'XSUBTR') #get machine subtransient impedance
    iMbus=iMbus[0]
    cMids=cMids[0]

    Mparameter = np.zeros((Nmach,23),dtype=float)
    Mparameter[:,0] = np.arange(1, Nmach+1, dtype=np.int) #todo read correct machine IDs
    Mparameter[:,1] = iMbus
    Mparameter[:,2] = MBASE[0]
    Mparameter[:,4] = np.array(mr).real
    Mparameter2 = np.zeros((Mparameter.shape[0],Mparameter.shape[1]))
    Mparameter2[:,0] = Mparameter[:,0]
    Mparameter2[:,1] = Mparameter[:,1]
    Mparameter2[:,2] = Mparameter[:,2]
    Mparameter2[:,4] = np.array(mr).real
    Mparameter2[:,5] = np.array(mrs)
    Mparameter2[:,6] = np.array(mrt)
    Mparameter2[:,7] = np.array(mrst)
    for iM in range(0,Nmach):  # iterate through the list of machines
        ibus=iMbus[iM]
        genId=cMids[iM]
        ierr,busN=psspy.notona(ibus)
        iH=0  # resetting the generator parameter value index
        ierr, icon0 = psspy.mdlind(ibus, genId, 'GEN', 'CON') # get initial CON address (index)
        if test !=-1:
            ierr, genMdl = psspy.mdlnam(ibus, genId, 'GEN') # get generator model name
            if ierr ==0:
                genMdl=genMdl.strip()                           # remove blanks
    #   Find absolute index iH in CONS array using relative CON index in the generator model and
    #   previously found starting CON index of the generator model
    #  (here shown only for the three most common models)

    #   Get value from CONS array corresponding to all generator parameters
        #ierr,H = psspy.dsrval('CON', iH)
        #extract parameters, this changes based on number of generator parameters, this needs to be dealt with
        #if generator type is unknown
```

```
            GENRO = np.array([11,6,8,10,0,1,7,9,10,2,3,4,12,13])
            indextrans2 = np.array([3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, 20])
            GENSA = np.array([9,5,7,8,0,1,6,8,2,3,10,11])
            indextrans4 = np.array([3,5,6,7,8,9,10,12,14,15,19,20])
            GENTPJU1 = np.array([12,6,8,10,0,1,7,9,11,2,3,4,13,14])
            indextrans6 = np.array([3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, 20])
            GENCLS = np.array([0,1])
            indextrans8 = np.array([15,16])
            if genMdl=='GENCLS':

                for i in range(0,2):
                    ierr,Mparameter[iM,i+3]=psspy.dsrval('CON', (icon0+i))

                    Mparameter2[iM, indextrans8[i]] = Mparameter[iM, (3+GENCLS[i])]


            if (genMdl=='GENSAL')|(genMdl=='GENSAE'):
                for i in range(0,11):
                    ierr, Mparameter[iM, i+3] = psspy.dsrval('CON', (icon0 + i ))
                for i in range(0,11):
                    Mparameter2[iM, indextrans4[i]] = Mparameter[iM, (3+GENSA[i])]
            if (genMdl=='GENROU')|(genMdl=='GENROE'):
                for i in range(0,13):
                    ierr,Mparameter[iM,i+3]=psspy.dsrval('CON', (icon0+i))
                for i in range(0,13):
                    Mparameter2[iM, indextrans2[i]] = Mparameter[iM, (3+GENRO[i])]
            if (genMdl=='GENTPJU1'):
                for i in range(0,15):
                    ierr,Mparameter[iM,i+3]=psspy.dsrval('CON', (icon0+i))
                for i in range(0,15):
                    if i<indextrans6.shape[0]:
                        Mparameter2[iM, indextrans6[i]] = Mparameter[iM, (3+GENTPJU1[i])]



    #replace indextrans with corresponding parameters locations based on generator type.
    #todo add damping to the translation matrix
    if test ==0 or test ==1:
        indextrans = np.array([11,6,8,10,0,1,7,9,10,2,3,4,12,13])
    if test == 1:
        indextrans3 = np.array([9,5,7,8,0,1,6,8,2,3,10,11])
        indextrans4 = np.array([3,5,6,7,8,9,10,12,14,15,19,20])

    if test ==0 or test ==1:
        indextrans2 = np.array([3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
         ↪  19, 20])
    if test ==1 or test ==0:
        for i in range(indextrans.shape[0]):
            if test == 1:
                Mparameter2[0:3,indextrans2[i]] =
                 ↪  np.array(Mparameter[0:3,(3+indextrans[i])])
                Mparameter2[4:6, indextrans2[i]] =
                 ↪  np.array(Mparameter[4:6, (3 + indextrans[i])])
                if i<indextrans3.shape[0]:
                    Mparameter2[3, indextrans4[i]] =
                     ↪  np.array(Mparameter[3, (3 + indextrans3[i])])
            else:
                Mparameter2[:, indextrans2[i]] = np.array(Mparameter[:, (3
                 ↪  + indextrans[i])])


