Propagation of Electromechanical Disturbances across Large Interconnected Power Systems and Extraction of Associated Modal Content from Measurement Data

Jason Noah Bank

Dissertation submitted to the Faculty of Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of:

Doctor of Philosophy

in

Electrical Engineering

Yilu Liu (Chair)
Jaime De La Ree Lopez
Virgilio A. Centeno
Richard W. Conners
Tao Lin

December 4, 2009
Blacksburg, Virginia

Keywords: Power System Dynamics, Electromechanical Wave Propagation, System Modeling and Reduction, Interarea Modes, Processing of Measurement Data, Measurement Data Visualization

Propagation of Electromechanical Disturbances across Large Interconnected Power Systems and Extraction of Associated Modal Content from Measurement Data

JASON NOAH BANK

ABSTRACT

Changes in power system operating conditions cause dynamic changes in angle and frequency. These disturbances propagate throughout the system area with finite speed. This propagation takes the form of a traveling wave whose arrival time at a particular point in the system can be observed using a wide-area measurement system (WAMS). Observations of these waves both through simulation and measurement data have demonstrated several factors that influence the speed at which a disturbance propagates through a system. Results of this testing are presented which demonstrate dependence on generator inertia, damping and line impedance. Considering a power system as an area with and uneven distribution of these parameters it is observed that a disturbance will propagate throughout a system at different rates in differing directions. This knowledge has applications in locating the originating point of a system disturbance, understanding the overall dynamic response of a power system, and determining the dependencies between various parts of that system.

A simplified power system simulator is developed using the swing equation and system power flow equations. This simplified modeling technique captures the phenomenon of traveling electromechanical waves and demonstrates the same dependencies as data derived from measurements and commercial power system simulation packages. The ultimate goal of this research is develop a methodology to approximate a real system with this simplified wave propagation model. In this architecture each measurement point would represent a pseudo-bus in the model. This procedure effectively lumps areas of the system into one equivalent bus with appropriately sized generators and loads. With the architecture of this reduced network determined its parameters maybe estimated so as to provide a best fit to the measurement data. Doing this effectively derives a data-driven equivalent system model.

With an appropriate equivalent model for a given system determined, incoming measurement data can be processed in real time to provide an indication of the system operating point. Additionally as the system state is read in through measurement data future measurements values along the same trajectory can be estimated. These estimates of future system values can provide information for advanced control and protection schemes.

Finally a procedure for the identification and extraction of interarea oscillations is developed. The dominant oscillatory frequency is identified from an event region then fit across the surrounding dataset. For each segment of this data set values of amplitude, phase and damping are derived for each measurement vector. Doing this builds up a picture of how the oscillation evolves over time and responds to system conditions. These results are presented in a graphical format as a movie tracking the modal phasors over time. Examples derived from real world measurement data are presented.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my primary advisor, Dr. Yilu Liu. Throughout my time in graduate school she provided invaluable advice and instruction which helped to further my studies. Additionally she allowed me a free hand to pursue several research topics which helped to develop the subject matter of this dissertation. Enough cannot be said about how instrumental her support was to this effort.

Additional thanks go out to the rest of the FNET development team members, both past and present. Among these is Dr. Richard Conners who provided insightful critical advice on several subjects which helped to define the direction of my research. Thanks also goes to Robert Gardner and Joshua Wang who were responsible for development of the necessary statistical models of the FNET data as well as pioneering the event triangulation and oscillation monitoring algorithms. Other essential FNET members include Tao Xia and Yingchen Zhang which maintained and expanded the infrastructure supporting the FNET system, without which the source data for much of this work would not be available.

Many thanks also go out to Dr. Jaime De la Ree Lopez of Virginia Tech who sparked my initial interest in power systems through his undergraduate courses. He also provided the graduate teaching assignments which constituted most of my financial support during my graduate career.

Last but not least I would like to thank Oak Ridge National Laboratory for the research fellowship under which I worked during the final stages of this effort. Through the support of the laboratory the developments of Chapter 5 were made possible. Related to this I would like to thank Olufemi Omitaomu who provided advice in the fields of data processing, pattern recognition, and signal analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

PROPAGATION OF ELECTROMECHANICAL DISTURBANCES ACROSS LARGE INTERCONNECTED POWER SYSTEMS

Electrical power systems are large interconnected networks consisting of thousands of individual elements spread over a large geographic area. These elements include electrical and mechanical equipment involved with the conversion and delivery of energy to the customer, as well as the customer loads themselves. This delivery process includes the conversion of energy from its primary fuel form to electrical energy, the transformation of that energy, the transmission of electrical energy across long distances, distribution to the customers and the control processes associated with these tasks. All of the elements of the system are interdependent in some manner and together form one large rotating machine spanning thousands of square miles. A power system is a constantly evolving over time and sensitive to changes in generation, demand and network topology.

A sudden change in the operating point of a power system caused by a step change in load demand or generation results in a transient time period over which there is a power mismatch at the point of the disturbance. This power mismatch is corrected as the system generators adjust their rotor angles and change their output electrical power to meet the new system power requirements and operating point. In order to accomplish this adjustment the rotor must change its speed and position relative to the stator. Additionally, each generator is a rotating mass connected mechanically to a prime mover. Since it is a physical element in motion the rotor has a mechanical inertia. Because of this inertia the generator is not capable of an instantaneous change and therefore moves along a slower trajectory to meet the system requirements.

The process by which this angle change occurs is governed by the swing equation, (0.1), which is developed from Newton's Second Law of Motion [1].

$$\frac{2H}{\omega}\frac{\partial^2\Delta\delta}{\partial t^2} = P_M - P_E - K_D\Delta\omega \qquad (0.1)$$

If the system (and thus the generator) is operating in steady-state the mechanical power, P_M, and electrical power, P_E, are equal. The speed change, $\Delta\omega$, will be zero in steady state as well, thus the right hand side of (0.1) will be zero. As the generator inertia, H, and speed, ω, are non-zero this means that the second derivative operator of (0.1) must be zero. This element is the second derivative of the change in machine angle, $\Delta\delta$, and describes the acceleration of the rotor angle. Since the acceleration is zero the rotor angle will remain in its current position.

If the system operating point should suddenly change the voltage angle on the local bus will change quickly to match it. The electrical power is dependent on this voltage angle and as such will change as well. For the most part the generator mechanical power can be considered constant over this time frame since the thermal and mechanical properties of the prime mover operate on a slower time constant than the electrical network. In this fashion a power mismatch is seen, the mechanical power has remained constant while the electrical power has changed. Since these two power values are now different and $\Delta\omega$ is still zero the right hand side of (0.1) is now non-zero. Now the acceleration of the rotor angle is non-zero and the machine will begin to move to meet the new operation point.

In this manner a generator will begin to move to meet the new electrical system conditions. As this process continues things become more complex. As the generator changes speed $\Delta\omega$ becomes non-zero and the generator damping factor K_D comes into play. Additionally, the acceleration of the generator angle will serve to influence the electrical power which then feed back into (0.1). If the final operating point is stable eventually the electrical power and mechanical power will balance out and the right hand side of (0.1) will approach zero. At this point the system has settled to its new stable operating point.

This process is not limited to one specific generator in the power system. When the step change event occurs at a point in the system the local generators will begin to respond in the manner described above. Since these local machines cannot respond fully in an instantaneous fashion a portion of power imbalance will be seen nearby generators, each of which then begins to respond. As these generators begin to respond another set outside of those will begin to response to the event. In this manner a propagating wavefront is seen moving through the span of the system as a change in angle as successively further generators begin their response to the event.

This behavior is known as electromechanical wave propagation, because it results from the interaction of electrical and mechanical elements of a power system. Unlike electromagnetic waves this propagation occurs at speeds much less than the speed of light. Electromechanical waves have been observed to move across large interconnected power systems at speeds in the range of 500 km/sec [2]. These lower speeds are mainly due to the generator inertias imposing a physical limitation on how quickly they can respond. Several other elements also govern the spatial speed of these wavefronts including network topology, transmission line impedance, and power flow directions.

The nature of electromechanical waves and their propagation has been investigated before in several works [2],[3],[4],[5]. This work aims to expand on this research field. The effects of system parameters and their distribution on propagation speed and wave shape will be assessed. A simplified dynamic system model is developed to expand this testing to very large, uniform, but discrete power system models. The simplified model is also employed to graphically demonstrate the electromechanical travelling wave phenomenon. The model is then used in conjunction with measurement data in a system reduction and approximation procedure. The final subject covered involves the extraction of interarea oscillatory modes from measurement data.

DESCRIPTION OF SOURCE DATA, THE FNET MEASUREMENT SYSTEM

Several sets of measurement data used in this document are drawn from the Frequency Network (FNET) system. FNET is an internet-based frequency monitoring system location in the North American Power System. This is an extremely low cost and quickly deployable wide-area frequency measurement system with high dynamic accuracy initially developed at Virginia Tech [3],[6],[7],[8],[9].

The FNET system is used to monitor and record the changes in voltage frequency in real time at various locations. The system consists of two major components: frequency disturbance recorders (FDRs) and the information management system (IMS). The FDRs perform local GPS synchronized frequency measurements and send data to a central server through the internet, and the IMS handles data collection, storage, communication, database operations, and a web service. There are currently more than 50 FDRs deployed around the United States. Figure 0.1 shows the spatial location of these FDRs and their respective interconnections (WECC, IE, or ERCOT). Each FDR measures the voltage phasor and frequency at a distribution level 110 V outlet and streams data at a rate of 10 points per second.

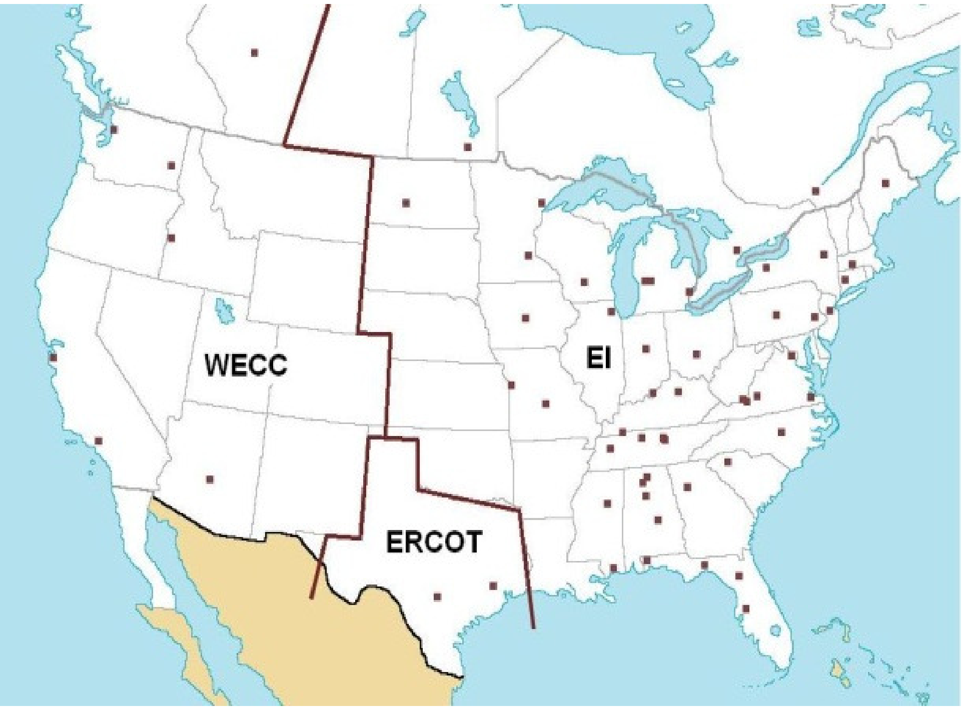


Figure 0.1: Spatial Distribution of FDRs as of June 2009

Observed Electromechanical Waves in Measurement Data

With the deployment on high time resolution Phasor Measurement Units (PMUs) it has become possible to observe the propagation of electromechanical waves as it happens. The FNET system is also capable of observing these disturbances in the frequency measurements. Up to this point the electromechanical wave has been described as a change in the angle of the voltage phasor. The voltage frequency is analogous to the integration of angle change and as such the electromechanical wave has a similar signature in the frequency measurements.

An example set of measurement data is presented in Figure 0.2. Here a generator tripped offline in the North American Eastern Interconnect (EI) power system, in Brunswick, NC. The frequency of the system responded to this event by dropping from 59.985 Hz to 59.960 Hz. The manner of this drop is governed by the swing equation and an electromechanical wave propagated throughout the system as part of this response. During this event seven FDRs recorded the voltage frequency at various points in the system. The frequency measurements at each of these seven points are given as a separate trace in Figure 0.2.

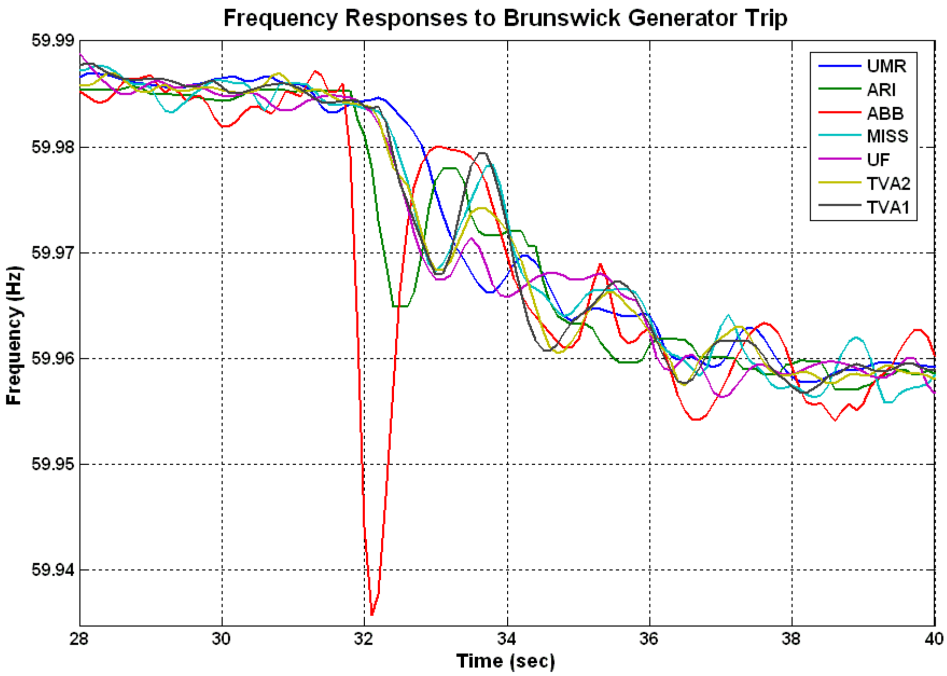


Figure 0.2: Example Frequency Response to Generation Trip

Inspection of Figure 0.2 reveals that the frequency of the ABB FDR located in Raleigh, NC begins to drop before the other six. This is as expected since it the closest measurement to the event location. As the travelling wave propagates outward from the event location it moves in all directions at approximately the same speed, thus it approaches the nearest measurement point first. The arrival of the wave in this case is seen as a drop in frequency. After moving past the ABB unit the wave is seen by the others units in order of their geographic distance from the event. This wavefront timing is graphically demonstrated by Figure 0.3. The time point at which each frequency measurement drops to 59.980 Hz was determined for each location and laid on a map with the location of the original event. Several concentric circles of different colors were also added to demonstrate the regions which the wave had moved through.

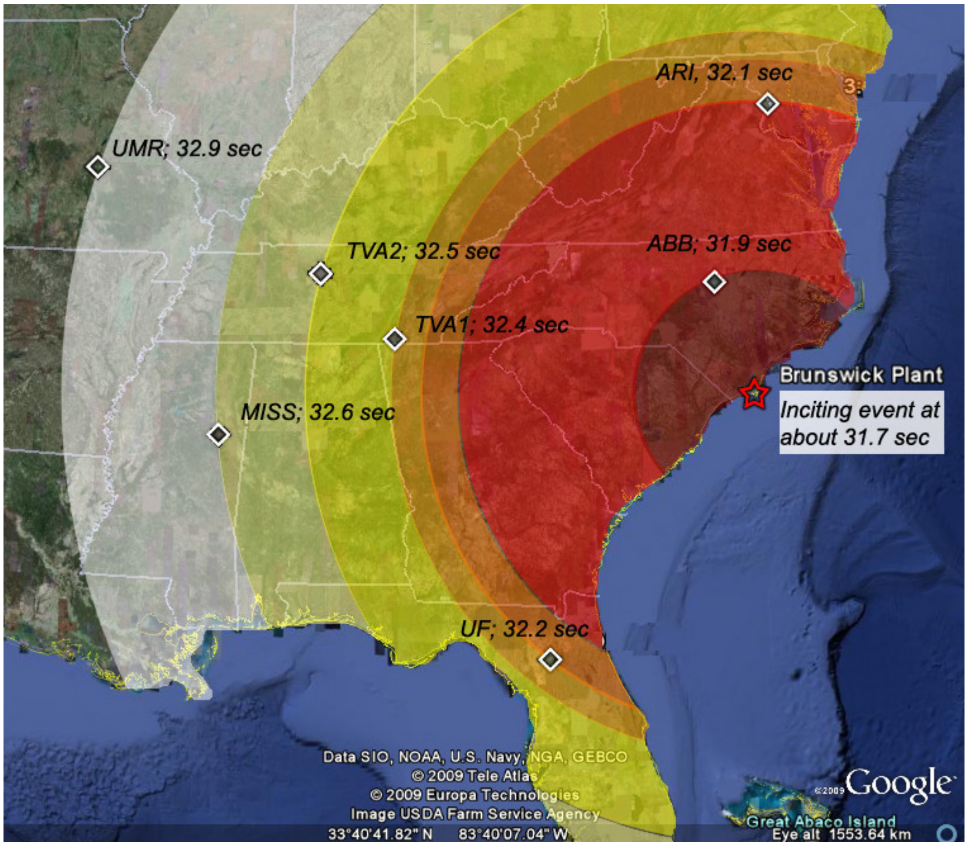


Figure 0.3: Progression of Traveling Wave across EI

The generation trip at Brunswick was estimated to occur at 31.7 seconds immediately afterward the resulting electromechanical wave began propagating outward in all directions at the same speed. It was then seen by the ABB unit at 31.9 seconds, by this point it had propagated through the dark red region of the map. After moving past the ABB unit the wave is next seen by the ARI unit at 32.1 seconds, covering the red region of the power system. This wave then continues to move through the power system and is seen by the remaining measurement points according to their distance from the originating location.

The preceding example is intended to demonstrate the effects of a sudden change in power (in this case the loss of a generator and the power it was producing). As such it is meant to provide some background on the phenomenon and the underlying properties. Several approximations have been made here to simplify the graphics. The major one being that wave speed is constant across all possible directions. This is only the case for a power system possessing a perfectly uniform spatial distribution of impedances, inertias, topology, and loads. This case corresponds to the continuum model of previous works [2]. In real systems these distributions are highly non-uniform and the electromechanical wave will propagate at different speeds in different directions. These non-uniformity effects result from the system architecture and will be considered in Chapter 3. With this background material present the discussion will now move into a more detailed description of each system parameter and how they affect the propagation of an electromechanical wave.

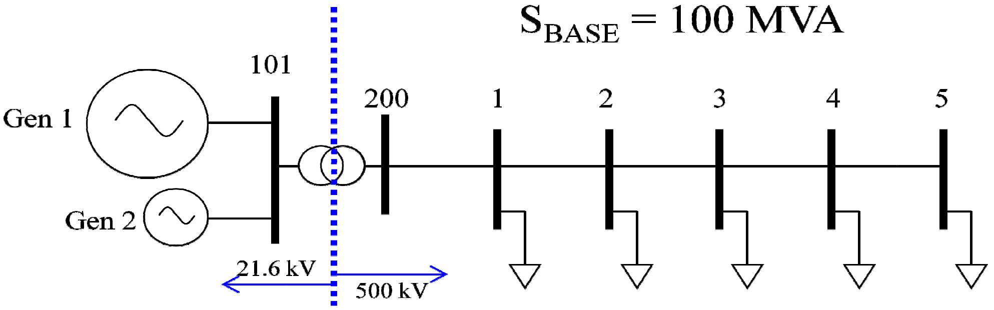
1 PRELIMINARY ELECTROMECHANICAL WAVE PROPAGATION STUDIES AND QUALITATIVE OBSERVATIONS

The investigation of propagating electromechanical waves which is covered throughout this work began with a series of simulations designed to demonstrate some of their basic properties. These simulations are based around small, simple power system models involving a series of busses connected in a single branch radial fashion by transmission lines. By varying the various parameters in these simple models and observing the wave propagation the critical parameters were asses qualitatively. These demonstrative test cases are built around very small and somewhat simplified models. Because of this the results presented in this chapter are less applicable to the large system multi-dimensional phenomenon that is the target of this investigation. However these experiments helped to define the research directions and help to explain some of the basic dependencies.

Two sets of these reduced order simulations are presented here. The first one is based in PSS/E and focused primarily on the effects of static loads and impedances in relation to wave propagation. The second study was implemented using MATLAB's SimPowerSystems and focuses more on the effects of rotating machines and inertia.

1.1 PSS/E SYSTEM STUDY AND LOADING EFFECTS

The first set of simulations used to explore the propagation of electromechanical waves was based in the PSS/E simulation package. A simple test system was established consisting of a string of busses attached to a generation station. This generation station consisted of a large generator and a smaller generator tied to a transformer. The other side of this transformer was then tied to the string of load busses. In order to incite an event the smaller generator would be dropped. Doing this would setup a propagating electromechanical wave. The architecture and parameters of this test system is depicted in Figure 1.1. Note that this system demonstrated very short electromechanical propagation delays as there is no mechanical inertia located on the string of load busses.



Generators:
Gen 1:
P = 650 MW
Q = 65 MVAR
Gen 2:
P = 100 MW
Q = 10 MVAR

Line Parameters:
R = 0.028 Ohm/km
X = 0.325 Ohm/km
B = 5.200 μS/km
Length = 80 km

Load Parameters:
P = 150 MW
Q = 70 MVAR
50% const. Current
50% const. Power

Figure 1.1: PSS/E Test System Base Case Configuration

For this set of testing the travelling waves were observed as disturbances in the frequency of the various bus voltages. Each simulation was initialized and run to 0.25 seconds; at this point generator 2 was removed from the system. After the removal of the generator the simulation was carried through to 5 seconds.

The bus voltage frequencies were recorded throughout this simulation procedure. The wavefront arrival at a particular bus was later determined from the point at which the bus frequency dropped to -0.004 per unit (59.76 Hz). Referencing this arrival time against the inciting event gives the propagation delay of the disturbance and equivalently a measure of the speed. This procedure of determining arrival time is demonstrated by Figure 1.2. In Figure 1.2 the axes are scaled to demonstrate the disturbance immediately after the generator loss. The generator was dropped at 0.25 seconds and it is seen that all of the bus frequencies fall off in order with the busses further away from the generators falling last. The disturbance arrival time at a particular bus is given by the time difference between when the small generator is dropped and when the bus frequency crosses -0.004 per unit. These arrival times for the Base Case are given in Table 1.1.