    Mparameter2[:,18] = Mparameter2[:,1]
    Mparameter2[:,21] = np.ones(Nmach,dtype=float)
    Mparameter2[:,22] = np.ones(Nmach,dtype=float)
    # fix fractions of generatation for when the same bus has multiple
     ↪  generators
    if True:
        for i in range(Nmach):
            temp = np.argwhere(Mparameter2[:,1]==Mparameter2[i,1])
            if temp.shape[0]>1:
                xxx = np.argwhere(bus[:,0]==Mparameter2[temp[0],1])
                sumP = np.sum(bus[xxx,3])
                sumQ = np.sum(bus[xxx,4])
                if sumP!=0:
                    Mparameter2[i,21] = abs(PQGenM[i].real/sumP)
                    if Mparameter2[i,21]>1:
                        print(Mparameter2[i,21])
                if sumQ!=0:
                    Mparameter2[i,22] = abs(PQGenM[i].imag/sumQ)
                    if Mparameter2[i,22]>1:
                        print(Mparameter2[i,22])

    Mparameter2[:,16] = 0
    Mparameter2[:,17] = 0
    if initiatepsse==True:
        psspy.close_powerflow()
        psspy.pssehalt_2()
    np.savez(matfile,bus=bus,line=line,Mparameter2=Mparameter2)

    return
```

savepssedata.py

```
import os
import sys

import psspy
from psspy import _i, _f, _s
import redirect
import numpy as np
import scipy.io as sio
import os.path as path
import sys
import SetupPSSE2
import uuid

def savepssedata(savefile,dynfile,rbus,rline,rmac,
                fromFile,matfile,iter,folder,initiatepsse):

    folderup = path.abspath(path.join(__file__ ,"../../.."))
    # setup Python to control PSSE
    if initiatepsse==True:
        #setup_psse_from_python()
        SetupPSSE2.setup_psse_from_python(output_to_file=True,
                                          append=False,CasePath=
                                            folderup+'\\temp\\')
        # open a case; or read a .raw file with psspy.read()
        psspy.case(savefile)

    if fromFile == True:
        mat = np.load(matfile + '.npz')
        rbus = mat['bus']
        rline = mat['line']
        rmac= mat['mac']

    ierr = psspy.dyre_new([1,1,1,1],dynfile,folderup+'\\temp\\'+
                          matfile+'_conec.flx',
                          folderup+'\\temp\\'+matfile+
                          '_conet.flx',folderup+'\\temp\\'+
                                matfile+'_compile1.bat')
    #
    # #uncomment if you want to load a snp file
    # #ierr = psspy.rstr('IEEE 39-Bus Dynamic DataWGoverner.snp')
    #
    subsystem = -1

    ierr = psspy.bsys(subsystem,0,[ 0.4,
    ↪   500.],len(areano),areano,0,[],0,[],0,[])
    ierr, Nmach=psspy.amachcount(subsystem, 1)            # get no of
    ↪   machines in the subsystem
    ierr, iMbus = psspy.amachint(subsystem, 1, ['NUMBER','WMOD']) # get
    ↪   machine bus numbers
    ierr, cMids = psspy.amachchar(subsystem, 1, 'ID')     # get machine IDs
    _iMbus=np.array(iMbus[0])
    _iMbusType = np.array(iMbus[1])
    # remove wind machines so that they are not changed
    iMbus = np.delete(_iMbus, np.where(_iMbusType !=0))
    cMids=np.array(cMids[0])

    for i in range(iMbus.shape[0]):
        if not any(map(len,np.where(rmac[:,1]==iMbus[i]))):
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 1, 0)
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 2, 0)
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 3, 0)
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 4, 0)
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 5, 0)
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 6, 0)
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 7, 0)
            ierr = psspy.plmod_status(iMbus[i], cMids[i], 8, 0)
            if not any(map(len,np.where(rbus[:,0]==iMbus[i]))):
                ierr = psspy.bus_chng_3(iMbus[i].astype(int), [4], [], _s)


    ierr, bus1 = psspy.abusint(subsystem, 1, ['NUMBER', 'TYPE'])
    bus1 = np.array(bus1).T
    ierr, busnames = psspy.abuschar(subsystem, 1, 'NAME')
    busnames = np.array(busnames).T


    for i in range(rbus.shape[0]):
        #if i >= bus1.shape[0]:
        if rbus[i, 9] == 1:
            rbus[i, 9] = 3
        elif rbus[i, 9] == 3:
            rbus[i, 9] = 1
        # bus numbers, voltages and angles
        if (i >= bus1.shape[0]) or not (any(map(len, np.where(bus1[:,0] ==
         ↪  rbus[i, 0])))):
            ierr = psspy.bus_data_3(rbus[i,0].astype(int),
             ↪  [rbus[i,9].astype(int), 9999, 1, 1], [0,rbus[i,1],
                                    rbus[i,2],1.1, 0.9, 1.1, 0.9],)
        else:
            if any(map(len, np.where(iMbus[:] == rbus[i, 0]))):
                if rbus[i,9].astype(int) == 1:
                    ierr = psspy.bus_chng_3(rbus[i,0].astype(int),
                     ↪  [rbus[i,9].astype(int)], [], busnames[i][0])
                elif rbus[i,9].astype(int) ==2:
                    subsystem = 1
                    busno = rbus[i, 0].astype(int)
                    ierr = psspy.bsys(subsystem,0,[ 0.4,
                     ↪  500.],0,[],1,[busno],0,[],0,[])
                    ierr, __Nmach =psspy.amachcount(subsystem, 1)
                     ↪  # get no of machines in the subsystem
                    ierr, __iMbus = psspy.amachint(subsystem, 1,
                     ↪  ['NUMBER']) # get machine bus numbers
                    ierr, __cMids = psspy.amachchar(subsystem, 1, 'ID')
                     ↪  # get machine IDs
                    ierr, __fMGQs = psspy.amachreal(subsystem, 1,
                     ↪  ['PGEN','QGEN'])
                    ierr, __totalPG = psspy.agenbuscplx(subsystem, 3,
                     ↪  ['PQGEN'])
                    __totalPG = __totalPG[0][0].real
                    if __Nmach>1:
                        for kk in range(__Nmach):

                            matrix = rmac[np.where(rmac[:,1]==
                                        rbus[i,0].astype(int))]
                            remove = True
                            for jj in range(matrix.shape[0]):
                                check = __totalPG*matrix[jj,21]
                                #print 'check: '
                                #print check
                                if abs(__fMGQs[0][kk] - check)>0.0001:
                                    pass


                                else:
                                    remove = False
                            if remove==True:
                                ierr =
                                 ↪  psspy.machine_chng_2(__iMbus[0][kk],
                                 ↪  __cMids[0][kk], [0], [])
        subsystem = -1

        if rbus[i,9]==2 or rbus[i,9]==3:
            if not any(map(len, np.where(iMbus[:] == rbus[i, 0]))) and
             ↪  any(map(len, np.where(rmac[:,1] == rbus[i, 0]))):
                ierr = psspy.plant_data(rbus[i,0].astype(int),
                 ↪  [0],[rbus[i,1],100])#todo how to capture and save
                #scheduled voltage at plants here we are assuming bus
                 ↪  voltage = scheduled voltage which is not always the
                 ↪  case
                #todo check that using xd'' is the right thing for Zsource
                ierr = psspy.machine_data_2(rbus[i,0].astype(int),
                        '1' , [1,1,0,0,0,0],[rbus[i,3]*100,
                        rbus[i,4]*100,
                        9999.0, -9999.0, 9999.0, 0.0,
                        rmac[np.where(rmac[:,1]==
                                rbus[i,0].astype(int)),2],
                        rmac[np.where(rmac[:,1]==
                                rbus[i,0].astype(int)),4],
                        rmac[np.where(rmac[:,1]==
                                rbus[i,0].astype(int)),7],
                        0,0, 1,1,1,1,1,1])

    ierr, line1 = psspy.abrnint(subsystem, 1, 1, 3, 1,
     ↪  ['FROMNUMBER','TONUMBER'])
    line1 = np.array(line1)

    for i in range(line1.shape[1]):
        if not (any(map(len, np.where((rline[:,0] == line1[0, i]) &
         ↪  (rline[:,1] == line1[1, i]))))):
            if rline[i,5] ==0 and rline[i,6] ==0:
                ierr = psspy.branch_chng(line1[0,i], line1[1,i], _s,
                 ↪  [0,_i,_i,_i,_i,_i],
                 ↪  [_f,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f,_f])
            else:
                ierr = psspy.two_winding_chng_4(line1[0,i], line1[1,i],
                 ↪  _s, [0,_i,_i,_i,_i,_i,_i,_i,_i,_i,_i,_i,_i,_i,_i], [_f
                 ↪  for n in range(24)], '')

    for i in range(rline.shape[0]):
        if i>=line1.shape[1] or not (any(map(len, np.where((line1[0,:] ==
         ↪  rline[i, 0]) & (line1[1,:] == rline[i, 1]))))):
```

```
        if (rline[i,5]==0 and rline[i,6]==0):# or (rline[i,5]==1 and rline[i,6]==0):

            ierr = psspy.branch_data(rline[i, 0].astype(int), rline[i, 1].astype(int), uuid.uuid4().hex[:2].upper(),
                                       [1, rline[i, 0].astype(int), 1, 0, 0, 0],
                                       [rline[i, 2], rline[i, 3], rline[i, 4],0,0,0, 0, 0, 0, 0, 0, 1, 1, 1, 1])

            ierr, realaro = psspy.two_winding_data_4(rline[i, 0].astype(int), rline[i, 1].astype(int),uuid.uuid4().hex[:2].upper() ,
                                                     [1, rline[i, 0].astype(int), 1, 0, 0, 0, 33, 0, rline[i, 0].astype(int), 0, 1, 0, 1, 1, 1],
                                                     [rline[i, 2], rline[i, 3], 100, rline[i, 5], 0, rline[
                                                         i, 6], 1, 0, 0,0,0,1, 1, 1, 1, 0, 0, 1.1, 0.9, 1.1, 0.9, 0, 0, 0],'')