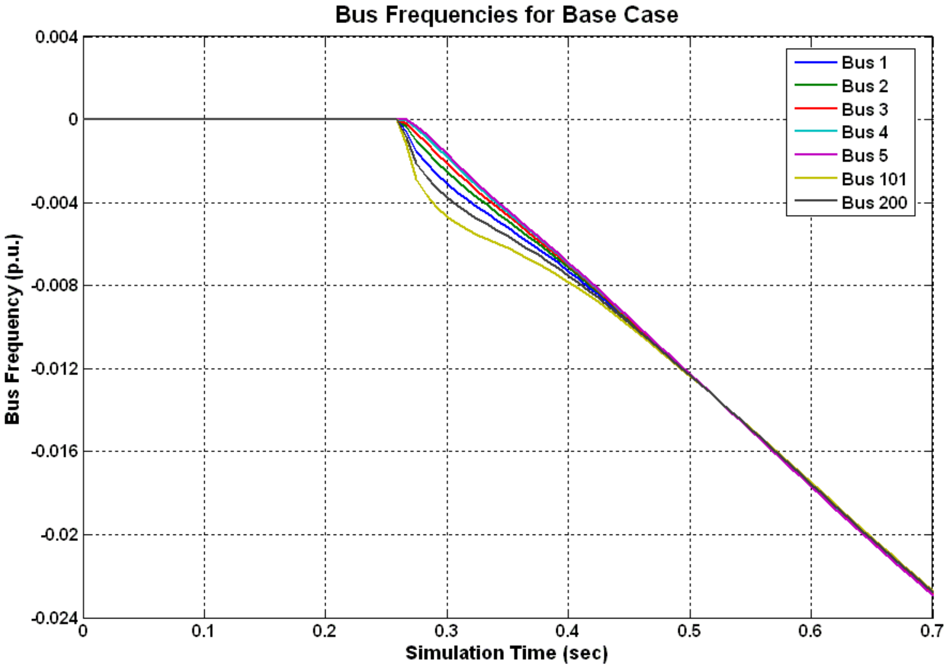


Figure 1.2: Response of Base Case

Bus Number	Arrival Time (ms)
101	38.2
200	55.4
1	69.8
2	80.1
3	86.7
4	90.6
5	92.4

Table 1.1: Wavefront Arrival Times for Base Case

In each of the subsequent tests one characteristic of the Base Case will be modified and the effects will be investigated holding everything else constant. Doing this allows investigation of one specific parameter, isolating it from the other system properties. Several different simulations were performed across different values of each parameter under test. Thus the dependence of wave propagation speed on the test parameter is observed. The tested parameters for these simulations included line length, load modeling and load power factor.

1.2 EFFECTS OF LINE LENGTH – PSS/E TESTS

The first batch of tests implemented on this PSS/E system involved the line lengths throughout the system. For each simulation the length of the transmission lines connecting the load busses were changed. The line length determines the total line impedance and shunt admittance and thus the system impedances and propagation delays change. Nine simulations were performed with line lengths ranging from 40 km to 120 km. After completing each of these simulations the propagation delays were computed in a fashion similar to that of Table 1.1.

The results of the line length tests are given in Figure 1.3. Here the independent axis is the length of the lines between the load busses while the dependant variable gives the propagation delays. Each color coded line corresponds to the propagation delays for one of the busses with the gaps between the lines giving the delays between successive busses. As expected here the busses closer to the disturbance see the disturbance first and thus have the shortest propagation delays.

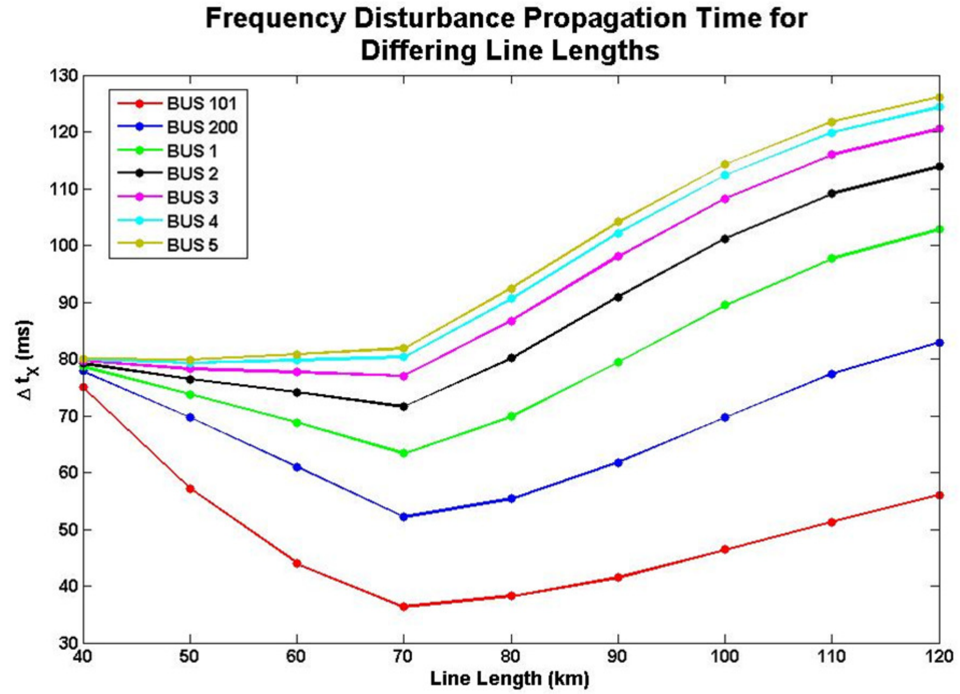


Figure 1.3: Results from Line Length Tests

Several observations can be made from the results presented in Figure 1.3. Firstly as expected increasing the line length in general produces an increase in the propagation delay times; this is seen as the general upward trend on the right hand side of the plot. At 70 km a clear minimal point is demonstrated, especially for the busses near the event. Further investigation showed that for line lengths less than 70 km the generators were initially supplying reactive power and above 70 km they were consuming reactive power. This indicated that for this system a line length of 70 km was the line size for which the reactive components of the line helped to support the load requirements. For this specific line length the reactive power sources and sinks were equally distributed and the line flows were minimized. Additionally the largest delays are seen between the first few busses despite the fact that all of the busses were evenly spaced in the simulations. Once again this can be attributed to power flows on the lines, the first line in the string has to carry all the load of the subsequent lines and thus it incurs the largest delays. These observations indicate that the propagation delay of an electromechanical wave is heavily dependent on the amount of power flow, both real and reactive, along the propagation path.

1.3 EFFECTS OF LOAD POWER FACTOR – PSS/E TESTS

The next set of testing using the PSS/E system involved modification of the load power factors. Here the apparent power of the loads was held constant while varying the power factor. Thus the ratio of real to reactive power was modified while holding the apparent power constant. Through each successive simulation the power factors of the loads were moved together, ranging from purely capacitive through unity to purely inductive. Several of these simulations demonstrated unstable cases. These unstable cases encompassed all of the capacitive loads and the inductive loads for which the power factor was less than 0.4. For all of these cases the system power flow failed to converge initially due to improper voltage support. The propagation delays seen through the stable simulations are presented in Figure 1.4.

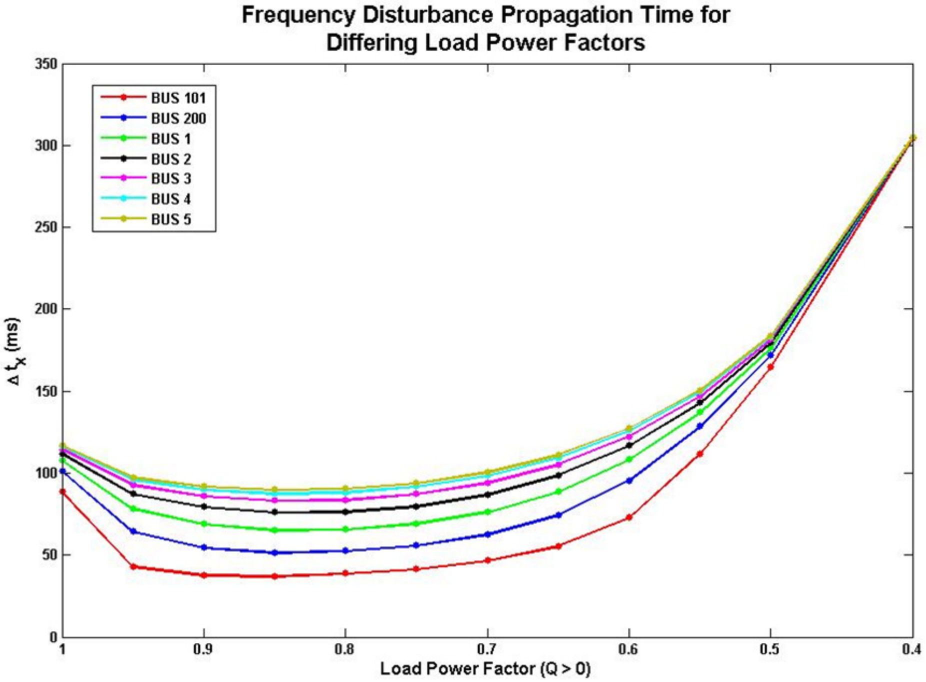


Figure 1.4: Results from Power Factor Tests

Figure 1.4 demonstrates a clear point at which the propagation delays are minimized. This point occurs when the load power factors are set at about 0.85 lagging. Further investigation of the line power flows for this case demonstrated that the reactive power flows were minimized. Here the reactive power demands of the load were counter balanced by the line effects, minimizing the line flows and the reactive power output of the generators. Observation of Figure 1.4 also reveals that as the power factor is pulled away from the minimal point the propagation delays increase in an exponential fashion with the delays at 0.4 *pf* more than 6 times those of the minimum point. Additionally the inter-bus delays decrease as the power factor is moved from the minimal point with all of the busses seeing the disturbance at nearly the same time.

This testing indicates that wave propagation is more influenced by reactive power flow than real power. Since apparent power was constant through these simulations the same effective power is flowing through the system with the only different between simulations being the relative amounts of real and reactive power. The observed delays are shortest when the lowest amount of reactive power is flowing in the system. This demonstrates a proportional relationship between the propagation delay through a path and the amount of reactive power flowing in that path.

1.4 EFFECTS OF LOAD MODELING – PSS/E TESTS

The last set of simulations present here for this PSS/E system focused on the effects of load modeling. The effects of three basic load models were investigated. These models included constant power, constant current and constant admittance loads. Throughout these simulations the load sizes remained constant with only the voltage characteristics changing. For each simulation the loads were modeled with set percentages of two specific load types. Doing this resulted in three series of simulations with the loads moving from 100% of one type of model to 100% of another model. These three series are defined by the three unique pairs of load model types.

Figure 1.5 gives the results from the simulations investigating a change from constant current to constant impedance loads. The left hand side of the abscissa axis, where the value is zero, represents a system with 100% constant current load modeling. Progression to the right corresponds to a reduction in the amount of constant current load and replacement with constant power load. Thus the 50 point is the 50% constant current, 50% constant power load modeling case and the 100 point is the 100% constant power load case. All of these loads are the same size across all the simulations; their only difference is their dynamic characteristics. As before the ordinate of Figure 1.5 is the propagation delay time.

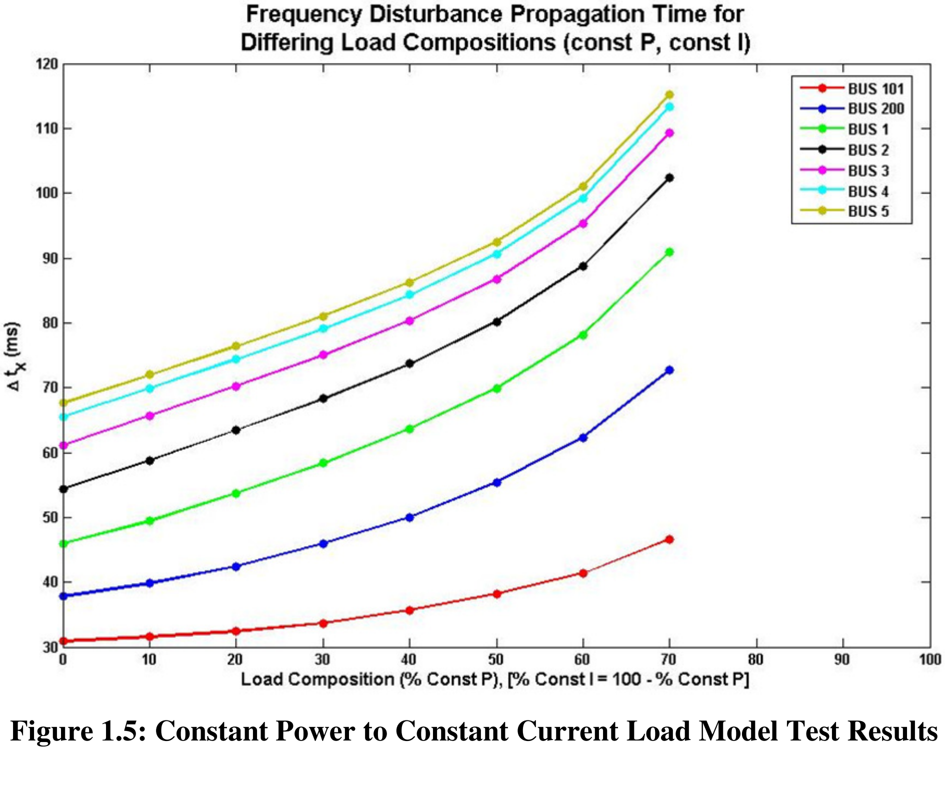


Figure 1.5: Constant Power to Constant Current Load Model Test Results

As seen in the results of Figure 1.5 a travelling electromechanical wavefront propagates quicker through constant current loads than it does through constant power loads. As the loads are changed from constant current to constant power the delay time increases in a near exponential fashion. Although the simulations with load models having more than 70% constant power failed to converge the general trend can be seen in the approach to that point.

The next set of load modeling simulations involved a transition from constant admittance to constant power loads. These results are presented in Figure 1.6. The majority of these simulations failed to converge initially and as such many of the data points are missing from Figure 1.6. Despite this it is once again observed that the constant power loads result in longer delay time and thus slower wavefront propagation.

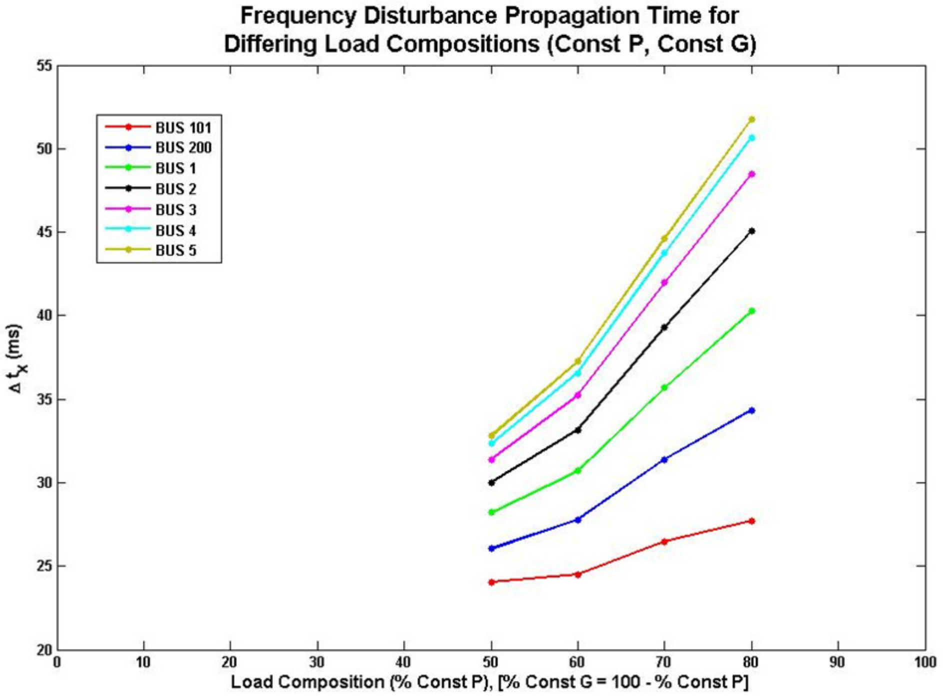


Figure 1.6: Constant Power to Constant Admittance Load Model Test Results

The third pair wise combination of load models, constant current and constant admittance, was also tested through a set of simulations. The propagation delays for this testing are given in Figure 1.7. The plot is similar to the two preceding ones but in this case the abscissa ranges from 100% constant admittance to 100% constant current. Although the load models involving a lot of constant current relationships failed to converge a similar trend to that of Figure 1.5 and Figure 1.6 emerges. In this case the exponential growth in delay time is seen as the load progresses to a constant current model. The observation here is that constant current loads slow wave propagation more than constant admittance loads of the same size.

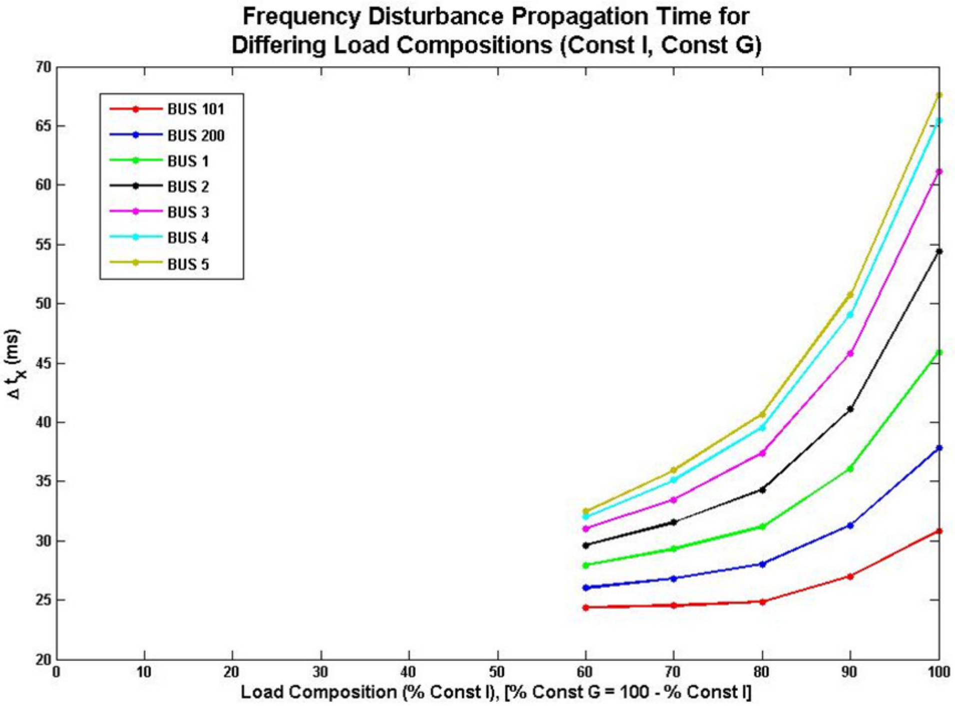


Figure 1.7: Constant Current to Constant Admittance Load Model Test Results

Throughout these load modeling simulations it was observed that the induced electromechanical wave propagated quickest across loads which were predominantly constant admittance. The induced waves propagated slower through constant current loads and finally they moved the slowest through constant power loads. These observed results are expected based on the nature of the simple load models. The power drawn by a constant admittance load is proportional to the square of the local bus voltage. This dependency implies that the system power flow magnitude will drop in response to a depressed voltage. These reduced power flows will then decrease the propagation delay times as observed in the line length tests. As the loads are changed to constant current models the power is now proportional to the voltage instead of its square. Because of this the power flows are less sensitive to system conditions and hinder the travelling wave. Finally the constant power loads are completely insensitive to the system voltage and the power flows remain constant across time. The insensitivity here has the effect of slowing the electromechanical wave.

1.5 PRELIMINARY INERTIAL EFFECTS STUDY

The second system used to conduct this qualitative testing was designed to demonstrate the effects of adding rotating machines into the medium through which the electromechanical wave propagates. With the introduction of this inertia into the system the wave should move noticeably slower than those of the preceding simulations. As a result this system is more typical of the bulk transmission system of interest.

The test system for these studies was built using the SimPowerSystem package within MATLAB's Simulink. The implemented test system consisted of a single string of busses connected to a generator, similar to the structure of the previous PSS/E simulations. The major difference here is that in addition to the static loads a small generator was placed on each bus. The events for this system were initiated by switching the main generator with a three phase breaker. The basic system architecture for this system is given in Figure 1.8. This architecture was designed to be easily extensible to larger numbers of load busses as denoted by the ellipsis on the left hand side of Figure 1.8. The test system could be enlarged with the introduction of additional copies of the basic load bus. For the majority of the testing detailed here 10 load busses were utilized within this structure.

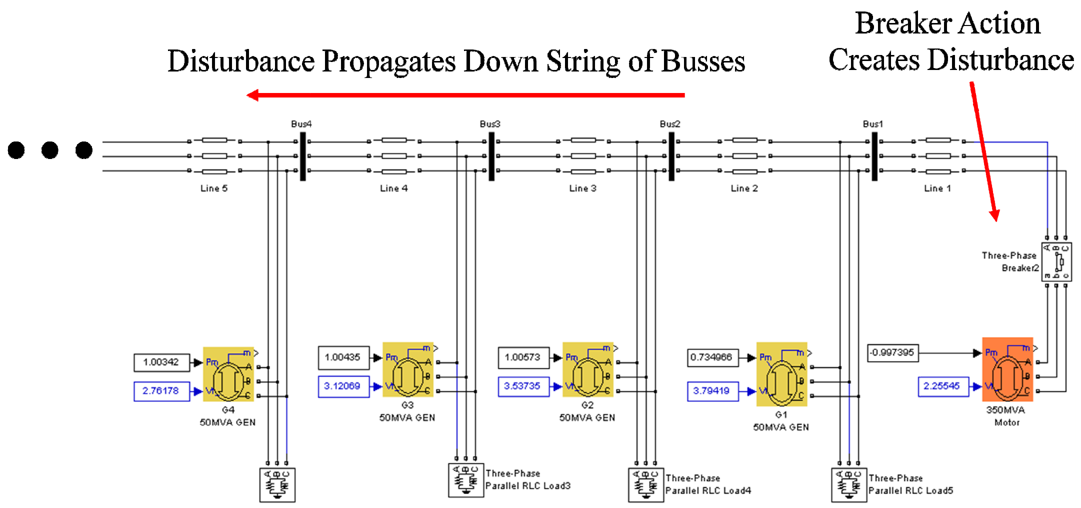


Figure 1.8: System Used for SimPowerSystem Based Simulations

As before a series of simulation were performed for each system parameter under test. Between each successive simulation that parameter would be modified. The wavefront arrival times for each of these simulations would be determined in order to assess the effect of that parameter. The wavefront arrival times for these cases were determined from the maximal frequency excursion point at the bus. This method was used as opposed to the static threshold because this test system demonstrated a more distinct response due to load inertias. The method for determining wavefront arrival is demonstrated in Figure 1.9. Figure 1.9 also demonstrates a typical response characteristic for this system. Immediately after the breaker action at 0.5 seconds the electromagnetic portion of the system response is seen as a frequency spike. The longer term electromechanical response is seen in the slower frequency swings at each bus. The magnitude of these responses decreases with distance as indicated by the smaller deviations on the more remote busses. Additionally the distance also effects the duration of the disturbance with the busses further away seeing a longer disturbance time frame.