# add machine models to existing file, assumes GENROU model and zero dumping except when t'qo = 0 then use GENSAL
for i in range(rmac.shape[0]):
    if not any(map(len, np.where(iMbus[:]==rmac[i, 1]))):
        if rmac[i,11]!=0 and rmac[i,13]!=0:
            ierr = psspy.add_plant_model(rmac[i,1].astype(int), '1',1, 'GENROU', 0, '', 0,
                                []', [], 14, [rmac[i,8], rmac[i,9], rmac[i,13],
                                 ↪  rmac[i,14], rmac[i,15], 0, rmac[i,5],
                                 ↪  rmac[i,10], rmac[i,6], rmac[i,11],
                                 ↪  rmac[i,7], rmac[i,3], rmac[i,19],
                                 ↪  rmac[i,20]])
            else:
                ierr = psspy.add_plant_model(rmac[i,1].astype(int), '1',1,
                 ↪  'GENSAL', 0, '', 0,
                                [], [], 12, [rmac[i,8], rmac[i,9], rmac[i,14],
                                 ↪  rmac[i,15], 0, rmac[i,5], rmac[i,10],
                                 ↪  rmac[i,6], rmac[i,7], rmac[i,3],
                                 ↪  rmac[i,19], rmac[i,20]])

    ierr = psspy.dyda(-1, 1, [2,1,0], 0,folder+
     ↪  str(iter)+'_COH_OUTPUT.dyr')
    ierr = psspy.save(folder+str(iter)+'_COH_OUTPUT.sav')
    if initiatepsse==True:
        psspy.close_powerflow()
        psspy.pssehalt_2()
    return
```

C.3. Simulation and Automation Tools

Description of main modules:

- *RunDynamicCase.py*: this module prepares the case for dynamic simulations and performs the specified dynamic simulation. It uses available PSS®E modules to perform the simulations without needing to run the study manually on PSS®E. This module also performs static network reduction using PSS®E EEQV tool
- *PlotDynamicData*: this module uses the raw results of *RunDynamicCase.py* to plot specified signals and calculate MSE between full case and reduced case

RunDynamicCase.py

```
# main program extracts data from psse save and dyr files,
# it is a function and has to be run from within coh_main.

import os
import sys
#import psse34
import psspy
from psspy import _i,_f
import redirect
import os.path as path
from SetupPSSE2 import SetupPSSE
from datetime import datetime
from CleanFolder import CleanFolder
t1=datetime.now()
def RunDynamicCase(savefile,dynfile,outfile,studybus):

    folderup = path.abspath(path.join(__file__ ,"../.."))

    # open a case; or read a .raw file with psspy.read()

    psspy.case(savefile)


    # solve the case
    #psspy.fnsl([0,0,0,1,0,0,99,0])

    psspy.fdns([0,0,0,1,0,0,99,0])


    ierr = psspy.dyre_new([1,1,1,1],dynfile,folderup+
                         '\\temp\\'+studyname+'_conec.flx',
                         folderup+'\\temp\\'+studyname+'_conet.flx',
                         folderup+'\\temp\\'+studyname+'_compile.bat')

    #uncomment if you want to load a snp file
    #ierr = psspy.rstr('IEEE 39-Bus Dynamic DataWGoverner.snp')

    psspy.cong(1)
    psspy.conl(0,1,1,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,2,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.conl(0,1,3,[0,0],[ 100.0,0.0,0.0, 100.0])
    psspy.fact()
    psspy.tysl(0)
    # Channels

    # for IEEE 39-Bus
    #psspy.chsb(0,1,[-1,-1,-1,1,1,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,2,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,3,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,4,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,5,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,6,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,7,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,12,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,13,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,16,0])
    #
    ##for branches MW/MVar flow
    #psspy.chsb(0,1,[-1,-1,-1,1,17,0])
    #psspy.chsb(0,1,[-1,-1,-1,1,18,0])


    # for Alberta

    psspy.bsys(1,0,[0.0,0.0],0,[],3,[773,772,129],0,[],0,[])
    psspy.chsb(1,0,[-1,-1,-1,1,1,0])
    psspy.chsb(1,0,[-1,-1,-1,1,2,0])
    psspy.chsb(1,0,[-1,-1,-1,1,3,0])
    psspy.chsb(1,0,[-1,-1,-1,1,4,0])
    psspy.chsb(1,0,[-1,-1,-1,1,5,0])
    psspy.chsb(1,0,[-1,-1,-1,1,6,0])
    psspy.chsb(1,0,[-1,-1,-1,1,7,0])
    psspy.chsb(1,0,[-1,-1,-1,1,13,0])
    psspy.chsb(1,0,[-1,-1,-1,1,16,0])

    #psspy.set_relang(1,30,r"""1""")
    psspy.set_relang(0, -1, "")
    psspy.dynamics_solution_param_2([_i,_i,_i,_i,_i,_i,_i,_i],[_f,_f,
     ↪  0.00416,_f,_f,_f,_f,_f])
    psspy.strt(0,outfile)
    psspy.run(0, 1.0,0,1,0)
    psspy.dist_bus_fault(studybus,1, 0.0,[0.0,-0.2E+10])
    psspy.run(0, 1.0667,0,1,0)
    psspy.dist_clear_fault(1)
    #psspy.dist_branch_trip(160, 772, '03')
    #psspy.dist_machine_trip(30,'1')
    #psspy.dist_branch_trip(2,25,r"""1""")
    psspy.run(0, 20.0, 0, 1, 0)
    ierr = psspy.delete_all_plot_channels()
    psspy.close_powerflow()

    return


if __name__ == '__main__':

    if len(sys.argv)==3:
        studybus = int(sys.argv[2])
        studyname = sys.argv[1]
    else:
        studybus = 23


    folderup = path.abspath(path.join(__file__ ,"../.."))
    if not os.path.exists(folderup+'\\output\\'+'\\'+studyname+'\\'):
        os.makedirs(folderup+'\\output\\'+'\\'+studyname+'\\')
    CleanFolder(folderup+'\\output\\'+'\\'+studyname+'\\','.out')
    folder = folderup+'\\cohcases\\'+'\\'+studyname+'\\'
    case_list = []
    case_list.append('Original')
    for file in os.listdir(folder):
        if file.endswith(".sav"):
            case_list.append(file.replace('.sav',''))

    files = case_list
    nooffiles = len(files)
    # setup Python to control PSSE
    #setup_psse_from_python()
    PSSEObject = SetupPSSE()
    temppath = folderup+'\\temp\\'
    PSSEObject.UpdateOutputFile(Outputfile=temppath+studyname)
    #SetupPSSE2.setup_psse_from_python(output_to_file=True,
     ↪  append=False,CasePath=folderup+'\\temp\\')

    NoDCCases = ['retain machine 37 New' , 'retain machine 37 30 and 39
     ↪  New',
                'retain machine 37 New test2' , 'retain machine 37 30 and
                 ↪  39 New test2',
                'podmore aggregation test','slow aggregation
                 ↪  test','inertial aggregation test',
                'podmore aggregation branch trip','slow aggregation branch
                 ↪  trip',
                'inertial aggregation branch trip']
    DCCases = ['retain machine 30 37 39_HVDC bus fault and HVDC All
     ↪  constant New' ,
               'retain machine 35 36_HVDC bus fault and HVDC All constant
                ↪  New',
               'retain machine 30 37 39_HVDC bus fault and HVDC Iden
                ↪  constant New' ,
               'retain machine 35 36_HVDC bus fault and HVDC Iden constant
                ↪  New']
    DCWindCases = ['retain machine 30 37 39_HVDC and Wind_bus fault and
     ↪  Both All constant New',
                   'retain machine 30 37 39_HVDC and Wind_bus fault and
                    ↪  Both Iden constant New',
                   'retain machine 30 37 39_HVDC and Wind_bus fault and
                    ↪  Both Iden constant New_Long']
    Alberta = ['Alberta test', 'Alberta test2','Alberta HVDC and
     ↪  Wind','Alberta HVDC and Wind Small Study Area',
               'Alberta HVDC and Wind Constant HVDC','Alberta HVDC and
                ↪  Wind Different Distrubance HVDC Terminal Fault',
               'Alberta HVDC and Wind Extended Study Area for HVDC Fault',
               'Alberta Aggregation Geographical Features','Alberta HVDC
                ↪  and Wind wsnd significant snr']
    if studyname in NoDCCases:
        savefile = folderup + '\\Full Cases\\IEEE 39-Bus Working
         ↪  Case_2.sav'
        dynfile =  folderup + '\\Full Cases\\IEEE 39-Bus Dynamic Data.dyr'
    elif studyname in DCCases:
        savefile = folderup + '\\Full Cases\\IEEE 39-Bus Working
         ↪  Case_2_HVDC.sav'
        dynfile =  folderup + '\\Full Cases\\IEEE 39-Bus Dynamic
         ↪  Data_HVDC.dyr'
    elif studyname in DCWindCases:
        savefile = folderup + '\\Full Cases\\IEEE 39-Bus Working
         ↪  Case_2_HVDC_Wind.sav'
        dynfile =  folderup + '\\Full Cases\\IEEE 39-Bus Dynamic
         ↪  Data_HVDC_Wind.dyr'
    elif studyname in Alberta:
        savefile = folderup + '\\Full
         ↪  Cases\\OPBC_2016SP_V33_R2_Latest.sav'
        dynfile =  folderup + '\\Full
         ↪  Cases\\OPBC_2016SP_Dynamic_V33_R2.dyr'
    for i in range(nooffiles):
        if files[i]=='Original':

            RunDynamicCase(savefile,
                           dynfile,
                           folderup + '\\output\\' + '\\' + studyname
                           + '\\' + files[i] + '.out', studybus)
        else:
            RunDynamicCase(folder+files[i]+'.sav',folder+files[i]+'.dyr',
                           folderup+'\\output\\'+'\\'+studyname+'\\'+
                           files[i]+'.out',studybus)



PSSEObject.PSSE_Output(False)
t2=datetime.now()
dt=t2-t1
print t1
print t2
```

```
s=dt.seconds+dt.microseconds*1.0e-6
h=int(s/3600)
m=int(s-h*3600)/60
s=(s-h*3600-m*60)
print 'time: %3i:%3i:%6.2f (hr:mi:ss)'%(h,m,s)
```

PlotDynamicData.py

```
from pylab import close,plot, title, show, grid, xlabel, ylabel
    ,xlim,ylim,legend,autoscale,figure,subplot,savefig,
    get_current_fig_manager,gcf
import matplotlib.pyplot as plt
import os, sys, collections
import numpy as np
import pandas as pd
import os.path as path
from datetime import datetime
from CleanFolder import CleanFolder
t1=datetime.now()
# 1. Data extraction/information

def PlotDynamicData(ori,outfile1,list,studymachine
                    ,studybus,first,studyname):
    import dyntools
    folderup = path.abspath(path.join(__file__ ,"../.."))