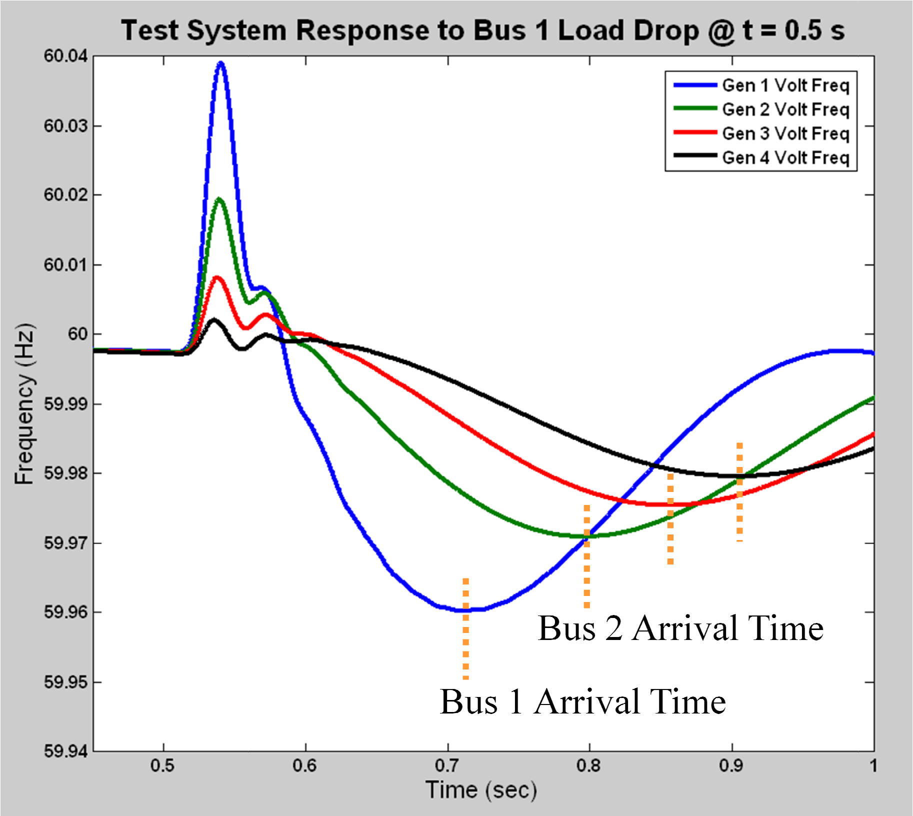


Figure 1.9: Typical Response of SimPowerSystem Test System

1.6 EFFECTS OF INERTIA – SIMPOWERSYSTEMS TESTING

The inertias of the load machines were varies over a series of simulations to observe the effects on wave propagation in the same fashion as the testing detailed in Sections 1.2, 1.3, and 1.4. The machines inertias were increased from 1 per unit up to 16 per unit throughout the simulation set. These values span the typical range for real synchronous machines of 2.5 per unit to 10.0 per unit [1]. Determining the wavefront arrival times for each simulation as per Figure 1.9 gives the propagation delays of Figure 1.10. This figure is constructed in a similar fashion to those from the PSS/E system tests with the abscissa giving the machine inertias and the ordinate is the associated bus propagation delays.

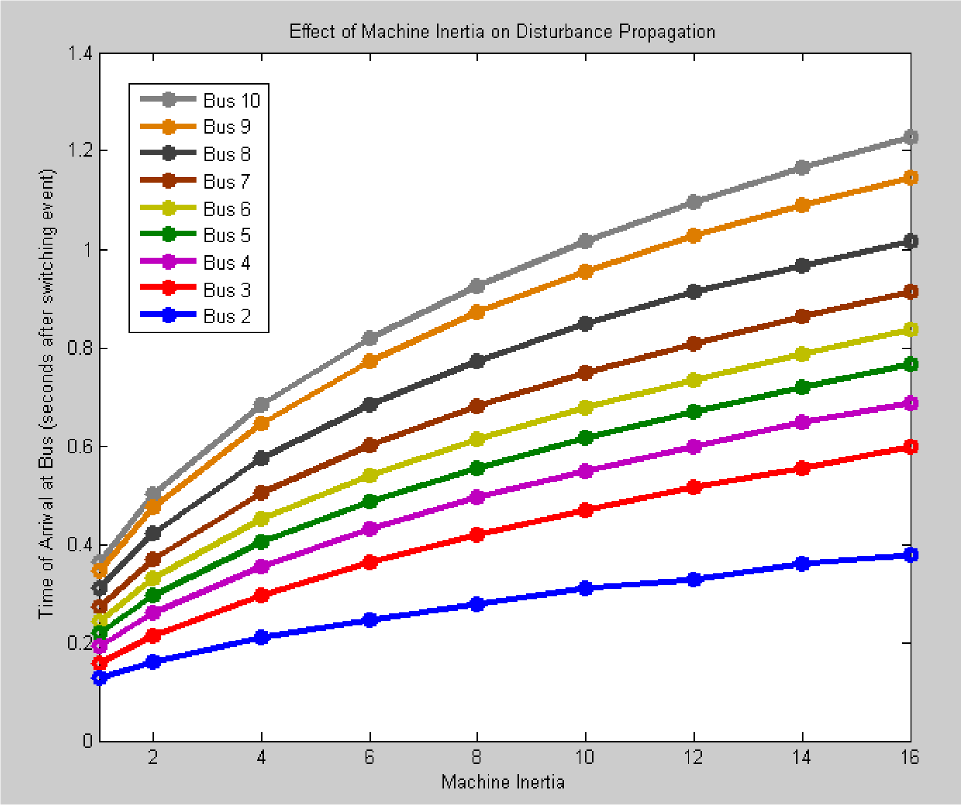


Figure 1.10: Inertia Testing Results

From Figure 1.10 is seen that as the inertia increases the traces for the individual busses spread apart. The trend shown here indicates that the wavefront slows as it moves past a point containing inertia. If a larger amount of inertia is encountered then the delay incurred increases. However this does not appear to be a directly linear relationship as seen in the non-linear traces of Figure 1.10. By fitting a few types of these curves it was determined that they approximately follow a square root trend, indicating that the speed of the advancing wavefront is proportional to the square root of the amount of inertia it is passing this result is predicted in previous studies [2]. Once again the largest delays are seen by the first few busses on the string indicating an additional dependency on the system power flows. This was confirmed through observation of the metered powered flow values. All of these results agreed closely with the expected system response.

1.7 EFFECTS OF LOAD POWER FACTOR – SIMPOWERSYSTEMS TESTS

The previous power factor testing performed on the PSS/E test system indicated that it had an effect on the delay time through altering the system reactive power flows. The structure of the PSS/E test system encountered voltage stability problems which caused failure of convergence for many of the simulations. Without these simulations the amount and quality of the obtained results was reduced. These stability issues were a result of trying to push too much reactive power down the feeder which had no reactive support (see Figure 1.1). Since the SimPowerSystems model contained synchronous machines near each load a more complete set of results could be obtained.

For this series of simulations the power factor of the static loads was varied from purely lagging through unity to purely leading while holding the total apparent power constant. Figure 1.11 plots these results with purely inductive loads on the left hand side of the abscissa, purely real loads in the center and purely capacitive loads on the extreme right. Although this system offered increased stability over the range of tests a few busses failed to produce a minimum point which is the reason for the missing data points for Busses 8, 9, and 10 over the capacitive power factor range.

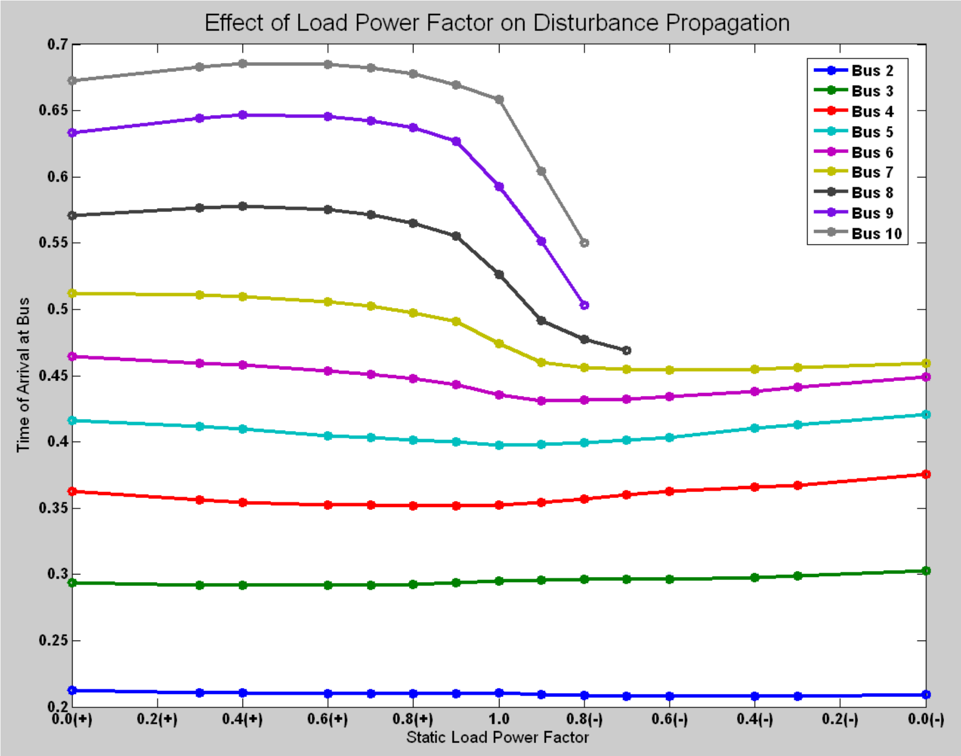


Figure 1.11: Results of Static Load Power Factor Tests

Figure 1.11 shows that the first few busses in the system show remarkably little dependence on the load power factors as they have nearly horizontal traces indicating no differences in delay time across the range of power factor values. Conversely, the busses located further away from the initial disturbance did demonstrate a change in delay as a result of the power factor modification. Over the range of inductive loads they showed larger delays than for capacitive loads. This difference is more extreme the further the bus is located from the disturbance.

1.8 FINAL OBSERVATIONS ON PRELIMINARY TESTING

The various simulations discussed in this chapter helped to build up a basic knowledge of electromechanical waves and how they move through a power system. The effects of various load characteristics, machine inertia and line impedance were observed. However, there are several features of interest which cannot be addressed by these systems.

The first major consideration here is the scale of the system. The two test systems detailed here are very small, consisting of only a few busses each. As the ultimate goal is the assessment of traveling waves in very large systems consisting of hundreds or thousands of elements these specific results are not directly applicable. Creation of an appropriate large scale test system within either PSS/E or SimPowerSystems is extremely time intensive both in terms of creating the system model and creating an initially stable simulation start point. Using an existing stable model with a hundred of busses also introduces problems as it would have non-uniform distribution of generation, load, and transmission lines, making it impossible to isolate one specific class of parameters as was done with the preceding examples.

Additionally the simulation packages used model many physical properties of power systems and control systems which have little to no effect on the propagation phenomenon under study. Removal of some of these properties should serve to improve system stability issues over a wide range of operating conditions and improve the required computation times.

For these reasons it was decided to develop and independent power system simulation package centered on a few physical equations. This simulator would be focused on a uniform easily scaleable system capable of responding to an event with the electromechanical waves of interest. This development and initial results are detailed in the following chapter.

2 DEVELOPMENT OF A SIMPLIFIED POWER SYSTEM DYNAMIC MODEL

The preceding modeling in PSS/E and SimPowerSystems has demonstrated the effects of inertia, damping, and line impedance on the propagation of an electromechanical disturbance through a power system. However these test systems are inherently limited in size. Additionally these models introduce added complexity into the system that has little to no effect on the electromechanical disturbances discussed here. In order to develop a simple, easily scalable model the relevant dynamic equations were modeled in a MATLAB code environment. This development is detailed in the ensuing chapter.

The simplified dynamic model developed centers around a state space representation of the swing equation for each machine in the system. These individual swing equations are linked together using DC power flow equations which tie the individual busses together. The resulting system model is able to capture the wave propagation effect as a function of the system parameters. Using the DC power flow helps to reduce the complexity of the system, providing a simpler test bed in which to investigate the effects of impedance, inertia, and damping on the propagation of these waves.

2.1 STATE-SPACE REPRESENTATION OF SWING EQUATION

Electromechanical waves propagate through a power system as a result of a power mismatch at a generator, which causes a rotor angle changed as determined by the swing equation. The swing equation results from a straightforward development of Newton’s second law of motion. This development is presented in several common power systems references. This development is performed by [1] resulting in the form of (2.1).

$$\frac{2H}{\omega}\frac{\partial^2 \Delta\delta}{\partial t^2} = P_M - P_E - K_D \Delta\omega \tag{2.1}$$

The per unitized swing equation including the effects of damping is given by (2.1). Where H, ω, and K_D are the generator inertia, angular speed, and damping factor respectively. P_M is the generator input mechanical power and P_E is the output electrical power. Finally $\Delta\delta$ and $\Delta\omega$ are the change in machine angle and change in machine speed. This equation represents how the rotor angle of a generator responds to a mismatch in mechanical and electrical power. For a power mismatch the right hand side of the equation will be non-zero, which will cause acceleration proportional to the machine inertia and speed.

The machine angle is analogous to a position and as such its first derivative is equivalent to the angular speed of the machine. Substituting the change in angle and speed into this gives (2.2).

$$\omega = \frac{\partial \delta}{\partial t} \quad \Rightarrow \quad \Delta\omega = \frac{\partial \Delta\delta}{\partial t} \tag{2.2}$$

Substituting (2.2) into the swing equation of (2.1) gives (2.3).

$$\frac{2H}{\omega}\frac{\partial \Delta\omega}{\partial t} = P_M - P_E - K_D \Delta\omega \tag{2.3}$$

Assigning $\Delta\delta$ and $\Delta\omega$ as state variables and isolating the derivative terms gives the state equations for a generator's dynamic response. These equations are given in (2.4).

$$\begin{aligned} \frac{\partial \Delta\omega}{\partial t} &= \frac{\omega}{2H}\left(P_M - P_E - K_D \Delta\omega\right) \\ \frac{\partial \Delta\delta}{\partial t} &= \Delta\omega \end{aligned} \tag{2.4}$$

Assuming that a machine starts from a steady-state value of rated speed the speed at any given time will be given by the rated speed added to the speed deviation value.

$$\omega = \omega_0 + \Delta\omega \tag{2.5}$$

Where ω_0 in (2.5) is the rated speed of the machine. Substitution of (2.5) into (2.4) gives (2.6).

$$\begin{aligned} \frac{\partial \Delta\omega}{\partial t} &= \frac{\omega_0 + \Delta\omega}{2H}\left(P_M - P_E - K_D \Delta\omega\right) \\ \frac{\partial \Delta\delta}{\partial t} &= \Delta\omega \end{aligned} \tag{2.6}$$

2.2 DISCRETIZATION OF STATE-SPACE EQUATIONS

Since the final goal of this development is to produce a simulation model which captures the electromechanical phenomenon of a power system, the preceding swing equation needs to be represented in a discrete-time domain to facilitate computer simulation. This can be accomplished by substituting the discrete representation of a derivative operator. The required representation is a common, straightforward development, which is repeated here for completeness [10].

The definition of a derivative is given by (2.7), the fundamental theorem of calculus.

$$\frac{\partial y(t)}{\partial t} = \lim_{h \to 0} \frac{y(t+h) - y(t)}{h} \tag{2.7}$$

Discretizing the time variable with an integer variable *n* and a sampling period of T_S results in (2.8).

$$\begin{aligned} t &= nT_S \\ \frac{\partial y(nT_S)}{\partial t} &= \lim_{h \to 0} \frac{y(nT_S + h) - y(nT_S)}{h} \end{aligned} \tag{2.8}$$

Since the time variable is now discrete the minimum possible time change is equivalent to one sampling period.

$$\begin{aligned} h &= T_S \\ \frac{\partial y(nT_S)}{\partial t} &= \lim_{T_S \to 0} \frac{y(nT_S + T_S) - y(nT_S)}{T_S} \end{aligned} \tag{2.9}$$

Converting (2.9) into the standard bracket notation for discrete functions gives (2.10).

$$\begin{aligned} \frac{\partial y(nT_S)}{\partial t} &= \lim_{T_S \to 0} \frac{y((n+1)T_S) - y(nT_S)}{T_S} \\ \frac{\partial y[n]}{\partial t} &= \lim_{T_S \to 0} \frac{y[n+1] - y[n]}{T_S} \end{aligned} \tag{2.10}$$

Thus the derivative of a discrete function may be approximated by the difference of two consecutive samples divided by the size of the sampling period. This approximation improves as the sampling period decreases as denoted by the limit operator.

Expressing the state-space swing equation of (2.6) in a discrete form gives (2.11).

$$\begin{aligned}\frac{\partial \Delta\delta[n]}{\partial t} &= \Delta\omega[n] \\ \frac{\partial \Delta\omega[n]}{\partial t} &= \frac{\omega_0 + \Delta\omega[n]}{2H}\left(P_M[n] - P_E[n] - K_D\Delta\omega[n]\right)\end{aligned} \tag{2.11}$$

Assuming a sufficiently small discrete time step the derivative operators of (2.11) can be substituted for (2.10), resulting in (2.12).

$$\begin{aligned}\frac{\Delta\delta[n+1] - \Delta\delta[n]}{T_S} &= \Delta\omega[n] \\ \frac{\Delta\omega[n+1] - \Delta\omega[n]}{T_S} &= \frac{\omega_0 + \Delta\omega[n]}{2H}\left(P_M[n] - P_E[n] - K_D\Delta\omega[n]\right)\end{aligned} \tag{2.12}$$

Solving (2.12) for the $n+1$ terms yields (2.13).

$$\begin{aligned}\Delta\delta[n+1] &= T_S\Delta\omega[n] + \Delta\delta[n] \\ \Delta\omega[n+1] &= T_S\,\frac{\omega_0 + \Delta\omega[n]}{2H}\left(P_M[n] - P_E[n] - K_D\Delta\omega[n]\right) + \Delta\omega[n]\end{aligned} \tag{2.13}$$

The representation of the swing equation in (2.13) gives a form that can be easily used with computer simulation architectures to solve for the speed and angle of a machine. At the current timestamp, *n*, all of the machine parameters and inputs are known including the simulation time step, T_S, rated speed, ω_0, present speed deviation, $\Delta\omega[n]$, present mechanical power, $P_M[n]$, present electrical power, $P_E[n]$, machine damping, K_D, and present angle deviation, $\Delta\delta[n]$. Using all of these values for the current time value the machine speed deviation and machine angle deviation of the next time step can be determined after which the simulation may progress onto the next time step and the process is repeated. Using this methodology the machine state variables maybe computed over the entire time span of a simulation.

Formulating the state-space expression of the swing equation for a system model with *m* machines gives the matrix form of (2.14) and (2.15) that will be utilized by the final simulator architecture to solve for the state variables for each machine in the system.

$$\begin{bmatrix} \Delta\delta_1[n+1] \\ \Delta\delta_2[n+1] \\ \vdots \\ \Delta\delta_m[n+1] \end{bmatrix} = T_S \begin{bmatrix} \Delta\omega_1[n] \\ \Delta\omega_2[n] \\ \vdots \\ \Delta\omega_m[n] \end{bmatrix} + \begin{bmatrix} \Delta\delta_1[n] \\ \Delta\delta_2[n] \\ \vdots \\ \Delta\delta_m[n] \end{bmatrix} \qquad (2.14)$$

$$\Delta\boldsymbol{\delta}[n+1] = T_S\Delta\boldsymbol{\omega}[n] + \Delta\boldsymbol{\delta}[n]$$

$$\begin{bmatrix} \Delta\omega_1[n+1] \\ \Delta\omega_2[n+1] \\ \vdots \\ \Delta\omega_m[n+1] \end{bmatrix} = T_S \begin{bmatrix} \frac{\Delta\omega_1[n]+\omega_0}{2H_1} & 0 & \cdots & 0 \\ 0 & \frac{\Delta\omega_2[n]+\omega_0}{2H_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\Delta\omega_m[n]+\omega_0}{2H_m} \end{bmatrix} \left(\begin{bmatrix} P_{M,1} \\ P_{M,2} \\ \vdots \\ P_{M,m} \end{bmatrix} - \begin{bmatrix} P_{E,1} \\ P_{E,2} \\ \vdots \\ P_{E,m} \end{bmatrix} - \begin{bmatrix} K_{D,1} & 0 & \cdots & 0 \\ 0 & K_{D,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_{D,m} \end{bmatrix} \begin{bmatrix} \Delta\omega_1[n] \\ \Delta\omega_2[n] \\ \vdots \\ \Delta\omega_m[n] \end{bmatrix} \right) + \begin{bmatrix} \Delta\omega_1[n] \\ \Delta\omega_2[n] \\ \vdots \\ \Delta\omega_m[n] \end{bmatrix}$$

$$\Delta\boldsymbol{\omega}[n+1] = T_S\boldsymbol{\alpha}\left(\mathbf{P}_M - \mathbf{P}_E - \mathbf{K}_D\Delta\boldsymbol{\omega}[n]\right) + \Delta\boldsymbol{\omega}[n] \qquad (2.15)$$

2.3 CONSTRUCTION OF BUSSES AND POWERFLOW EQUATIONS

With the discrete state-space representation of the swing equation derived the electromechanical dynamics of a generator can be modeled. The next step in the development of the model is to define what system these generators are operating in. In order to create an easily scalable test bed useable for several different experiments it becomes advantageous to create a generic bus. All of the test systems can then be composed of several copies of this generic bus.

Each of the busses used for these test systems had one generator and one load attached directly to the bus. Additionally each bus was connected to neighboring busses through lines modeled with constant impedance. As a simplified starting point for this development DC power flows were considered. Thus a bus with two connecting lines can be represented by the diagram of Figure 2.1.

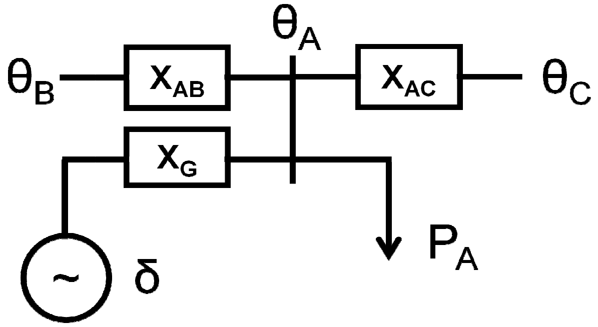


Figure 2.1: Typical Bus for Simulator

Where θ_A is the bus angle, θ_B and θ_C are the angles of the neighboring busses and δ is the generator angle. The three impedances X_{AB}, X_{AC}, and X_G are the impedance of line AB, the impedance of line AC and the generator impedance respectively. Here the load is represented as a constant power withdrawal. Summing the power flows out of bus A and setting equal to zero by the conservation of energy gives (2.16).

$$P_{AB} + P_{AC} + P_{GEN} + P_A = 0 \tag{2.16}$$

Substituting the DC power flow equations into (2.16) and treating the load as a constant power sink gives (2.17).

$$\frac{\theta_A - \theta_B}{X_{AB}} + \frac{\theta_A - \theta_C}{X_{AC}} + \frac{\theta_A - \delta}{X_G} + P_A = 0 \tag{2.17}$$

Rearranging and combining like terms of (2.17) gives (2.18).

$$P_A = -\theta_A\left(\frac{1}{X_{AB}} + \frac{1}{X_{AC}} + \frac{1}{X_G}\right) + \frac{\theta_B}{X_{AB}} + \frac{\theta_C}{X_{AC}} + \frac{\delta}{X_G} \tag{2.18}$$

Replacing the impedance terms of (2.18) with susceptance results in (2.19).

$$P_A = -\theta_A\left(B_{AB} + B_{AC} + B_G\right) + \theta_B B_{AB} + \theta_C B_{AC} + \delta B_G \tag{2.19}$$

The equation of (2.19) gives the power flow equation for bus A, which is connected to bus B and bus C. It is this form of equation that will be solved to determine the power flows through a system. By inspection the general form of (2.19) can be seen. Generalizing this equation for the ith bus in a system of n busses gives the form of (2.20).

$$P_i = \theta_i\left(-\sum_{j=1}^{n} B_{ij}\right) + \left(\sum_{\substack{j=1\\ j\neq i}}^{n} \theta_j B_{ij}\right) + \delta_i B_{ii} \tag{2.20}$$

In (2.20) P_i is the size of the constant power load at the i^{th} bus, θ_i is the bus angle of the i^{th} bus, B_{ij} is the line susceptance of the line connecting busses i and j, δ_i is the angle of the generator attached to the i^{th} bus and B_{ii} is the susceptance of the generator attached to the i^{th} bus. For this equation if there is no line connecting bus i to bus j then the susceptance B_{ij} is equal to zero, effectively removing the terms which appear in (2.20) but do not fit the form of (2.19).

Expanding the power flow equation of (2.20) to represent a system with n busses gives the matrix form of (2.21).

$$\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix} = \begin{bmatrix} -\sum_{i=1}^{n} B_{1i} & B_{12} & \cdots & B_{1n} \\ B_{22} & -\sum_{i=1}^{n} B_{2i} & \cdots & B_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n1} & B_{n2} & \cdots & -\sum_{i=1}^{n} B_{ni} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} + \begin{bmatrix} B_{11} & 0 & \cdots & 0 \\ 0 & B_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_{nn} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_n \end{bmatrix} \tag{2.21}$$

$$\mathbf{P}_{LOAD} = \mathbf{B\theta} + \mathbf{M}_B \boldsymbol{\delta}$$

Solving (2.21) for the bus angle vector gives (2.22).

$$\boldsymbol{\theta} = \mathbf{B}^{-1}\left(\mathbf{P}_{LOAD} - \mathbf{M}_B \boldsymbol{\delta}\right) \tag{2.22}$$

Expressing (2.22) in terms of the state variable *Δδ* and representing the power flow for an arbitrary time stamp, *n*, gives (2.23).

$$\boldsymbol{\theta}[n] = \mathbf{B}^{-1}\left(\mathbf{P}_{LOAD}[n] - \mathbf{M}_B\left(\Delta\boldsymbol{\delta}[n] + \boldsymbol{\delta}_0\right)\right) \tag{2.23}$$

The expression of (2.23) is used to solve the power flow in the dynamic simulator architecture. The assumptions used in deriving this necessitate a DC system, hence the voltage magnitudes are omitted and the voltage angles will be treated as state-variables and solved for across the simulation time frame. Additionally the load model used here is a constant power load, different load models maybe implemented by replacing the P_{LOAD} term with the appropriate functional description.

2.4 SIMULATION PROCEDURE

The state-space expression of the swing equation and the DC power flow equation developed above form the basis of the implemented dynamic system model. The structure of the procedure is outlined in Figure 2.2.

This procedure treats the system power injections, the load powers and the machine mechanical powers, as inputs and solves for the bus and internal machine voltage angles at each time step. The first step in this procedure is to solve the initial system power flow from the initial load power and machine power injections. This solution contains the voltage angles and provides the starting point for the simulation. This procedure is assuming that the system is initially operating in steady state and the state variables are initialized as such. The final initialization step sets up the time index, n. This variable is an integer which represents the timestamps to be simulated, for this case time steps uniformly by T_S from 0 to the specified end time. Where T_S is the user specified simulation time step. All system parameters and power variables in this system will be treated as per unit measurements. Transformer components are not considered here thus every component is considered to be on that same voltage level and thus on the same per unit base.

After these initialization steps the simulation enters the main loop which steps through time sequentially and computes the system response. In the first step of this looping process the output electrical power is computed for each machine in the system using a DC power flow formula. The computed electrical power is then used with the present state of the machine and the input mechanical power to derive the state of the machine for the next timestamp, this computation utilizes the state-space expression of the swing equation derived previously. Upon completion of this step all of the state-variables for the next timestamp are known and the time index can be incremented. At this point any disturbances maybe applied to the system. In the implemented version of the simulator these disturbances are presented by through modification of the machine mechanical powers or load powers.

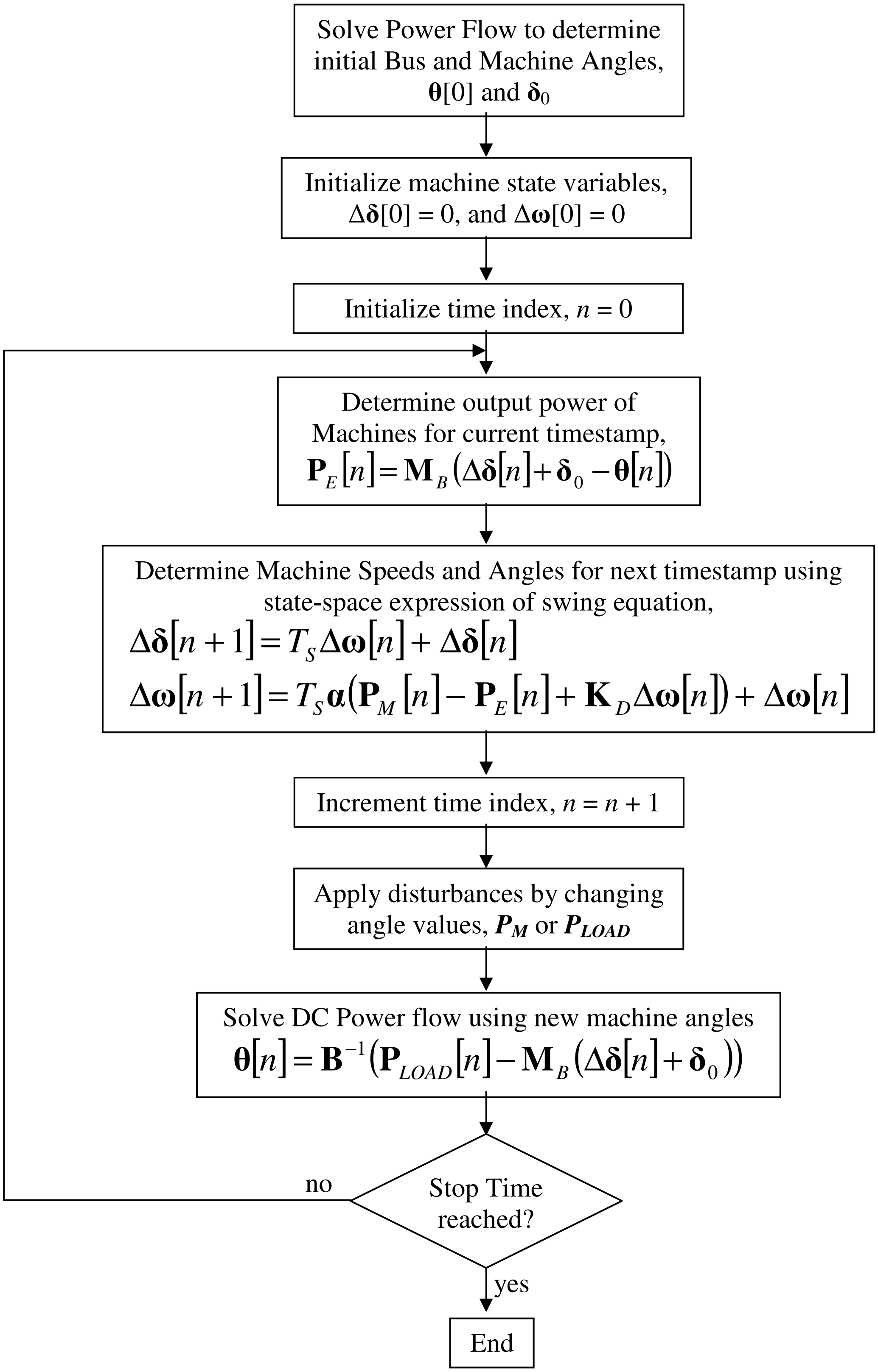

Solve Power Flow to determine initial Bus and Machine Angles, $\boldsymbol{\theta}[0]$ and $\boldsymbol{\delta}_0$
Initialize machine state variables, $\Delta\boldsymbol{\delta}[0] = 0$, and $\Delta\boldsymbol{\omega}[0] = 0$
Initialize time index, $n = 0$
Determine output power of Machines for current timestamp, $\mathbf{P}_E[n] = \mathbf{M}_B(\Delta\boldsymbol{\delta}[n] + \boldsymbol{\delta}_0 - \boldsymbol{\theta}[n])$
Determine Machine Speeds and Angles for next timestamp using state-space expression of swing equation,
$\Delta\boldsymbol{\delta}[n+1] = T_S\Delta\boldsymbol{\omega}[n] + \Delta\boldsymbol{\delta}[n]$
$\Delta\boldsymbol{\omega}[n+1] = T_S\boldsymbol{\alpha}(\mathbf{P}_M[n] - \mathbf{P}_E[n] + \mathbf{K}_D\Delta\boldsymbol{\omega}[n]) + \Delta\boldsymbol{\omega}[n]$
Increment time index, $n = n + 1$
Apply disturbances by changing angle values, $\boldsymbol{P_M}$ or $\boldsymbol{P_{LOAD}}$
Solve DC Power flow using new machine angles
$\boldsymbol{\theta}[n] = \mathbf{B}^{-1}(\mathbf{P}_{LOAD}[n] - \mathbf{M}_B(\Delta\boldsymbol{\delta}[n] + \boldsymbol{\delta}_0))$
no
Stop Time reached?
yes
End

Figure 2.2: Simulator Flow Chart

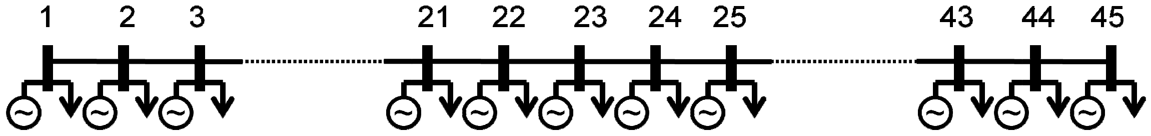
It is these disturbances that will change the system operating point and incite the type of travelling waves under investigation. The next step in the simulation procedure is to solve the DC power flow equations for the bus angles of the new time index. This is done using the power flow equations derived above along with the load powers and the machine angles that were computed from the preceding swing equation calculation. After this calculation is complete the bus angles for the current timestamp are known and the processing can return to the swing equation calculations. This looping process then continues until the required stop time is reached.

The MATLAB code implementing the developed dynamic simulator is presented in APPENDIX A; Sections I, II, III, IV, and V.

2.5 SIMULATION RESULTS

After construction of the simulation code several tests runs were performed to assess the validity of the procedure and underlying mathematical models. In general the observed simulation results closely matched the type of response seen in measurement data as well as in commercial simulation packages. The introduction of a disturbance set up a travelling wave spreading away from the inciting location. This traveling wave is easily observed as a deviation in the system bus angles from their initial values.

An example of a typical simulation will be demonstrated by the test system of Figure 2.3. This system consists of a string of 45 instances of the simulator bus, numbered 1 to 45. Each bus in this system is connected to its immediate left and right neighbors with the end busses only having one connection. The system was setup as uniform with all machines, loads and lines identically sized. The system and simulation parameters were set as per Figure 2.3, the system parameters are specified on a per unit basis.



System Parameters:		Simulation Timing:
$H = 1$	$X_L = 0.05$	$T_S = 10^{-5}$ sec
$K_D = 0.01$	$P_L = 1$	*run time*: $t = 0 \rightarrow 4$ sec
$X_G = 0.0005$	$P_M = 1$	

Figure 2.3: Radial Example System

The disturbance applied to this system came in the form of doubling the load on the middle bus (Bus 23) to 2 per unit for $0.3 \leq t \leq 0.35$ sec.

For the purpose of better demonstrating the nature of the resulting travelling wave the simulation results were expressed as an angle deviation on a bar plot. In the following figures the horizontal axis corresponds to different busses in the string. The vertical axis gives the deviation of the bus angle from its initial value in radians. Each figure represents the system at one specific timestamp during the simulation.

Immediately after the load doubling on the middle bus the first angle deviations are noticed, this case is depicted in Figure 2.4 as the middle busses in the system begin to respond to the load change. As time progresses the traveling wave begins moving out from the center bus toward the edges of the system. This moving wavefront can be observed in the transition from Figure 2.4 to Figure 2.5. The wavefront hits the boundaries of the system by Figure 2.6 and begins to reflect back towards the center of the system as seen in Figure 2.7. This process continues through Figure 2.8 as the wavefront reaches the middle of the system again. The wave continues reflecting back and forth across the length of the system until the damping effects in the generators finally attenuate the wave and the system reaches the steady state of Figure 2.9.

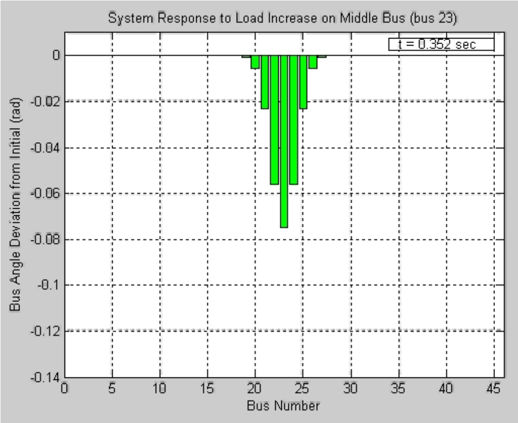


Figure 2.4: Radial System Capture 1

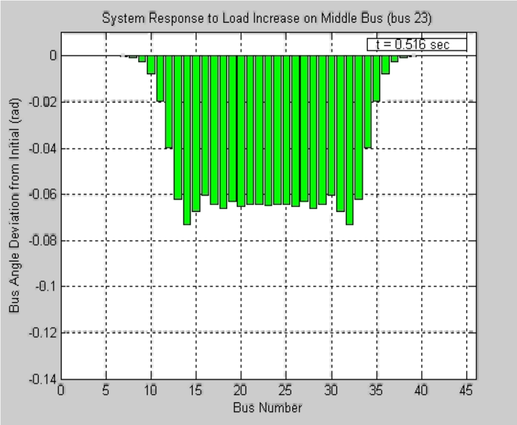


Figure 2.5: Radial System Capture 2

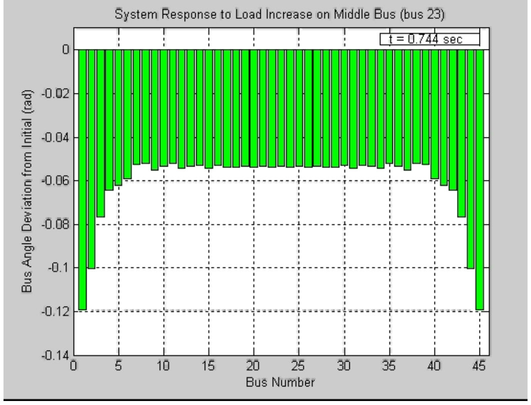


Figure 2.6: Radial System Capture 3

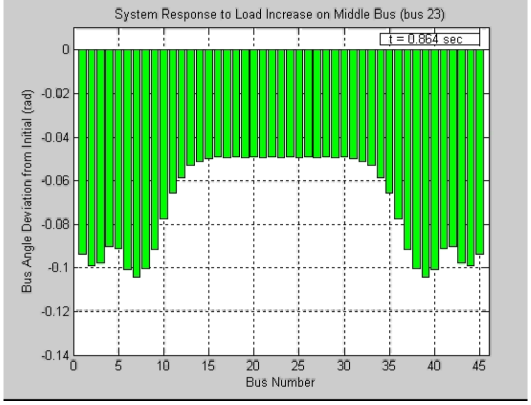


Figure 2.7: Radial System Capture 4

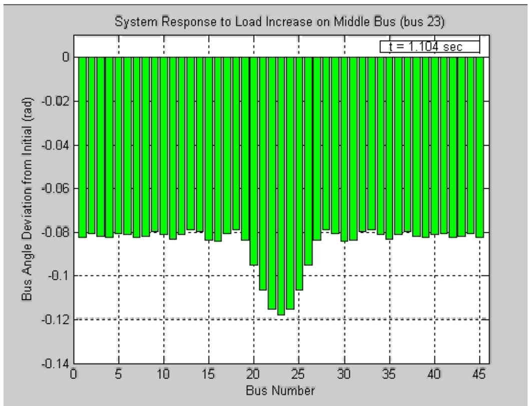


Figure 2.8: Radial System Capture 5

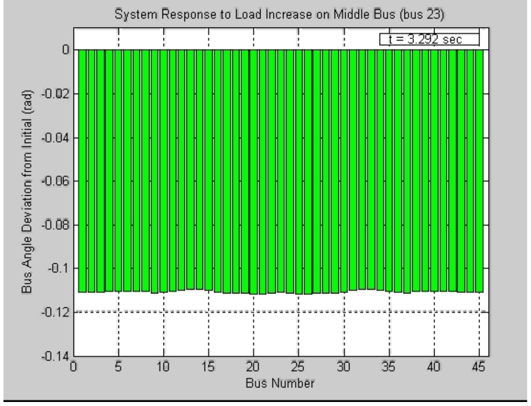


Figure 2.9: Radial System Capture 6

2.6 PROPAGATION EFFECTS IN TWO DIMENSIONAL SYSTEMS

The simulation discussed above involved a completely uniform system with an even distribution of machine inertia, machine damping, and line impedance. This style of system produced a traveling wave that propagated at the same speed in all directions. As noted in previous studies all of these factors contribute to the speed of wave propagation. As a system deviates from non-uniformity some areas will have a higher concentration of these parameters and it will be lower in other areas. With these areas of different concentration it is expected that the propagation speeds will differ as the wave moves along differing paths.

An example of this effect is demonstrated by the results of another simulation performed using the developed simulation tool. The test system consisted of 625 busses connected by lines to form a regular 25 by 25 grid with a bus at each vertex connected by lines to the neighboring 4 busses. For this simulation the busses and lines were setup with the following parameters (all units on a per unit basis, unless otherwise noted).

System Parameters:		Simulation Timing:
$K_D = 0.01$	$X_L = 0.05$	$T_S = 10^{-5}$ sec
$X_G = 0.0005$	$P_L = 1$	*run time*: $t = 0 \rightarrow 4$ sec
$P_M = 1$		

In order to introduce non-uniformity into the system the machine inertias were initialized differently. The machines tied to busses in the upper 10 by 10 portion of the grid were given inertia of 5 per unit. The inertia on the remainder of the machines in the system was set to 1 per unit. The disturbance applied to this system came in the form of doubling the load on the middle bus (Bus 313) to 2 per unit for $0.3 \leq t \leq 0.35$ sec.

As before the plots used to demonstrate the traveling wave consist of bar plots representing the bus angle deviation from initial. Since the system is now two dimensional the differing busses are expressed in the XY plane with the angle deviation values expressed in the Z dimension. The plotted angle values are -1 times the true value so that they can be seen easier. The differing bar colors is to help differentiate neighboring busses; this color does not represent anything other than position. Finally the area of higher machine inertias is shaded with orange.

Following the inciting event at the center bus first angle deviations are noticed, this case is depicted in Figure 2.10, the central busses see the disturbance and it begins to propagate outwards. The moving wave front can be observed in the transition from Figure 2.10 to Figure 2.11. These two figures also demonstrate the effect of increased inertia as the wavefront slows when entering the area with higher inertias. This effect is demonstrated further with the transition from Figure 2.11 to Figure 2.12. The portion of the wave that travelled through the higher inertia has been noticeably slowed. Additionally the propagation paths that moved the greatest distance through the high inertia area are slowed the most, producing the concave wavefront shape. The effects of system parameters and non-uniformity will be investigated in greater detail in Chapter 3.

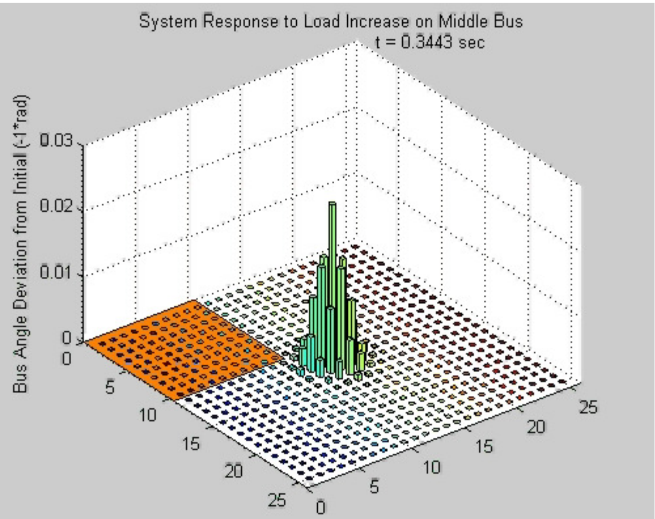


Figure 2.10: Non-Uniform Grid System Capture 1

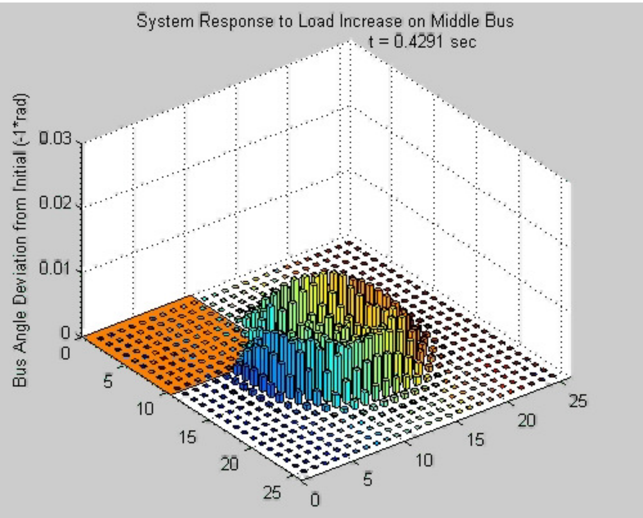


Figure 2.11: Non-Uniform Grid System Capture 2

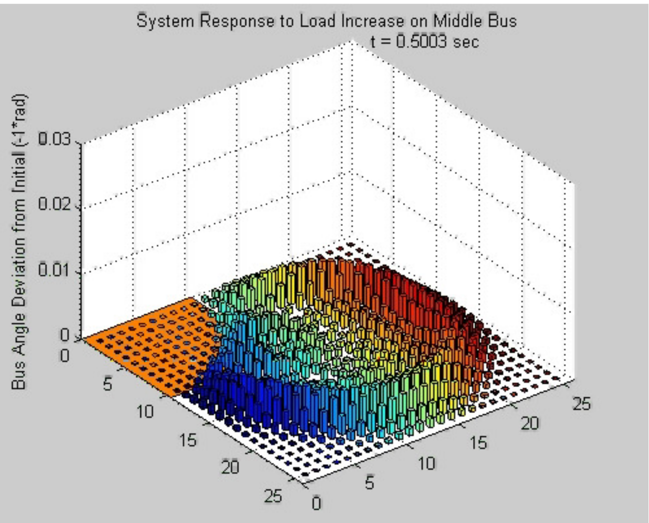


Figure 2.12: Non-Uniform Grid System Capture 3

3 EFFECTS OF SYSTEM PARAMETERS ON TRAVELING WAVES

Travelling electromechanical waves will now be investigated in depth using the power system simulator developed in Chapter 2. In particular the effect of individual system parameters on these waves will be explored. Through several simulations the dependence of a traveling wave on inertia, impedance and damping will be observed. It is desired to find a qualitative relationship between each of these parameters and the resulting traveling wave characteristics.

3.1 SYSTEM RESPONSE AND CHARACTERISTICS OF THE RESULTING WAVE

As a start point for these simulations a base case was established. This base case system would define the test bed for the wave propagation testing. All of the subsequent simulations involve the same system structure with differing values for the system parameters. As each parameter is varied individually the effects on wave propagation speed and magnitude will be measured.