    # create object
    chnfobj1 =
     ↪  dyntools.CHNF(folderup+'\\output\\'+'\\'+studyname+'\\'+outfile1)
    sh_ttl1, ch_id1, ch_data1 = chnfobj1.get_data()
    chnfobj2 =
     ↪  dyntools.CHNF(folderup+'\\output\\'+'\\'+studyname+'\\'+ori)
    sh_ttl2, ch_id2, ch_data2 = chnfobj2.get_data()
    # Extract time as a variable
    time1 = ch_data1['time']
    time2 = ch_data2['time']
    # Number of channels, (-1 as 'time' is represented in channel data)
    count = len(ch_id1) - 1

##rotor angles

    noofmachines=0
    machlist = ''
    for k,v in ch_id1.iteritems():

        if 'ANGL' in v:
            noofmachines+=1
            machlist = machlist + str(v[5:7]+',')

    if studymachine != 773:
        keydata1 = [k for k, v in ch_id1.iteritems() if v == r"""ANGL
         ↪  """+str(studymachine)+r"""[ 22.000]1"""][0]
        keydataori = [k for k, v in ch_id2.iteritems() if v == r"""ANGL
         ↪  """+str(studymachine)+r"""[ 22.000]1"""][0]
    else:
        keydata1 = [k for k, v in ch_id1.iteritems() if v == r"""ANGL
         ↪  """+str(studymachine)+r"""[SECST 20.000]1"""][0]
        keydataori = [k for k, v in ch_id2.iteritems() if v == r"""ANGL
         ↪  """+str(studymachine)+r"""[SECST 20.000]1"""][0]



    data1 = ch_data1[keydata1]
    dataori = ch_data2[keydataori]
    keydataori2 = False
    keydata2 = False
    for k,v in ch_id2.iteritems():
        if studymachine==37:
            if v==r"""ANGL 30[ 22.000]1""":
                keydataori2 = k
        elif studymachine==35:
            if v==r"""ANGL 36[ 22.000]1""":
                keydataori2 = k
        elif studymachine==773:
            if v==r"""ANGL 129[SUND#1GN 18.000]1""":
                keydataori2 = k

    if len(data1)< len(dataori):
        AMSE = np.square(np.subtract(data1, dataori[0:len(data1)])).mean()
    elif len(data1)>len(dataori):
        AMSE = np.square(np.subtract(data1[0:len(dataori)],
         ↪  dataori)).mean()
    elif len(data1)==len(dataori):
        AMSE = np.square(np.subtract(data1, dataori)).mean()
    fig = figure(1)
    ax = fig.add_subplot(111)
    l1, = ax.plot(time1,data1,label='Reduced',color='red')
    l2, = ax.plot(time2,dataori,label='Full',color='blue')
    if outfile1!='Original.out':
        ax.text(0.95, 0.01, 'MSE = '+'% .2E' % AMSE+'\n'+'# of Gens = '
                +str(list.loc[list.File==
                int(outfile1[0:2].replace('_',''),
                '#Gen Reduced'].values[0])+'\n',
            verticalalignment='bottom',
            horizontalalignment='right',
            transform=ax.transAxes,
            color='black', fontsize=10)
    ylabel("Angle (Deg)")
    ax.grid()
    title('Rotor Angle of Machine '+str(studymachine))
    fig.legend((l1, l2), ('Equivalent', 'Full'), loc='upper right')

    xlabel('Time: Seconds')

    ax.autoscale(enable=True, axis='both', tight=None)
    xlim([0,5])

    fig.savefig(folderup+r"""\\output\\plots\\"""+'\\'+studyname+'\\'+
                outfile1+'_'+ch_id1[keydata1]+'_'+str(studymachine)+'.png'
                ,dpi=300)
    fig.clear()


    fig2 = figure(2)
    ax1 = fig2.add_subplot(311)
    ax1.plot(time1, data1, label='Reduced', color='red')
    ax1.plot(time2, dataori, label='Full', color='blue')
    if outfile1 != 'Original.out':
        ax1.text(0.95, 0.2, 'MSE = ' + '% .2E' % AMSE + '\n' + '# of Gens
         ↪  = ' + str(
            list.loc[list.File == int(outfile1[0:2].replace('_', '')),
             ↪  '#Gen Reduced'].values[0]) + '\n',
                 verticalalignment='bottom', horizontalalignment='right',
                 transform=ax1.transAxes,
                 color='black', fontsize=12)
    ax1.set(ylabel = 'Angle (deg)')
    ax1.grid()
    ax1.title.set_text('Rotor Angle of Machine ' + str(studymachine))