The base case system consists of a string of busses similar to that of Figure 2.3. The system in this case is extended to 151 busses to provide a larger set of busses to sample and a wider region before reflection effects are encountered. As before the inciting event will occur at the central bus (bus 76 in this case) so as to produce symmetrical wave propagation. Otherwise the base case is identical to the system simulation and results described in Section 2.5. As before the machine damping values are being set artificially low in order to demonstrate the reflection effects of the travelling waves.

A simulation result for this base case is given in Figure 3.1. If plotted as the bar plots of Figure 2.4 through Figure 2.9 this simulation would look similar but have a wider span due to the larger number of busses. Figure 3.1 gives the bus angle deviation as it evolves over time for three selected busses. The terms used throughout the remainder of this chapter will be defined with respect to the features of this plot. Wavefront arrival at a point in the system will be defined as the time point at which a maximal excursion in bus angle is seen. In the simulation the load on bus 76 is doubled to 2 per unit from t = 0.3 sec to t = 0.35 sec, this action produces the inciting event for the travelling wave. As a result the bus angle of bus 76 begins to move in response to the power mismatch, this bus is the first to see the wavefront as it is where the inciting event occurs.

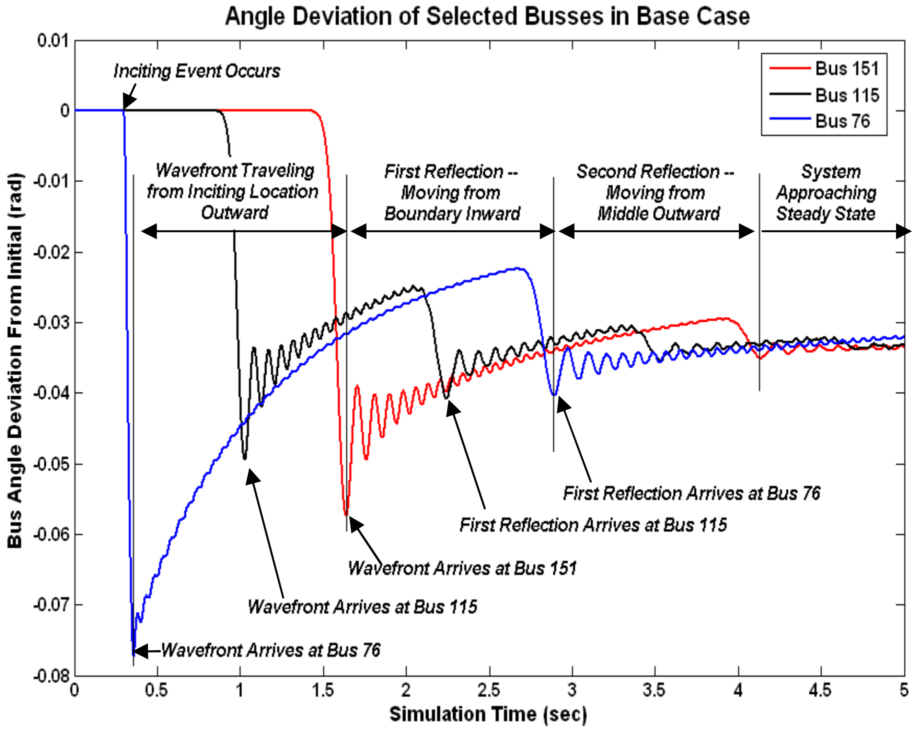


Figure 3.1: Electromechanical Wave Propagation through Base Case System

As with the radial example of Section 2.5 the wavefront travels outward from the middle bus. This region is labeled as such in Figure 3.1. The majority of the busses are omitted from the plot for clarity but this progression can be seen by the succession of wavefront arrivals from bus 76 to bus 115 to bus 151. All of the other bus angle arrivals fall within this span in order when plotted. Once the boundary of the system is reached at bus 151 the wave reflects back toward the center. Figure 3.1 shows this in the span labeled as '*First Reflection*', in this region the arrival times are seen in reverse order as the wave is moving back toward the system center. Due to the light damping of this base case a second reflection can be clearly seen in which the wave propagates from the center outward once again. After this the damping effects reduce the wave magnitude and the system approaches its final steady state value.

The progression of Figure 3.1 gives the standard to which further simulations will be compared. As system parameters are modified the effects on each of the labeled regions will be assessed. In particular the size of the induced wave and the propagation time will be compared to the base case for various changes in machine inertia, machine damping and line impedances.

3.2 EFFECTS OF MACHINE INERTIA ON WAVE PROPAGATION

Several simulations were performed to investigate the effects of machine inertia on the magnitude and speed of the induced electromechanical wave. The majority of these involved permutations of the base case. Holding everything else in the system constant the machine inertias were varied across each simulation. Thus through comparison of the individual results the inertial effects could be assessed.

The inertia of a machine appears within the swing equation and as such it will introduce a dependency in the resulting system response. The swing equation is given by (3.1) as discussed in Section 2.1.

$$\frac{2H}{\omega}\frac{\partial^2 \Delta\delta}{\partial t^2} = P_M - P_E - K_D \Delta\omega \tag{3.1}$$

Rearranging (3.1) to isolate the machine angle deviation, $\Delta\delta$, gives (3.2).

$$\frac{\partial^2\Delta\delta}{\partial t^2}=\frac{\omega}{2H}\left(P_M-P_E-K_D\Delta\omega\right) \tag{3.2}$$

The propagating electromechanical wave is seen as a deviation in angle. Thus a larger change in angle would produce a more pronounced wave. The left hand side of (3.2) is a second derivative and thus gives the acceleration in machine angle. A larger value in the right hand side of (3.2) would then induce a greater acceleration in machine angle. Since the inertia, *H*, appears in the denominator in the right of (3.2) a larger inertia value would serve to limit the acceleration and thus the change in angle. Conversely, smaller machine inertia would serve to increase the angle deviation and thus the result wave. This is as expected in a physical sense as a higher inertia machine would be less responsive to changes, reducing the effects of a disturbance and the resulting traveling wave.

Figure 3.2 presents the results from these inertia tests. Three simulations are represented in this plot, the base case, a simulation with the machine inertias set to 2 per unit, and a simulation with the machine inertias set to 5 per unit. As expected the simulations for the higher inertia cases produced a slower moving wavefront. The base case with the machine inertias set to 1 per unit initiated a wave that took 1.28 seconds to propagate from the middle of the bus string to the edges. When the machine inertias were doubled to 2 per unit this propagation time increased to 1.82 seconds. Finally for the 5 per unit simulation the wave took 2.87 seconds to reach the edge of the system. These propagations delays are demonstrated for the three simulations of Figure 3.2 by the labeled time spans, which are defined by the inciting event and the wavefront arrival at the final bus, Bus 151.

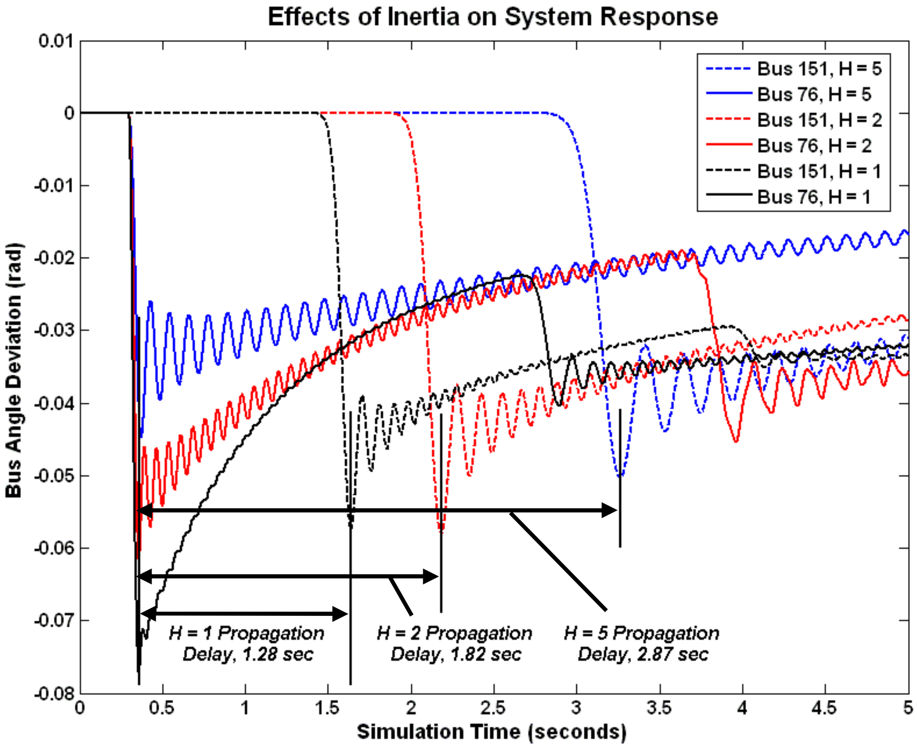


Figure 3.2: Effects of Machine Inertia on the Base Case Simulation

These inertia tests also produced some effects on the wave magnitude as seen in Figure 3.2. The simulations with machines that had lower inertia induced a lower magnitude traveling wave at the point of the event. As seen in Figure 3.2 the base case demonstrated a -0.077 rad deviation at the center bus in response to the doubling of the load. The 2 per unit and 5 per unit simulations induced initial bus angle deviations of -0.062 rad and -0.045 rad respectively. This is as expected as the higher inertia machines are less responsive and result in smaller swings. An interesting note here is that although the higher inertias produced less of an initial disturbance, they did demonstrate higher oscillation and reflection magnitudes than the base case. This effect is seen in Figure 3.2 with the higher oscillation magnitudes for the 2 per unit and 5 per unit responses on bus 76. In the end all of these simulations settled out the same final angle deviations, this is as expected because they were all incited by identical disturbances and thus had the same power mismatch over the same period of time.

The preceding example simulations demonstrated the effect of machine inertia on the speed at which an electromechanical wave propagates. As observed the wave moves slower through an area with a higher density of inertia per unit area. Several additional simulations were performed to build a better understanding of this effect in more complex system architectures.

Figure 3.3 presents the results of one of these simulations. Using the simulator detailed above an event was incited by doubling the load on the middle bus of a 25 by 25 meshed grid for $0.3 \leq t \leq 0.35$ sec. For this case the machine inertias in the upper left portion of the system were set to 5 times that of the rest of the system. Additionally the machine inertias in the lower right were set to one fifth that of the machines in the center. All other system parameters were set constant across the system and in line with the values used for the base case. These two areas of lower and higher inertia would qualitatively demonstrate the effects of inertia for a two dimensional system.

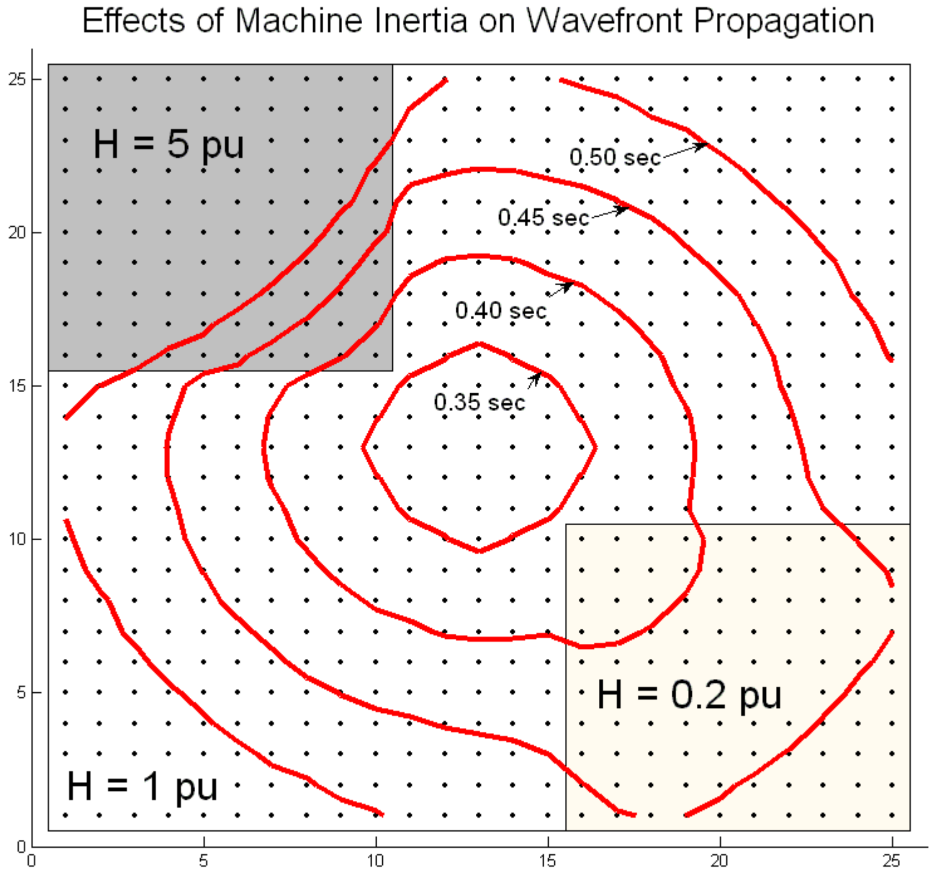


Figure 3.3: Effect of Machine Inertia on Wave Propagation in 2-D System

The axes in Figure 3.3 represent the geographic directions, with each point representing a bus, although it is not shown on this plot each bus is connected to its horizontal and vertical neighbors by identical transmission lines. In a post processing step the wave reflections and oscillations were removed numerically to avoid distorting the wavefront arrival time. The concentric rings give the points in the system that saw the wavefront at the same. As seen from the figure the higher inertia region causes the wave to be slowed, creating the convex shape in the wavefront. On the other hand the wave propagated quicker through the region of decreased inertia as demonstrated by the bulge in the wavefront. This bulge resulted from the wave covering a greater distance in the same amount of time as compared with the unaltered portion of the system.

3.3 EFFECTS OF LINE IMPEDANCE ON WAVE PROPAGATION

The effects of line impedance in the base case were assessed by another set of simulations. As with the inertia testing the parameter of interest was varies over a series of independent trials, holding all other system characteristics constant. In this case the line impedances were modified across the set of simulations. As in the preceding testing the effects of these changes are presented in a graphical nature.

Inspection of the swing equation of (3.2) shows no explicit dependence on line impedance in the given form. However the electric output power of the machine, P_E, is given by (3.3) where δ is the machine angle, θ is the local bus angle and X_G is the machine impedance.

$$P_E = \frac{\delta - \theta}{X_G} \tag{3.3}$$

If (3.3) is substituted into (3.2) the result is given by (3.4).

$$\frac{\partial^2 \Delta\delta}{\partial t^2} = \frac{\omega}{2H}\left(P_M - \frac{\delta - \theta}{X_G} - K_D \Delta\omega\right) \tag{3.4}$$

Thus the swing equation shows a dependence on the local bus angle, namely as the bus angle increases, holding everything else constant the acceleration of the machine angle will increase as well. Since the bus angle is a load flow solution it involves a dependence on the line impedances of the system. Thus the wavefront should propagate differently as the line impedance changes.

From a conceptual perspective the wave is expected to propagate slower as the line impedance increases. Two busses connected by low impedance are strongly coupled together thus a disturbance on one bus will be seen by each other. As this connecting impedance is increased the busses are increasingly isolated from each other. Thus it is expected that higher line impedances in the system will isolate the busses from each other and hinder the propagation of the wave, slowing the traveling wavefront as it moves across the system. Additionally this increased isolation should result in more extreme swings in bus angle as a disturbance is hidden from the neighboring busses, causing the local bus to absorb the power mismatch as an angle swing instead.

These hypotheses are confirmed by the simulation results summarized in Figure 3.4. The results of three simulations are presented in Figure 3.4, the base case which used line impedances of 0.05 per unit, a simulation with the line impedances modified to 0.10 per unit, and finally one with the impedances set to 0.25 per unit. As expected the propagation delay down the string of busses increases with the line impedance. The total delay down the string of busses increased from 1.28 seconds to 1.82 seconds as the line impedance was stepped up from 0.05 per unit to 0.10 per unit. The total delay increased again to 2.90 seconds when the impedance was increased to 0.25 per unit.

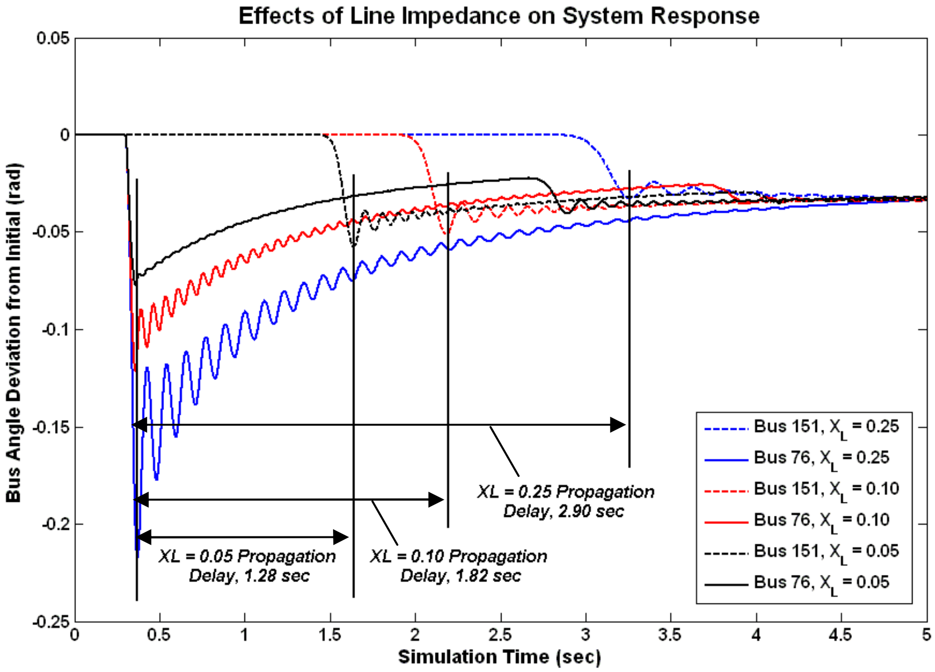


Figure 3.4: Effects of Line Impedance on Base Case Simulation

The wave magnitude also demonstrated variation through the differing simulations. The most prominent of these magnitude differences occurred on the central bus at which the load doubling event occurred. For the base case this middle bus saw a maximum angle deviation of 0.076 radians occurred in immediate response to the load change. As the line impedance increased to the 0.10 case this angle deviation increased with it to 0.121 radians. Finally for the 0.25 case the maximum deviation was 0.216 radians. This is in agreement with the hypothesis that increased line impedance serves to increase the magnitude of the angle changes as the busses become increasing isolated.

A simulation similar to the two dimensional example of Figure 3.3 was implemented to investigate the effect of line impedance on the propagation of a traveling wave in a multi-dimension case. For this simulation the impedance of the transmission lines in the upper left were set to 0.25 pu, five times that of the base value. To demonstrate the opposing effect the impedance of the lines in lower right were set to 0.01 pu, one fifth of the base value. Using the same simulation procedure and wavefront identification process as before the results in Figure 3.5 were generated.

As Figure 3.5 demonstrates increasing line impedance produces the same effect as increased generator inertia, slowing a traveling wave. As with the previous inertia test it was also noted that the size of the wave was unaffected. The opposite was shown to be true for areas of reduced line impedance. In these regions with lower impedance the wave front moved faster.

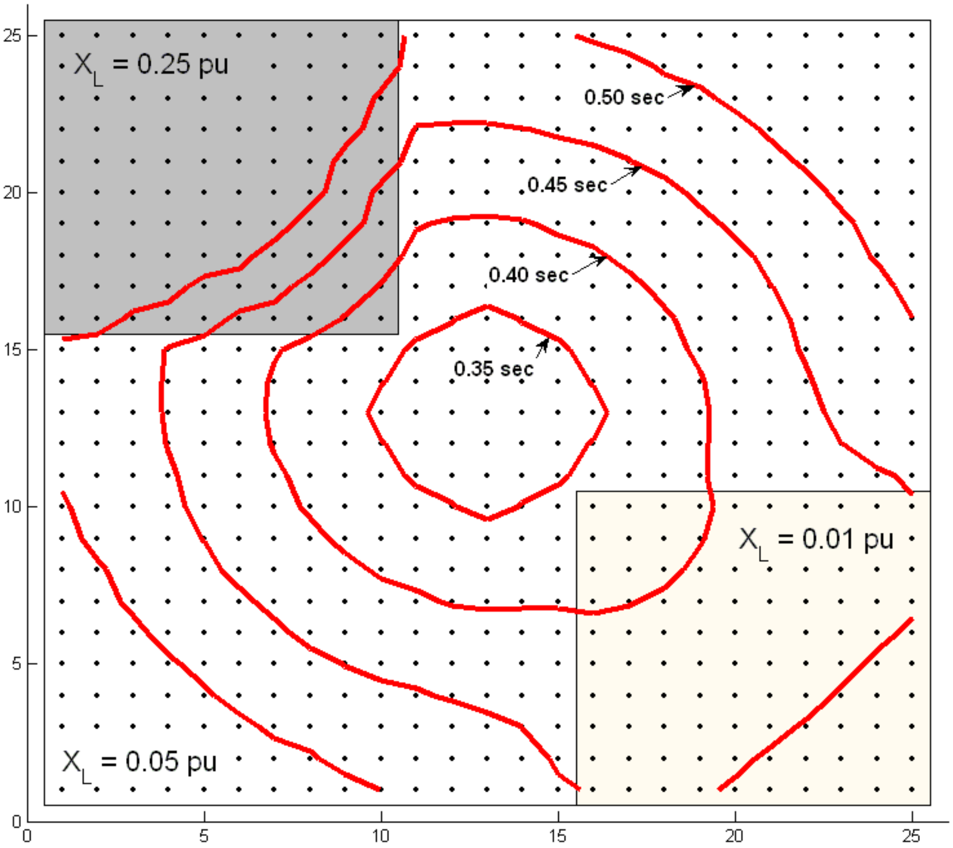


Figure 3.5: Effect of Line Impedance on Wavefront Propagation in 2-D System

3.4 EFFECTS OF MACHINE DAMPING ON WAVE PROPAGATION

The final set of simulations performed was designed to assess the effects of the machine damping parameter. Once again a series of simulations were performed varying the test parameter between each one. The test parameter in this simulation set was the machine damping factor. Figure 3.6 give the results for a selection of these tests, namely with the damping factors set to 0.01 per unit (base case), 0.02 per unit, and 0.05 per unit. For all of these cases the observed propagation delay through the system was 1.28 seconds. This indicates little to no dependence of wavefront propagation on the damping factors. The initial angle deviations at the central bus were identical across the set of simulations. However other effects were observed which served to shape the electromechanical waves. The factors that the damping did affect were the oscillatory content riding on the wave signature and the final steady state value the simulation achieved.

As expected the most prominent effect of the machine damping was reducing oscillatory content within the system response. With an increased amount of damping in the system the induced oscillations die out much quicker. This can be seen in Figure 3.6 through comparison of the 0.01 and 0.02 cases against the 0.05 case. The first two cases demonstrate a visible oscillation, especially on the first reflection of the wave off the edge of the system. The 0.05 case did not produce any pronounced oscillations in the response, with the signals approaching their final values in a purely exponential decay.

Another noteworthy difference between this set of simulations and those for the inertia and impedance test is the final steady state location of the system. Throughout all the previous testing each simulation settled to pretty much the same steady state location. For these damping tests each individual simulation approached a different steady state location as seen by the right side of Figure 3.6. This result indicates that the new operating point a system achieves after a disturbance is heavily influenced by the amount of damping in the system.