##power swings
    if studymachine != 773:
        keydata1 = [k for k, v in ch_id1.iteritems() if v == r"""POWR
         ↪  """+str(studymachine)+r"""[ 22.000]1"""][0]
        keydataori = [k for k, v in ch_id2.iteritems() if v == r"""POWR
         ↪  """+str(studymachine)+r"""[ 22.000]1"""][0]

    else:
        keydata1 = [k for k, v in ch_id1.iteritems() if v == r"""POWR
         ↪  """+str(studymachine)+r"""[SECST 20.000]1"""][0]
        keydataori = [k for k, v in ch_id2.iteritems() if v == r"""POWR
         ↪  """+str(studymachine)+r"""[SECST 20.000]1"""][0]
    data1 = ch_data1[keydata1]
    dataori = ch_data2[keydataori]

    if len(data1)< len(dataori):
        PMSE = np.square(np.subtract(data1, dataori[0:len(data1)])).mean()
    elif len(data1)>len(dataori):
        PMSE = np.square(np.subtract(data1[0:len(dataori)],
         ↪  dataori)).mean()
    elif len(data1)==len(dataori):
        PMSE = np.square(np.subtract(data1, dataori)).mean()
    fig = figure(1)
    ax = fig.add_subplot(111)
    l1, = ax.plot(time1,data1,label='Reduced',color='red')
    l2, = ax.plot(time2,dataori,label='Full',color='blue')
    fig.legend((l1, l2), ('Equivalent', 'Full'), loc='upper right')
    if outfile1!='Original.out':
        ax.text(0.95, 0.01, 'MSE = '+'% .2E' % PMSE+'\n'+'# of Gens = '+
        str(list.loc[list.File==int(outfile1[0:2].replace('_','')),
        '#Gen Reduced'].values[0])+'\n',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax.transAxes,
        color='black', fontsize=15)
    ylabel('Power (PU)')
    ax.grid()
    ax.autoscale(enable=True, axis='both', tight=None)
    xlim([0,5])
    title('Power Swing for Machine '+str(studymachine))
    xlabel('Time: Seconds')

    fig.savefig(folderup+r"""\\output\\plots\\"""+'\\'+studyname+'\\'+
                outfile1+'_'+ch_id1[keydata1]+'_'+str(studymachine)+'.png'
                ,dpi=300)
    fig.clear()

    ax2 = fig2.add_subplot(312,sharex=ax1)

    ax2.plot(time1,data1,label='Reduced',color='red')
    ax2.plot(time2,dataori,label='Full',color='blue')
    if outfile1!='Original.out':
        ax2.text(0.95, 0.01, 'MSE = '+'% .2E' % PMSE+'\n'+'# of Gens = '+
        str(list.loc[list.File==int(outfile1[0:2].replace('_','')),
        '#Gen Reduced'].values[0])+'\n',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax2.transAxes,
        color='black', fontsize=12)
    ax2.set(ylabel = 'Power (PU)')
    ax2.grid()
    ax2.title.set_text('Power Swing for Machine '+str(studymachine))
    plt.setp(ax.get_xticklabels(), visible=False)

##voltage at bus 2
    if studymachine != 773:
        keydata1 = [k for k, v in ch_id1.iteritems() if v == r"""VOLT
         ↪  """+str(studybus)+r""" [ 345.00]"""][0]
        keydataori = [k for k, v in ch_id2.iteritems() if v == r"""VOLT
         ↪  """+str(studybus)+r""" [ 345.00]"""][0]
    else:
        keydata1 = [k for k, v in ch_id1.iteritems() if v == r"""VOLT
         ↪  """+str(studybus)+r""" [ENMX25S7 240.00]"""][0]
        keydataori = [k for k, v in ch_id2.iteritems() if v == r"""VOLT
         ↪  """+str(studybus)+r""" [ENMX25S7 240.00]"""][0]
    data1 = ch_data1[keydata1]
    dataori = ch_data2[keydataori]

    if len(data1)< len(dataori):
        VMSE = np.square(np.subtract(data1, dataori[0:len(data1)])).mean()
    elif len(data1)>len(dataori):
        VMSE = np.square(np.subtract(data1[0:len(dataori)],
         ↪  dataori)).mean()
    elif len(data1)==len(dataori):
        VMSE = np.square(np.subtract(data1, dataori)).mean()
    fig = figure(1)
    ax = fig.add_subplot(111)
    l1, = ax.plot(time1,data1,label='Reduced',color='red')
    l2, = ax.plot(time2,dataori,label='Full',color='blue')
    fig.legend((l1, l2), ('Equivalent', 'Full'), loc='upper right')
    if outfile1!='Original.out':
        ax.text(0.95, 0.01, 'MSE = '+'% .2E' % VMSE+'\n'+'# of Gens = '+
                str(list.loc[list.File==int(outfile1[0:2].replace('_',''))
                ,'#Gen Reduced'].values[0])+'\n',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax.transAxes,
        color='black', fontsize=15)
    ylabel('Voltage (PU)')
    ax.grid()
    ax.autoscale(enable=True, axis='both' , tight=None)
    xlim([0,5])
    title('Voltage of Bus '+str(studybus))
    xlabel('Time: Seconds')
    fig.savefig(folderup+r"""\\output\\plots\\"""+'\\'+
                studyname+'\\'+outfile1
                +'_'+ch_id1[keydata1]+'_'+
                str(studymachine)+'.png',dpi=300)
    fig.clear()