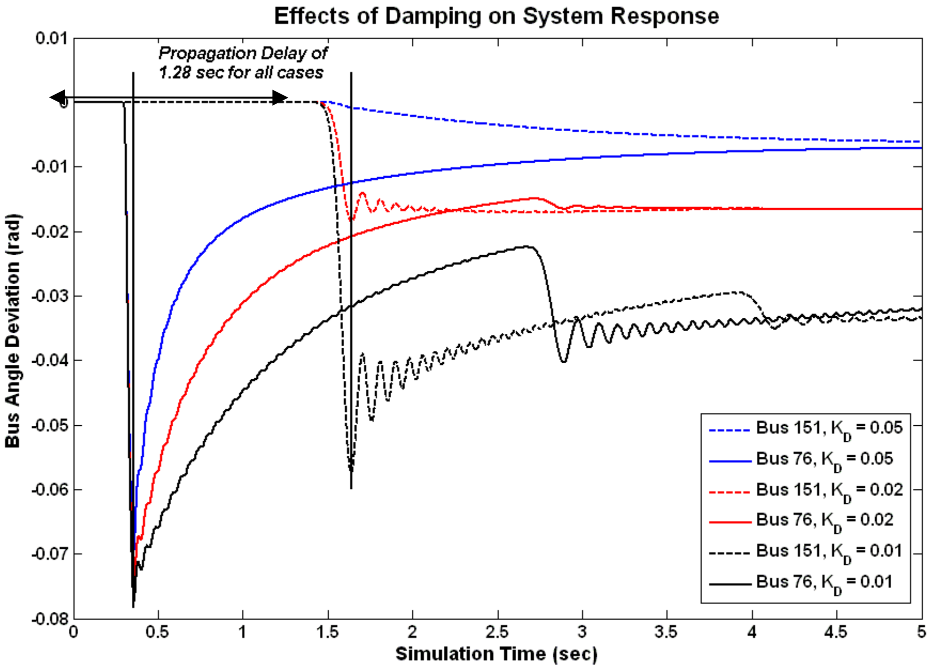


Figure 3.6: Effects of Machine Damping Factor on Base Case Simulation

3.5 CONCLUSIONS ON PARAMETER TESTING

This chapter has detailed the result from several different simulations based around the simplified dynamic model. It has been observed that increased inertia and line impedance serve to slow wave propagation while additional damping reduces the amount oscillatory content and wave reflections.

These procedures have been built on a large, completely uniform system so as to better depict the wave propagation in a spatial sense. A real world system is obviously very non-uniform with differing distributions of inertia, impedance and machine damping. This non-uniformity complicates matters with the wave propagating at differing speed in differing spatial directions. Essentially the responses of Figure 3.3 and Figure 3.5 are taken to the extreme with the regular mesh changed to a high dimension graph where geographic distance and electrical distance are not necessarily correlated. Additionally the distribution of inertia in a practical system is primarily focused at the large generation stations and as such is limited to a relatively small percentage of the system busses. Although these differences exist between a practical system and the test cases the underlying properties are similar and these observations made here are easily extensible from a conceptual perspective.

As the observation scale of a practical system increases the non-uniformity would tend to be minimized. As more and more pieces of the overall system are incorporated into the study area each individual one takes on proportionally less significance and an analogy of a continuum or high order discrete point distribution emerges [2]. Thus, when observed in a wide-area sense the system resembles something closer to uniform with pockets of higher and lower inertia and impedance. In this case the modeling here takes on added relevance.

4
DEVELOPMENT OF SYSTEM APPROXIMATION PROCEDURE FROM THE SIMPLIFIED DYNAMIC MODEL

The preceding model development will now be applied to a large scale power system in order to provide a reduced order system model. This reduced model would provide advantages in terms of system analysis and dynamic state estimation. With a sufficiently accurate reduced model the interdependencies and characteristics of the original system can be extracted. With an appropriate equivalent model for a given system determined, incoming measurement data can be processed in real time to provide an indication of the system operating point. Additionally as the system state is read in through measurement data future measurements values along the same trajectory can be estimated. If sufficiently accurate these estimates of future system values can provide information about the system state and stability and for advanced control and protection schemes.

Several types of power system reduction techniques have been proposed previously [11],[12],[13],[14]. The majority of existing reduction techniques rely on knowledge of the existing system structure including topology and electrical parameters of the major pieces of infrastructure. The system reduction is then accomplished through some form of aggregation to combine nearby components and form a lower order model. The proposal here takes a different route in that it attempts to derive the reduced model directly from wide-area measurement data without knowledge of the underlying system components. The measurements are used to fill out the parameter values of the simplified dynamic model in such a way to best match the true system.

4.1 SYSTEM REDUCTION USING THE SIMPLIFIED DYNAMIC MODEL

The basic principle of this system reduction method will center on the measurement points of the system. A measurement point in this sense will refer to a point of the system metered by a device capable of performing phasor measurements at a high time resolution, typically a Phasor Measurement Unit (PMU). Each of these points in the system would be represented by one pseudo-bus in the reduced system. These pseudo-busses are identical to the typical simulator bus of Figure 2.1 each having one generator and one load. In this formulation each pseudo-bus aggregates the surrounding area of the system. Its generator is an equivalent of the generators in its area while its load is the summation of the real loads in the represented area. The pseudo-busses are connected to each other by a set of ideal lines modeled by a series reactance as used by the simulator architecture of Chapter 2.

This system construction procedure is illustrated by the following series of figures. Suppose that it is wished to approximate the Eastern Interconnect with this simplified model architecture. Given the set of hypothetical measurements indicated in Figure 4.1 a model pseudo-bus is placed at each measurement location as shown in Figure 4.2. With each bus placed the generators and loads are sized according to the training measurement data and represent an aggregation of the surrounding system area. Next a set of lines are inserted connecting the pseudo-busses into a network as shown in Figure 4.3. The impedance of each line is then calculated to provide a best match to the training data. With the generators, load and lines sized the final reduced system model is achieved, the pictorial example this is given by **Figure 4.4**.

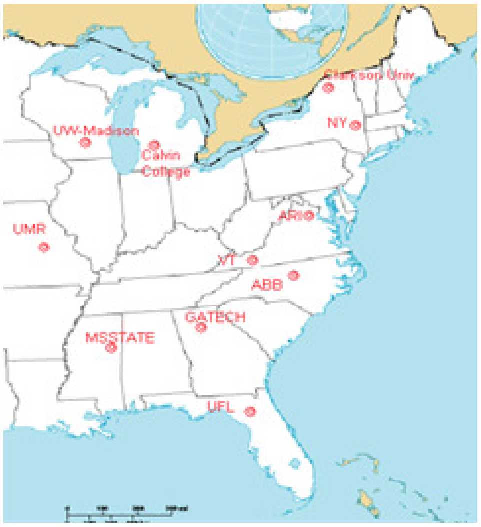


Figure 4.1: Distribution of Hypothetical Measurements

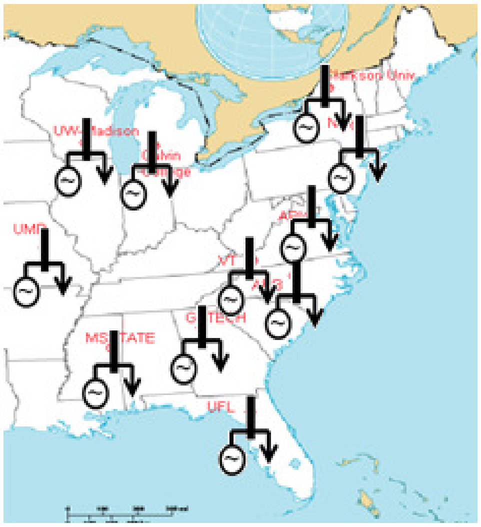


Figure 4.2: Placement of Pseudo-Busses

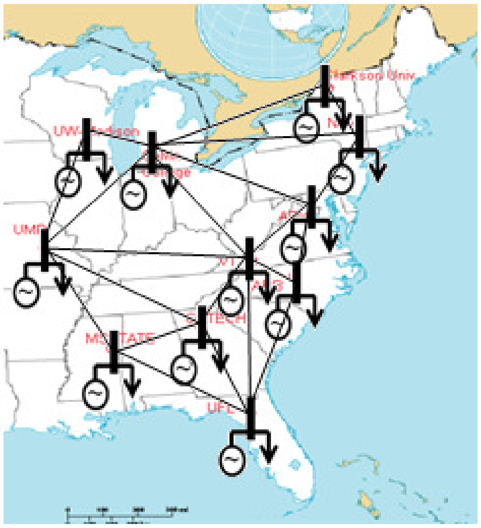


Figure 4.3: Insertion of Connecting Lines into System

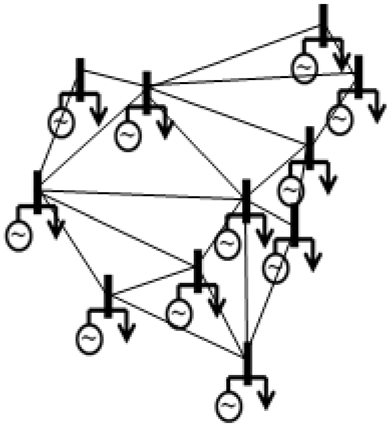


Figure 4.4: System Approximation

Given a set of high time resolution measurement points the best fit of the reduced system will be derived. This data set includes the bus angle, θ_i, generator angle, δ_i, generator mechanical power, P_{Mi}, and load power, P_{Li}, for each pseudo-bus, i, of the reduced model. The power quantities here are not directly measured but are instead the total powers in the area represented by pseudo-bus i. With P_{Mi} giving the total of the mechanical input powers to the machines in the area represented by pseudo-bus *i*. Similarly P_{Li} is the summation of the real power loads in area *i*.

The ultimate goal is to derive a reduced order system model which provides a best fit to the original measurement data. This will be implemented as a two step process. First the parameters of the generators in the reduced system model will be estimated. In the second step the necessary line impedances will be estimated. With these two results the reduced order system is completely defined and can be modeled using the simulation procedure of Figure 2.2.

4.2 ESTIMATION OF MACHINE PARAMETERS

The goal is to place a bus at each measurement point in the system. The system approximation will then be determined from a given training set of data. The system architecture in the simplified model has a generator and load at each bus. Thus a generator needs to be placed at each bus in the approximation. The parameters of each generator will be determined from measurement data contained within the training set. This training set contains the bus angle, θ, machine angle, δ, and mechanical input power, P_M, for each measurement point in the system to be approximated. From these measurement parameters the machine parameters can be estimated as outlined below.

Since the machine speed, ω, is the first derivative of machine angle the machine speed can be determined. Utilizing the discrete nature of the measurements and the fundamental theorem of calculus produces (4.1). In (4.1) as before the Δ indicates a change from initial conditions. This is the same result as derived in Chapter 2.

$$\Delta\omega[n] = \frac{\Delta\delta[n+1] - \Delta\delta[n]}{T_S} \tag{4.1}$$

Since the machine angle is a measured quantity the change in machine angle, $\Delta\delta$, is also known with the selection of a reference point as the initial condition. After determining the change in machine angle (4.1) can be used to find the change in machine speed parameter for the training set.

In the derivations of the Chapter 2 the swing equation was expressed in a discretized form. This formula is repeated below in (4.2). Additionally the electrical power of a generator is given by the difference between the machine and bus angle multiplied by the machine admittance, this formula is given by (4.3).

$$\Delta\omega[n+1] = T_S \frac{\omega_0 + \Delta\omega[n]}{2H}\left(P_M[n] - P_E[n] - K_D\Delta\omega[n]\right) + \Delta\omega[n] \tag{4.2}$$

$$P_E[n] = B_G\left(\delta[n] - \theta[n]\right) \tag{4.3}$$

Using (4.3) to replace P_E in (4.2) gives (4.4)

$$\Delta\omega[n+1] = T_S \frac{\omega_0 + \Delta\omega[n]}{2H}\left(P_M[n] - B_G\left(\delta[n] - \theta[n]\right) - K_D\Delta\omega[n]\right) + \Delta\omega[n] \tag{4.4}$$

A further simplification will be introduced to (4.4). For realistic operating cases the rated machine speed, ω_0, is much larger than and speed changes, $\Delta\omega$. Thus ω_0 dominates the sum of the two. Using this property (4.4) can be reduced to (4.5).

$$\Delta\omega[n+1] = T_S \frac{\omega_0}{2H}\left(P_M[n] - B_G\left(\delta[n] - \theta[n]\right) - K_D\Delta\omega[n]\right) + \Delta\omega[n] \tag{4.5}$$

In order to determine the equivalent machine for a bus the parameters need to be estimated. In order to accomplish this, values of inertia, H, admittance, B_G, and damping, K_D

which provide a best fit to the training data are calculated. Rearranging (4.5) as a linear combination of the machine parameters produces (4.6).

$$2\frac{\Delta\omega[n+1]-\Delta\omega[n]}{T_S\omega_0}H+(\delta[n]-\theta[n])B_G+\Delta\omega[n]K_D=P_M[n] \qquad (4.6)$$

Inspection of (4.6) reveals that the machine parameters, H, B_G, and K_D, are the only unknowns. The sampling period, T_S, and rated machine speed, ω_0, are constants. Additionally $\Delta\omega$, δ, θ, and P_M are all measured quantities or easily derived from the measurements. It then becomes necessary to solve an equation with three unknowns. Obviously one equation with three unknowns is an underdetermined set of equations, but this equation applies over all time. Thus additional equations can be incorporated by extending the solution across several timesteps. Since there are three unknowns the minimum number of equations is three. This would require measurements from four consecutive discrete timesteps. The estimation can be improved by including a larger set of measurement times, producing an overdetermined system of equations. Doing this would provide a best fit to across a large span of data, helping to counter the effects of measurement errors.

Given a set of measurement data, the system of equations given by (4.7) can be established. This set of equations is based on *N* consecutive discrete measurement points numbered 1 to *N*. Each measurement point contains the machine angle, bus angle, machine mechanical power, and change in machine speed. One instance of (4.7) needs to be established for each machine in the system approximation model.

$$\begin{bmatrix} 2\dfrac{\Delta\omega[2]-\Delta\omega[1]}{T_S\omega_0} & \delta[1]-\theta[1] & \Delta\omega[1] \\ 2\dfrac{\Delta\omega[3]-\Delta\omega[2]}{T_S\omega_0} & \delta[2]-\theta[2] & \Delta\omega[2] \\ \vdots & \vdots & \vdots \\ 2\dfrac{\Delta\omega[N]-\Delta\omega[N-1]}{T_S\omega_0} & \delta[N-1]-\theta[N-1] & \Delta\omega[N-1] \end{bmatrix} \begin{bmatrix} H \\ B_G \\ K_D \end{bmatrix} = \begin{bmatrix} P_M[1] \\ P_M[2] \\ \vdots \\ P_M[N-1] \end{bmatrix} \tag{4.7}$$

$$\mathbf{Am} = \mathbf{P_M}$$

Premultiplying (4.7) by the Moore-Penrose Pseudoinverse of **A** isolates the vector of machine parameters, **m**, giving (4.8).

$$\mathbf{m} = \mathbf{A}^{+}\mathbf{P_M} \tag{4.8}$$

The equation given by (4.8) represents the least squares solution of machine parameters which produces a best fit to the measurement data. With the vector of machine parameters determined the machine in the simplified model is completely described. This calculation needs to be performed for every machine to be inserted into the reduced system model.

As each bus in the reduced order system approximation is representing an area of the true system the preceding measurement values need to be considered in an area-wide sense. The machine mechanical input power, P_M, is the summation of the mechanical powers for all of the machines in the represented area. The angle values are those values measurement by the synchronized measurement device. Since every measurement device will have an associated bus in the reduced model these angles completely define the voltage angles in the system approximation. This procedure lumps the machines in an area of the system into one machine and appropriately sizes its internal parameters to best match the measurement data from the training set.

4.3 ESTIMATION OF LINE ADMITTANCES

With the generators of the reduced system specified it is necessary to derive impedances for the lines which connect the pseudo-busses. These line impedances coupled with the machine parameters and the aggregated loads give a complete definition of the reduced system model. This derivation requires the same training set of measurement data along with the generator impedances determined in the previous step. As outlined in the Chapter 2 each bus in the reduced model supports a generator, a constant power load and a number of lines connecting it to the neighboring busses. Since a DC power flow is used each of these lines is lossless, modeled by a series impedance. For a pseudo-bus *i*, connected to the set of pseudo-busses given by *M*, this DC power flow is given by (4.9), adapted from the derivations in the Chapter 2.

$$P_{Gi} + \sum_{j \in M} P_{ji} = P_{Li} \tag{4.9}$$

Upon substitution of the DC power flow equations and rearranging, (4.10) results.

$$\sum_{j \in M} \frac{\theta_j - \theta_i}{X_{ij}} = P_{Li} - \frac{\delta_i - \theta_i}{X_{Gi}} \tag{4.10}$$

Here P_{Li}, δ_i, θ_j, and θ_i are all directly measured variables from the training set and X_{Gi} was determined with the machine parameters in the previous step. The load power, P_{Li}, in (4.10) is the total load at the pseudo-bus *i* and it represents the total real power of the load in the area defined by pseudo-bus *i*. The unknowns in this equation that need to be determined are the line impedances X_{ij}. As one of these equations results for every bus a set of equations exists and the impedance of each line appears in this set as an unknown. Depending on the structure of the reduced system this set of equations may or may not be solvable. Despite this the set of equations can be brought into an overdetermined case by adding additional copies of each equation across multiple timestamps. Doing this follows the same principle as outlined in the previous section and serves to counter the effects of measurement errors while improving the fit over a broad span of data.

Formulating this set of equations for an example system over a discrete timespan 1 to *N* as a matrix operation results in (4.11). Note that (4.11) is formulated for a specific case, the final structure of these matricies is dependent on the structure of the desired reduced system model. Each row of the **Θ** matrix corresponds to the line flows for a pseudo-bus at a given timestamp. One power flow equation maybe described by several rows in the **Θ** matrix with each one using the data from a different measurement point. The zero entries in the **Θ** matrix correspond to lines that are not directly connected to the pseudo-bus described by that row. The **Θ** matrix is multiplied by a vector of inverse line admittances, **g**, with one entry for each line in the reduced system. The result of this multiplication is equal to the difference of the machine and load powers of the bus for the associated measurement point.

One Element for each Line in System

$$\begin{bmatrix} \theta_1[1]-\theta_2[1] & \theta_3[1]-\theta_2[1] & 0 & 0 & \cdots & 0 \\ \theta_1[2]-\theta_2[2] & \theta_3[2]-\theta_2[2] & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boxed{\theta_1[N]-\theta_2[N]} & \boxed{\theta_3[N]-\theta_2[N]} & \boxed{0} & \boxed{0} & \boxed{\cdots} & \boxed{0} \\ 0 & \theta_2[1]-\theta_3[1] & \theta_5[1]-\theta_3[1] & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \theta_2[N]-\theta_3[N] & \theta_5[N]-\theta_3[N] & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} X_{1,2}^{-1} \\ X_{2,3}^{-1} \\ X_{3,5}^{-1} \\ X_{4,7}^{-1} \\ \vdots \\ X_{ij}^{-1} \end{bmatrix} = \begin{bmatrix} P_{L2}[1]-\dfrac{\delta_2[1]-\theta_2[1]}{X_{G2}} \\ P_{L2}[2]-\dfrac{\delta_2[2]-\theta_2[2]}{X_{G2}} \\ \vdots \\ \boxed{P_{L2}[N]-\dfrac{\delta_2[N]-\theta_2[N]}{X_{G2}}} \\ P_{L3}[1]-\dfrac{\delta_3[1]-\theta_3[1]}{X_{G3}} \\ \vdots \\ P_{L3}[N]-\dfrac{\delta_3[N]-\theta_3[N]}{X_{G3}} \\ \vdots \end{bmatrix} \tag{4.11}$$

Line Flows for pseudo-bus 2 at Timestamp *N*

$$\mathbf{\Theta g} = \mathbf{P}$$

Machine and Load Powers for pseudo-bus 2 at Timestamp *N*

Premultiplying by the pseudoinverse of the **Θ** matrix yields the least squares solution of the line impedances vector as shown in (4.12).

$$\mathbf{g} = \mathbf{\Theta}^{+}\mathbf{P} \tag{4.12}$$

With the **g** vector determined the inverse of each element can be taken to find each of the individual line impedances. These impedance values are those which provide a best fit to the training set data in a least squares sense. Used in conjunction with the machine parameters derived previously a complete description of the reduced system is achieved.

The reduced system model as described approximates a large power system by placing a pseudo-bus at each measurement point. Thus the placement of the pseudo-busses in the reduced system is predetermined by the location of measurement devices in the real system. Although the location of pseudo-busses is established the placement of connecting lines is not governed by any properties of the original system. In actuality a line can be inserted connecting every pair of pseudo-busses, an impedance value can then be derived for each of these lines using the procedure of (4.11) and (4.12). If this is done many of the lines will be assigned extremely large impedances, indicating those two pseudo-busses are highly isolated from each other and a connection between them is unnecessary in the reduced model. In practice only those pseudo-busses that are natural neighbors need connecting lines. Using this reduced set of lines would drastically reduce the size of the matricies in (4.12) and subsequently reduces the required computation time.

4.4 VERIFICATION OF SYSTEM REDUCTION TECHNIQUE

The first step in the validation and testing of this system reduction technique involved the approximation of a system fitting the simulator architecture. A large system was constructed using the simulator bus archetype and line modeled by pure impedance, this system was then reduced as detailed above. Hence the system being reduced only employs the swing equation and DC power flows, other more complex power system relationships are not involved in the training data set. Thus this first batch of testing does not validate the reduction technique against a true AC power system. The value in these tests serves to demonstrate the accuracy achieved in removing system components. Although a realistic set of input data is not employed the concept of replacing multiple busses and lines with a numerically derived equivalent will be demonstrated and verified.

The input data for this reduction testing was generated using the simulator detailed in Chapter 2. The implemented simulator model consisted of 225 busses connected in a 15 x 15 grid with lines connecting the horizontal and vertical neighbors. The parameters for this system and simulation were set as follows:

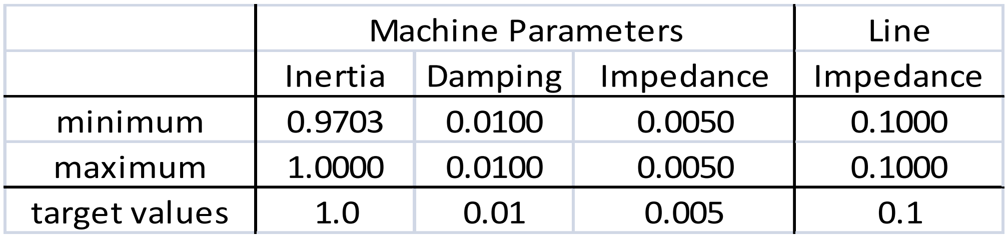
System Parameters:		*Simulation Timing*:
$H = 1$	$X_L = 0.1$	$T_S = 10^{-4}$ sec
$K_D = 0.01$	$P_L = 1$	*run time*: $t = 0 \rightarrow 5$ sec
$X_G = 0.005$	$P_M = 1$	

Inciting Event:
Load on middle bus (Bus 113) doubled to 2 p.u. for $0.30 \leq t \leq 0.35$

With the reference simulation performed the machine and load powers were cataloged along with the machine and bus angles. It is this data set that was used as the start point for the system reduction testing that follows.

The first approximation test performed used the same number of busses as the original system. This in fact does not reduce the system at all but provides a simple test of the parameter estimation equations, for this reason it will be referred to as a full order approximation. A pseudo-bus was inserted for each bus location in the original system. Next line positions were chosen by using a Delaunay Triangulation [16] of the pseudo-busses. A line was inserted for each edge in the triangulation; this effectively connected every bus to its natural neighbors. In addition to the lines along the horizontals and verticals present in the original system this method added one diagonal line for every cell in the original system. Although this procedure inserted more lines than necessary it is easily automated and as covered previously the insertion of extra lines will not adversely affect the calculation of system parameters.