    ax2 = fig2.add_subplot(313,sharex=ax1)
    l1, = ax2.plot(time1,data1,label='Reduced',color='red')
    l2, = ax2.plot(time2,dataori,label='Full',color='blue')
    if outfile1!='Original.out':
        ax2.text(0.95, 0.01, 'MSE = '+'% .2E' % VMSE+'\n'+'# of Gens = '+
        str(list.loc[list.File==int(outfile1[0:2].replace('_','')),
        '#Gen Reduced'].values[0])+'\n',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax2.transAxes,
        color='black', fontsize=12)

    ax2.set(ylabel = 'Voltage (PU)', xlabel='Time: Seconds')
    ax2.grid()
    ax2.autoscale(enable=True, axis='both', tight=None)
    ax2.set_xlim(0.98,5)
    ax2.title.set_text('Voltage of Bus '+str(studybus))

    fig2.legend((l1, l2), ('Equivalent', 'Full'), 'upper right')
    fig2.tight_layout()
    fig2.savefig(folderup+r"""\\output\\plots\\"""+'\\'+studyname+'\\'+
    outfile1+'_'+str(studymachine)+'_Combined'+'.png',dpi=300)
    fig2.clear()
    close(fig2)



    if outfile1!='Original.out':

        File = int(outfile1.split('_')[0])

        df = pd.DataFrame.from_dict({'File':[File],
                            'Angle MSE':[AMSE],
                            'Power MSE':[PMSE],
                            'Voltage MSE':[VMSE],
                            #'# Gen': [noofmachines],
                                    #'Gens':[machlist],
                                    'n':list.loc[list.File==File,'n'].tolist(),
                                    'tol':list.loc[list.File==File,'tol'].tolist(),
                                    '#Gen Full':list.loc[list.File==File,'#Gen
                                     ↪  Full'].tolist(),
                                     '#Gen Reduced':list.loc[list.File==File
                                         ,'#Gen Reduced'].tolist(),
                                    'Modes_Original':list.loc[list.File==File,
                                         'Modes_Original'].tolist(),
                                    'Modes_After':list.loc[list.File==File
                                    ,'Modes_After'].tolist(),
                                    },)

        if first:
            df.to_csv(folderup+'\\output\\'+'\\'+studyname+'\\'+
            studyname+'.csv',columns=['File','#Gen Reduced','Angle MSE',
            'Power MSE','Voltage MSE','n','tol','#Gen
             ↪  Full','Modes_Original',
            'Modes_After'],index=False, mode='a')
        else:
            df.to_csv(folderup+'\\output\\'+'\\'+studyname+'\\'+
            studyname+'.csv',columns=['File','#Gen Reduced','Angle MSE',
            'Power MSE','Voltage MSE','n','tol','#Gen
             ↪  Full','Modes_Original',
            'Modes_After'],header=False,index=False,mode='a')









if __name__ == '__main__':

#    import psse34



    showB      = False      # True  --> create, save and show Excel
     ↪  spreadsheets and Plots when done
                            # False --> create, save but do not show Excel
                             ↪  spreadsheets and Plots when done


    if len(sys.argv)==4:
        studyname = sys.argv[1]
        studymachine = int(sys.argv[2])
        studybus = int(sys.argv[3])
    else:
        studymachine = 36
        studybus = 23

    folderup = path.abspath(path.join(__file__ ,"../.."))
    if not
     ↪  os.path.exists(folderup+'\\output\\plots\\'+'\\'+studyname+'\\'):
        os.makedirs(folderup+'\\output\\plots\\'+'\\'+studyname+'\\')
    CleanFolder(folderup+'\\output\\plots\\'+'\\'+studyname+'\\','.png')

    try:
        os.remove(folderup+'\\output\\'+'\\'+studyname+'\\'+
        studyname+'.csv')
    except OSError:
        pass
    list=pd.read_csv(folderup+'\\output\\'+'\\'+studyname+'\\'+'list.csv')

    case_list = []
    for file in os.listdir(folderup+'\\output\\'+'\\'+studyname+'\\'):
        if file.endswith(".out"):
            case_list.append(file.replace('.out',''))
    files = case_list
    nooffiles = len(files)
    first = True
    for i in range(nooffiles):
        PlotDynamicData('Original.out',files[i]+'.out',list,
        studymachine,studybus,first,studyname)
        if files[i]!='Original':
            first = False
table =
 ↪  pd.read_csv(folderup+'\\output\\'+'\\'+studyname+'\\'+studyname+'.csv')

table.sort_values(by=['Power
 ↪  MSE','n','tol'],ascending=[True,False,False],inplace=True)
table.to_csv(folderup+'\\output\\'+'\\'+studyname+'\\'+studyname+'.csv',
            columns=['File','#Gen Reduced',
                'Angle MSE','Power MSE','Voltage MSE','n','tol','#Gen
                 ↪  Full',
                'Modes_Original','Modes_After'],
                index=False,mode='w')

```

```
t2=datetime.now()
dt=t2-t1
print t1
print t2
s=dt.seconds+dt.microseconds*1.0e-6
h=int(s/3600)
m=int(s-h*3600)/60
s=(s-h*3600-m*60)
print 'time: %3i:%3i:%6.2f (hr:mi:ss)'%(h,m,s)
# =============================================
```