Upon computing the approximated system parameters as per the equations derived in Sections 4.2 and 4.3 the results proved nearly identical to the original system. Using this procedure a separate set of parameters was calculated for each machine and line in the approximated system. The spread of these computed system parameters is given by Table 4.1. Every computed system parameter was nearly identical to the original system values in fact the values of machine damping, machine impedance and line impedance were identical to the working precision. The fourth class of system parameter, the machine inertia was within three percent of the original values. Finally the impedances of the extra diagonal line elements were found to be on the order of ~5000 per unit, indicating these new lines were acting as open circuits, i.e. they were removed in the computation process.



	Machine Parameters			Line
	Inertia	Damping	Impedance	Impedance
minimum	0.9703	0.0100	0.0050	0.1000
maximum	1.0000	0.0100	0.0050	0.1000
target values	1.0	0.01	0.005	0.1

Table 4.1: Span of Computed System Parameters for Full Order Approximation

Since the parameters of the approximated system are almost completely identical to those of the original system the simulations of the two systems were very nearly identical. Comparing the bus angle deviations of the central bus between the reference and approximated system gives Figure 4.5.

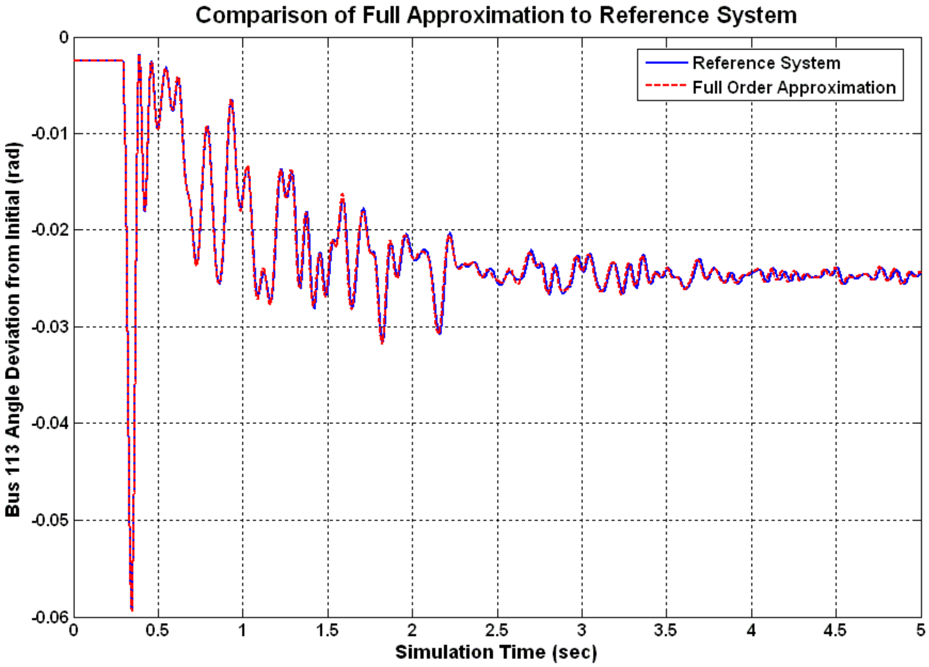


Figure 4.5: Comparison between Reference System and Full Order Approximation

An error metric will now be defined that will be used throughout the system approximation validation procedures. This metric is given by (4.13) it is the absolute value of the difference between the bus angle in the reference system and the approximated system, averaged across time and all of the system busses. This gives the average bus angle error incurred by the approximation process.

$$\varepsilon = \frac{\sum_{n}\sum_{i}\left|\theta_{i,Orig}[n]-\theta_{i,Approx}[n]\right|}{n(i)} \tag{4.13}$$

Applying the error metric of (4.13) to the full order approximation gives an average bus angle error of 1.3476*10^{-4} radians. This value is negligible to the necessary angle precisions in a practical power system, reinforcing the observation that the full order approximation is nearly identical to the reference system. This first test helped to validate the approximation procedure and showed that the system parameters can be accurately computed from the measurement data.

This approximation procedure will now be tested for more appropriate system reductions involving the removal of busses from the system. For each test case a given number of busses in the original 225 bus system would be randomly selected for deletion. The power lost through these deletions would be distributed among the remaining nearby busses to preserve the system size and flow directions. This bus removal procedure is demonstrated here by a series of figures. Figure 4.6 give the original system used for this demonstration. The system used here is a 25 bus system arranged as a 5x5 regular grid as opposed to the 225 bus test system in order to provide a simple demonstration. Each bus in Figure 4.6 is represented by a red dot; they are connected by lines running along the horizontals and verticals. Each of these busses has a load and generator attached as per the basic simulator model. The machine mechanical input power, P_M, and load power, P_L, is identical for each bus, thus initially the lines are unloaded much like the test system. These power injections and withdrawals are represented by arrows pointing to and from the busses in Figure 4.6.

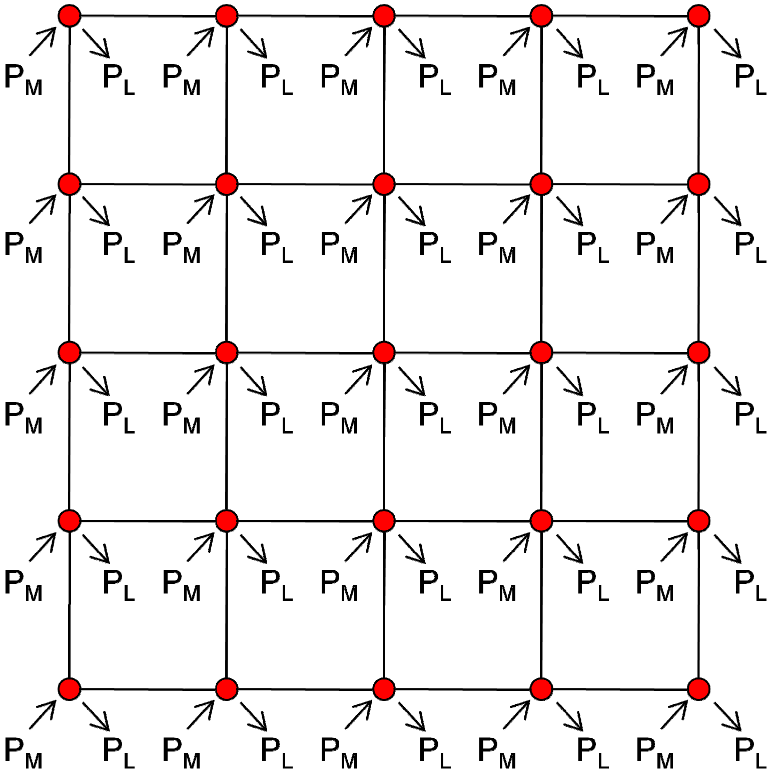


Figure 4.6: Original System for Bus Removal Demonstration

Removing the lines from Figure 4.6 and randomly selecting 5 busses for deletion gives Figure 4.7. The power flowing in and out of these removed busses has been evenly distributed among the nearest busses, which now become the pseudo-busses in the system approximation. This extra power assigned to the nearby pseudo-busses has been denoted by heavier arrows. For the subsequent parameter sizing the power flows used will be these larger flows resulting from the removed busses. This procedure of splitting the missing power keeps the system at the same size in terms of total power but reduces the overall system order due to the lower number of bus elements. Each pseudo-bus is thus now representing an area of the system instead of one bus from the original system as with the previous case.

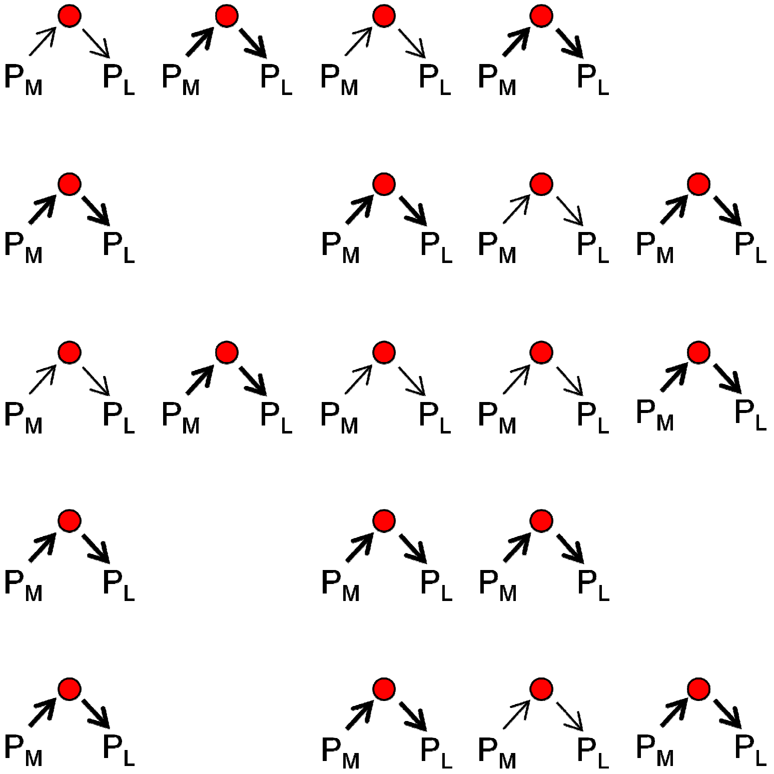


Figure 4.7: Demonstration System with Lines and 5 Busses Removed

Now that the pseudo-bus locations and power injections have been established the associated machine parameters can be derived using (4.8). Doing this sizes each equivalent machine in the reduced system. Next lines are inserted connecting each of the pseudo-busses. Once again a Delaunay triangulation is used for line placement as it is easily automated and guarantees a set of lines which will completely connect the reduced system. This set of connecting lines and the final structure of the reduced demonstration system is given in Figure 4.8. With these line connections established (4.12) can be used with the measurement data from the original system to determine the necessary line impedances. After sizing the lines the system approximation can be simulated and the results compared back to the results from the original system.

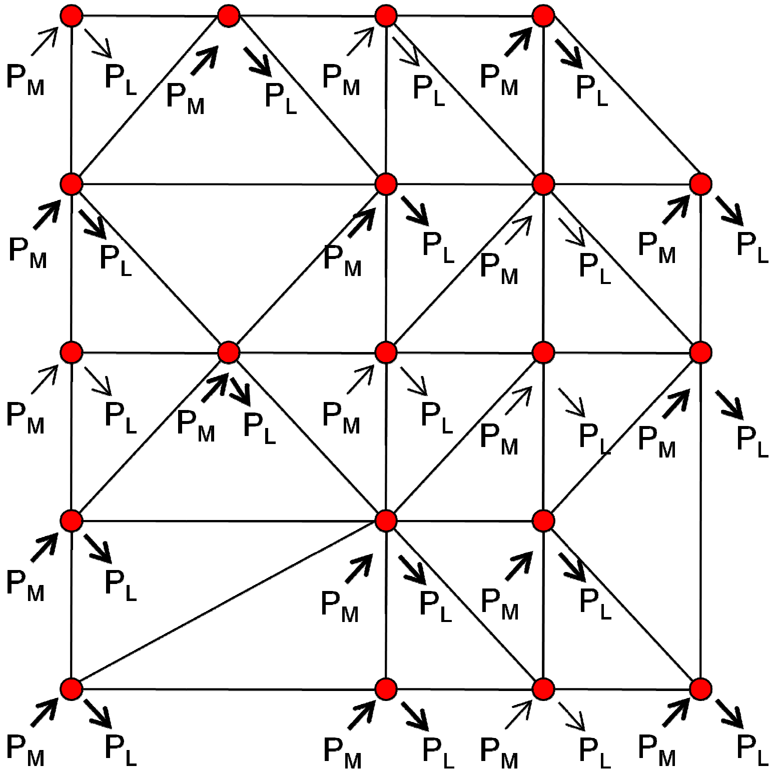


Figure 4.8: Demonstration System with Lines Inserted

The above procedure was applied to the 225 bus reference system with varying numbers of pseudo-busses used in each approximation. The 225 bus test system was approximated with systems containing sets of 200, 175, 150, 125, 100, 75, 50, 25 and 5 pseudo-busses. In these cases the appropriate number of busses in the reference system was randomly selected for deletion. The remaining busses in the system became the pseudo-busses of the reduced model then the approximation and comparison was executed. The test for each set size was repeated 25 times with a different set of randomly selected busses removed every time. The repetition here helped to provide an error distribution for each approximation size.

For most of the trials the approximation provided a good model of the underlying reference system. The error metric of (4.13) was computed for each test to be used as the measure of fit quality. These error metric results are summarized in Figure 4.9. The independent axis in this plot is the percentage of busses removed from the reference system in the approximation procedure. For example if 100 busses were removed from the system this represents 100 / 225 or 44.44% of the busses removed. The full order approximation appears as 0% removed while the 5 bus approximation tests are marked as 97.77% on the abscissa. The independent variable in Figure 4.9 is the computed error metric for each trial.

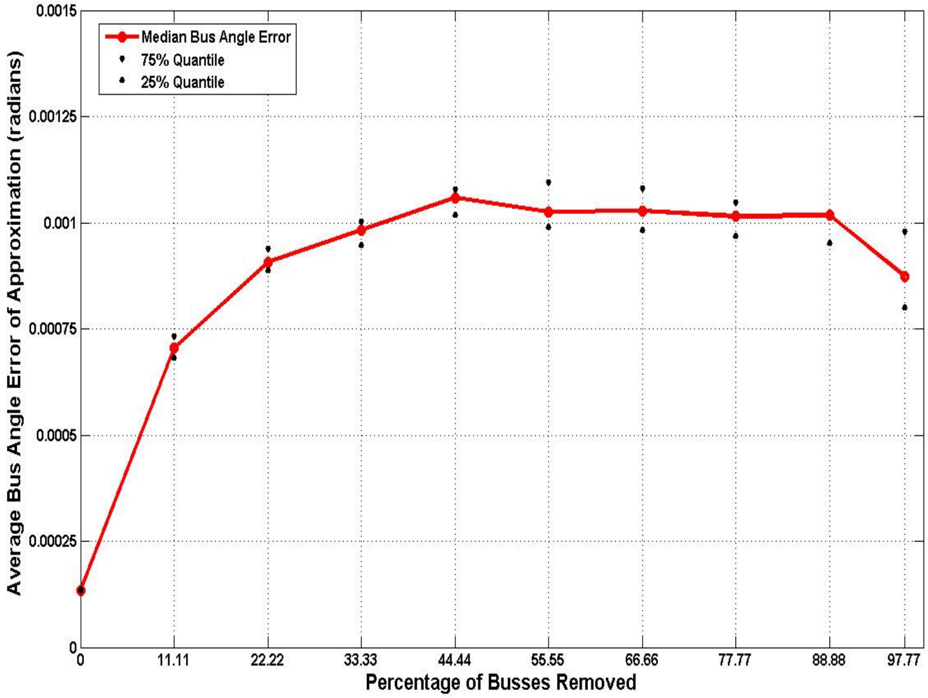


Figure 4.9: System Approximation Error as a Function of Number of Pseudo-Busses

For each number of busses removed 25 trials were performed and thus there are 25 distinct results each providing a different error metric value. This gives an error distribution for each system size. The red line plotted in Figure 4.9 gives the median value of this distribution for each approximated system size. The 75% and 25% quantiles of the error distributions are given by the upper and lower black dots respectively.

Several trends are demonstrated by Figure 4.9. Firstly for all of the test cases the median bus angle error was within 0.00125 radians, an extremely small value. This demonstrates the validity of this system reduction technique as even when a 225 bus system is approximated by a 5 bus system the expected average approximation error is within one one-thousanth of a radian. Additionally it is observed that as more than 33% of the system busses are removed the approximation error remains constant. Although the expected average error remains constant as more and more busses are removed the spread of the error distribution increases.

Some of these effects can be explained by looking at the bus angles resulting from these simulations. Figure 4.10 gives the bus angles measured at the same location in the reference system, the 150 bus approximation and the 5 bus approximation. Form this plot it is seen that reducing the system size does not affect the position of the final operating point as all three of the traces settle out to the same final value. The data feature that is lost through the approximation is the oscillatory content of the system response. This is expected as a larger more complex system has more generators interacting with each other creating more interarea swings as the system settles out to its final value. The smallest order system demonstrates a very low amplitude oscillation as it asymptotically approaches the new operating point. This explains the constant median bus angle error represented by the horizontal line of Figure 4.9 as all of the signals are oscillating around the same exponential decay function.

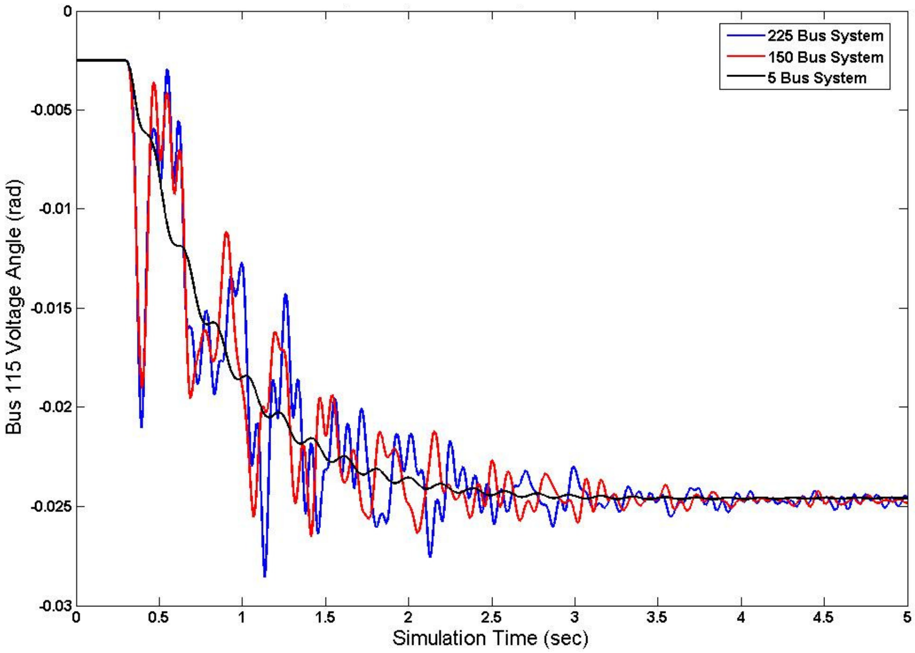


Figure 4.10: Representative Angle Traces from System Reduction Testing

For several of the preceding bus removal tests the resulting approximated system proved to be unstable. After computing the least squares solutions for the system parameters the ensuing simulation contained underdamped oscillations. This low damping caused the bus angle values to blow up and the system never stabilized after the simulation disturbance. This only happened in a minority of the test cases, specifically 15 of 221 tests produced unstable system approximations. In general this instability was more prevalent when a larger number of busses were removed. The tests removing 200 busses (88%) demonstrated the highest rate of unstable results with 6 of 25 tests producing unstable systems. This is the reason for the missing 75% quantile marker in Figure 4.9; in fact the 75% quantile was drawn out of the scale of the plot due to the high number of unstable test cases.

Overall the 225 bus test case detailed here provided encouraging results. It demonstrates the feasibility of system reduction using the least squares parameter estimation method. The majority of the test cases produced reduced order systems which responded in much the same fashion as the original. In all stable cases the reduced system settled to the same steady state operating point as the original along a similar exponential decay. The only different is the response came as a part of the oscillatory content. The original system which contained a large number of generators demonstrated oscillations of larger magnitude and higher frequency. As the system was reduced and the number of generators decreased the lower number of interacting units simplified the response.

4.5 APPROXIMATION OF AN AC SYSTEM MODEL

The system reduction tests described in Section 4.4 demonstrate the effectiveness of the reduction procedure on the simplified system model. In this case the power flow and swing equation modeling used in the reduction process identically matched the physical relationships of the original system. Although the system architecture has changed through the reduction procedure the underlying principles are the same. The ultimate goal of this research is to develop a reduction procedure applicable to a real-world power system. For this case the power flows in an AC system would be approximated by DC power flow equations. Thus the non-linearity is being modeled by a set of linear equations. In order to assess the errors incurred by the approximation procedure the preceding modeling techniques were applied to the data sets resulting from a PSS/E simulation.

The simulation model used for this test was the 23 bus sample system provided with the PSS/E package. This system was selected as a test bed because it was small, stable and easy to analyze. This system model contains 23 busses, 7 loads and 6 generators; the one line diagram is given by Figure 4.11. The source data for the reduction procedure was drawn from a simulation of this system from 0 to 10 seconds. The event in this system was incited by doubling the load on Bus 203 at 1 second and returning it to normal at 1.1 seconds. Throughout the simulation the bus angles, machine angles, load powers and machine mechanical powers were recorded. These recorded values were then used as the input measurement vectors for the reduction process.

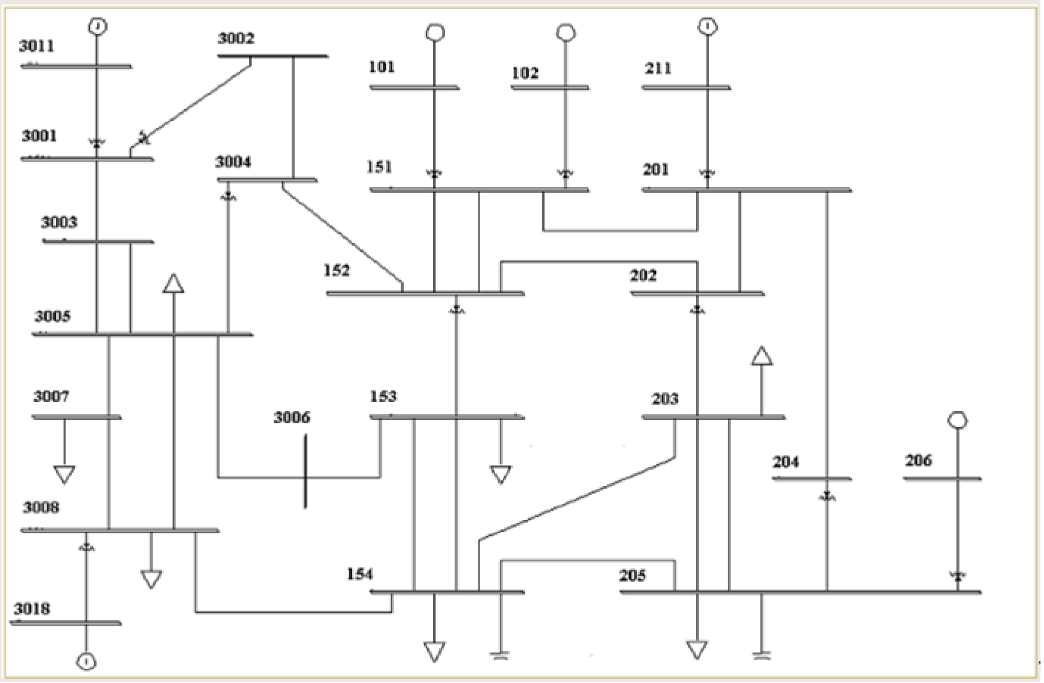


Figure 4.11: PSS/E Test System Used for Model Reduction

As this simulation is performed using the full AC power flows of PSS/E it includes transmission losses through the lines. The reduced model only incorporates DC power flow modeling and is thus a lossless system. This creates a power imbalance between the loads and mechanical power input in the reduction process. The simulation output data corresponds to a situation in which the generators are producing enough power to meet the needs of both the line losses and load demand. If these same mechanical and load power quantities are used in the reduction the generators will be supplying more power than drawn by the system. This power imbalance will cause the angles across the system to accelerate and an unstable system. To correct this imbalance all of the load measurement quantities were scaled up so that they matched the total input power.

Several tests were performed which attempted to reduce this system and match it to the simplified model. The procedure and results from one of these tests are presented here. For this example case the 23 bus system is approximated by a 6 bus simplified model. The first step in this process is to partition the system into 6 regions which will each be represented by a pseudo-bus from the reduced model. In order to do this 6 of the original busses are selected as the representatives and the other regions of the system are assigned to their nearest representative pseudo-bus. The implemented partitioning is given in Figure 4.12. Here the system area has been split into 6 regions numbered 1 through 6. Each regain has one of its busses indicated as the representative as denoted by the stars. It is the angle measurements from these busses which the reduced model will attempt to match.

After partitioning the measurement set resulting from the simulation is reduced to match the scope of the desired model. All of the bus and machine angles from the representative pseudo-busses are preserved, discarding the remainder. As with the previous reduction example the power injections are aggregated over each of the individual regions. For example the load power of region 4 is the summation of the loads on busses 3007 and 3008 and the input mechanical power to the machine model in region 2 is the summation of the mechanical power inputs to the machines on busses 101 and 102.

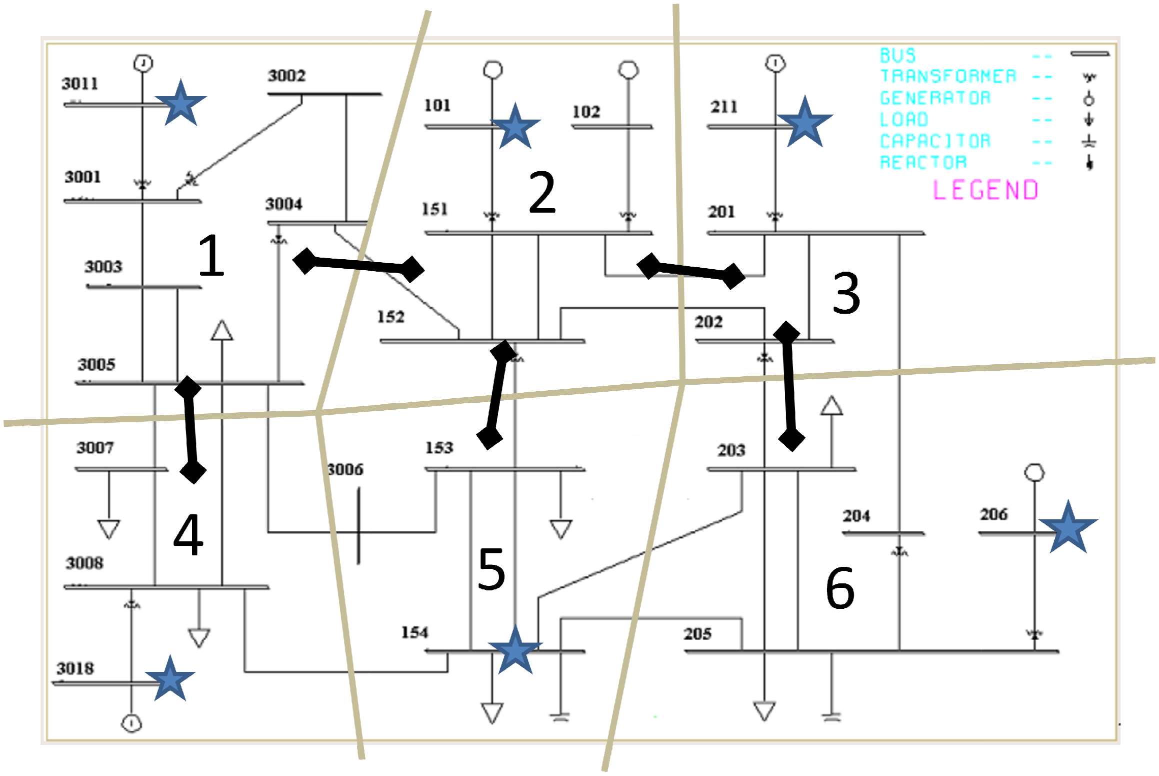


Figure 4.12: Partitioning of PSS/E Test System

The final step in determining the architecture of the reduced model is the placement of connecting lines. As discussed in the previous reduction testing lines maybe inserted anywhere in system. For the example detailed here the six lines indicated by the heavy black lines of Figure 4.12 were selected. This small set of connecting lines was chosen for the example to reduce the amount of necessary calculations. Any number of lines maybe inserted in this step, as mentioned with the previous example the estimation process will take care of extraneous connections by assigning them abnormally high impedances.

After selection of the representative busses and line placement the reduced system architecture is established. For the example here this architecture is given by the one-line diagram of Figure 4.13. With the reduced system architecture laid out the parameters which produce a best fit need to be derived. This is done by first setting the machine parameters using the measurement data and the least squares fitting described by (4.7) and (4.8). After this is completed the line impedances are estimated from the DC power flow equations using (4.11) and (4.12).

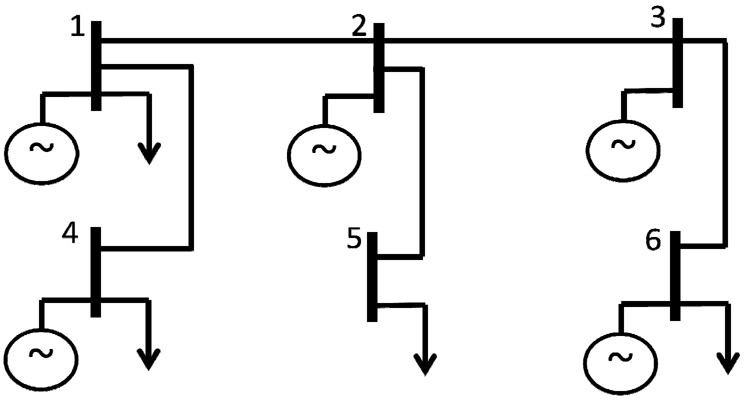
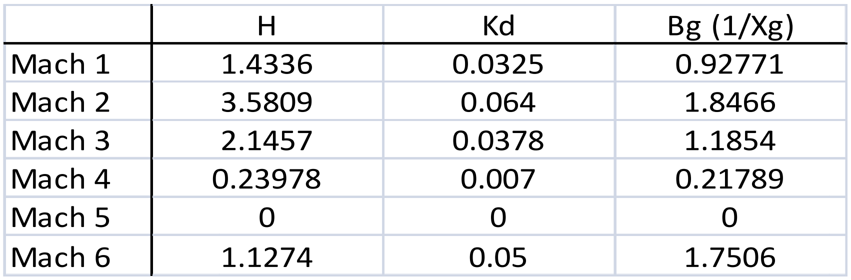


Figure 4.13: Reduced System for Example

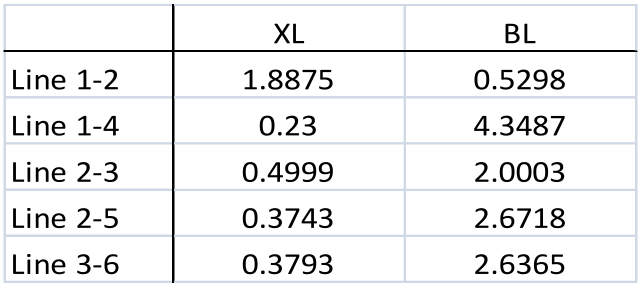
Using the aggregated mechanical power measurements along with the machine and bus angles for the representative pseudo-busses in (4.8) the machine parameters given by Table 4.2 are computed. Since pseudo-bus 5 of the reduced model has no generator its computed parameters are all zero valued.



	H	Kd	Bg (1/Xg)
Mach 1	1.4336	0.0325	0.92771
Mach 2	3.5809	0.064	1.8466
Mach 3	2.1457	0.0378	1.1854
Mach 4	0.23978	0.007	0.21789
Mach 5	0	0	0
Mach 6	1.1274	0.05	1.7506

Table 4.2: Estimated Machine Parameters for Reduced System

With the computed machine admittances, *Bg*, of Table 4.2 and the measurement data it is now possible to solve for the necessary line impedances. As discussed in Section 4.3 this estimation requires the metered load powers, bus angles, machine angles and the machine admittances. These values are plugged into (4.12) and the best fit values for the line impedances are determined. For the example system six lines were used to connect the pseudo-busses, the best fit impedance values of these lines is given in Table 4.3. The necessary line reactance is given by XL and its inverse, the susceptance, is given by BL. It is this susceptance which is used to construct the modified admittance matrix for the simulator module.



	XL	BL
Line 1-2	1.8875	0.5298
Line 1-4	0.23	4.3487
Line 2-3	0.4999	2.0003
Line 2-5	0.3743	2.6718
Line 3-6	0.3793	2.6365

Table 4.3: Estimated Line Impedances for Reduced System

With these estimated machine and line parameters the reduced system model can be constructed. Substituting the values of Table 4.2 and Table 4.3 in for the parameters of the system model in Figure 4.13 gives the full reduced system approximation. Simulation of this model by the procedure of Figure 2.2 with the reduced power measurements should recreate the measurement vectors observed at the representative busses in the original PSS/E simulation. Through independent validation this was shown to be the case with the results matching the original source data well.

The ultimate goal of this research is to develop an application of the reduced system model such that it can be used to predict the evolution of system dynamics given preceding measurement values. To test this, the example reduced system model was implemented in a fashion which predicted future measurement states. The simulation procedure was initialized and run as normal being fed from measurements from the original simulation. At each time step the simulation was carried through a set timespan to estimate a future system state. During this prediction process the current system power injections are held constant and the angle values are allowed to respond to the known power conditions and the governing system properties. This prediction process is repeated after the receipt of each new measurement point, effectively readjusting the prediction based on the most recent data.

Setting up this prediction method to look ahead one sampling period gives the results of Figure 4.14. Since the sampling period is 0.0083 seconds this implementation is predicting the system angle values a half cycle in advance. The solid traces plotted here correspond to the 6 bus angles in the reduced model. The dashed traces give the original angle responses for the six representative busses which are attempting to be matched. In this plot the predicted values are plotted corresponding to the timestamp of the prediction. By inspection it is seen that the predicted values closely matched the true values. The prediction accurately determines the upcoming values especially for the steady state regions. In fact when averaged across all the busses and simulation time span the error of the predictor is 0.0013 radians.

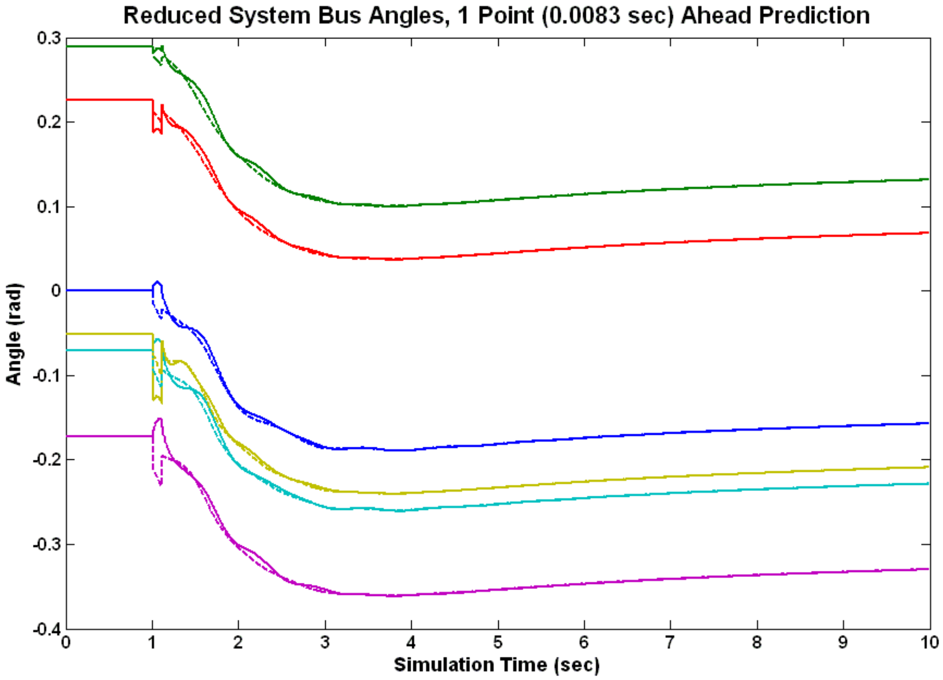


Figure 4.14: Predictor Response for 0.0083 sec Look Ahead

This prediction procedure was further tested with increasing look ahead ranges. Several of these tests were performed on the example system for various prediction times. The results for one of these tests are given in Figure 4.15; in this case the prediction period is set for 0.5 seconds. As before the model is fed with measurement data up to and including the current time point with the prediction based on carrying through the current system power injections up to the desired prediction objective. Once again the true values are plotted as dashed lines while the predicted values are given by the solid lines. In this case the offset between the true value and the prediction is noticed as the prediction traces start up later than the true values. This is also seen in the delayed response at the time of the event. As the event occurs the prediction fails to respond immediately as it is working from power measurement points that are 0.5 seconds old and thus the operating point change has not yet been observed. This delay is as expected because the prediction method is using a static system and input model and cannot anticipate shifts in power demand.

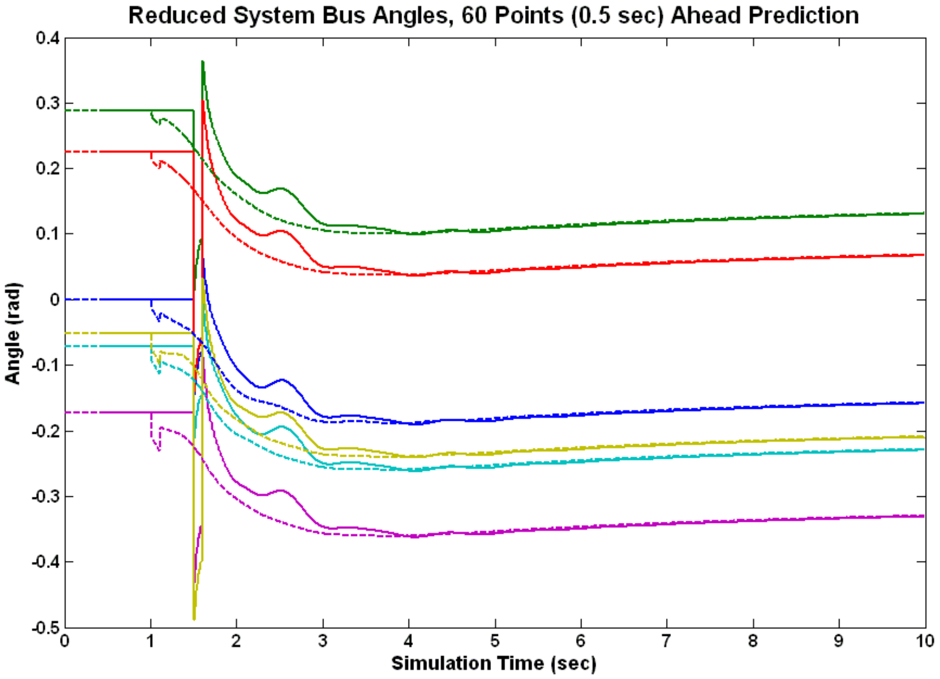


Figure 4.15: Predictor Response for 0.5 sec Look Ahead

Once the operating point changed is observed in the power measurements the predictor does respond accordingly. It experiences a shorter transient period as the angle measurements move quickly to achieve the final steady-state value of the system. Based on this response it is determined that the predictor performs better than expected considering it is dealing with a rather large look-ahead period of 0.5 seconds. Although the transient periods do not match particularly well the prediction achieves the final steady state value at about the same time as the original data did. The time average error across all busses for this case was 0.0108 radians or about 6 % of the total drop.

The trend in prediction error demonstrated through these two examples was mirrored in the rest of the other tests not detailed here. As the prediction time increases the prediction error increases with it. This is as expected because the predictor is attempting to match the system with progressively less information as the prediction look-ahead time increases. Similarly the lag in predictor response seen at the inciting event is exactly the same width as the desired prediction time. This is as expected and unavoidable because there is no way of foreseeing the changing conditions which incite the event. Despite these drawbacks it was found that this system reduction and state prediction method worked extremely well during the steady-state portions of the response. The predictions only showed significant deviations from the true values in the transient period immediately after the event.

There are several possible areas where this methodology maybe improved. These topics will be left for future research efforts. Firstly the feasibility of replacing the DC power flow modeling with a full AC power flow should be investigated. Doing this should help resolve many of the error sources but will drastically complicate the estimation process. The complication would result from the non-linearity of the AC equations as opposed to the linear approximations currently used. Since the line impedance estimation procedure is based on solving the power flow equations for line impedance this step will become a non-linear solution most likely requiring an iterative process. Additionally the power flow equations need to be solved for the bus angles during the simulation and prediction steps, placing increased computation requirements on these portions of the process. Another advantage of extending the system to an AC model is the incorporation of real impedances and the voltage phasor magnitudes. With the inclusion of line resistances the system is no longer lossless and the power imbalance encountered here would not be seen. Inclusion of the voltage magnitudes allows for modeling of additional load types which use the full voltage phasor. The current architecture only used constant power loads, inclusion of constant current, constant admittance and more complicated load models as needed should help to improve the system estimation results.

Another possible improvement would be dynamically updating the system architecture to better match the system topology across time. As the power injections change over time the net effect is that the impedances for the reduced model need to change as well. This is a result of the order to which the system is being reduced. If one pseudo-bus in the reduced model is representing several busses from the original system the location of loads are not being accurately represented. It is the line estimation procedure which helps to correct this inaccuracy as the line impedances are fitted to provide the correct flows between each region. It was found that changing load conditions required differing sets of line impedances to provide a best match to the current state of the system. Thus a better fit can be achieved by allowing the line impedances to vary overtime or periodically update to match the current system state. Something akin to a Kalman filter which provides a plant matrix that evolves over time should be able to help this.

5 DEVELOPMENT OF AN INTERAREA MODAL EXTRACTION AND VISUALIZATION PROCEDURE

Interarea mode oscillations in power systems are global small-signal stability problems based on the architecture of the system. Given a specific system topology these oscillatory modes result. Each system will have its own characteristic modes which can be excited under a various number of stress conditions. When excited these modes create oscillations in power, generating unnecessary power flows which serve to introduce added stress to the system. If under damped or left unchecked these oscillations have the potential to incite cascading blackouts if the resulting power flows exceed line limits.

Due to these hazards several different control schemes have been implemented specifically targeting modes of known oscillation frequencies [17],[18]. A major pitfall of these methods is that they rely on knowing the specific interarea modal frequencies in advance to design a control scheme which dampens them. These mode frequencies can be calculated from the system topology but in many cases the system models maybe inaccurate, incomplete or unavailable. Additionally as the system topology changes over time the mode frequencies will change with it. These factors can contribute to an improperly tuned control system which may do more harm than good.

A different strategy for determining these oscillatory modes is based on system measurements. With the proliferation of high time resolution power system measurement devices it becomes possible to observe these interarea modes as they occur. As more devices are installed across the system a more complete picture of the different regions is achieved. The increased number of measurement vectors allows for identification of coherent generator groups, and their associated geographic areas. Through analysis of many different oscillation events a complete description of the system modes maybe achieved. Some studies along these lines have already been undertaken for the US Western Interconnect [19],[20] as well as the US Eastern Interconnection [21].

Reference [1] defines two types of interarea modes seen in a power system as follows:

(a) A very low frequency mode involving all the generators in the system. The system is essentially split into two parts, with generators in one part swinging against machines in the other part The frequency of this mode of oscillation is on the order of 0.1 to 0.3 Hz

(b) Higher frequency modes involving subgroups of generators swinging against each other. The frequency of these oscillations is typically in the range of 0.4 to 0.7 Hz

It is these definitions which be used as the basis for the identification procedure that follows. Oscillations in the range of 0.1 Hz to 0.8 Hz are detected and coherent groups are identified based on the phase angle of each measurement vector.

5.1 FILTERING CONSIDERATIONS

In order to provide a better fit of the oscillatory content the measurement data needs to be properly conditioned first. An example of measured frequency data demonstrating an oscillatory mode is given in Figure 5.1. It is desired to extract the properties of this oscillation. The data for this example is drawn from FDR measurements which capture the system response to a generation trip at the Donald C. Cook nuclear plant located in southwestern Michigan. This event occurred at about 19:05:30 UTC on July 26, 2009. The resulting system frequency drop is seen in Figure 5.1 as a sharp decline from 60.05 Hz to 59.96 Hz. During this drop period a strong oscillation is also observed with Maine oscillating 180 degrees out of phase with North Dakota. This example data set will be used throughout to demonstrate the operation of the modal identification procedure.

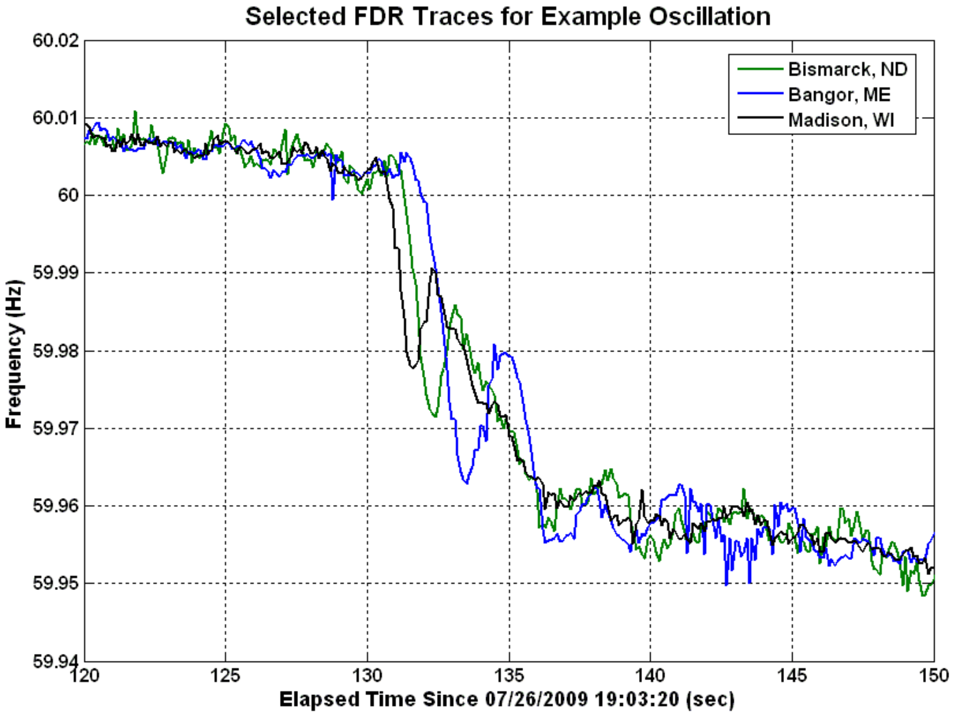


Figure 5.1: Raw Example Data Set

Although this oscillation is easily observed by visual inspection it is harder to isolate from a numerical perspective. The chief reason for this is the high DC component which dominates the frequency spectra. Taking the Fourier Transform of the signals in Figure 5.1 yields the frequency spectra of Figure 5.2. The frequency scale here is plotted on a logarithmic scale to better illustrate the low frequency components. All three of the data signals demonstrate similar frequency spectra as they all dominated by the same DC component (the 60 Hz system frequency). Figure 5.2 demonstrates the nature of the problem in that the interarea band is completely overshadowed by the lower frequency components. Additionally the magnitude of the lower frequency components draws the Fourier Transform and corrupts the results in the interarea band.

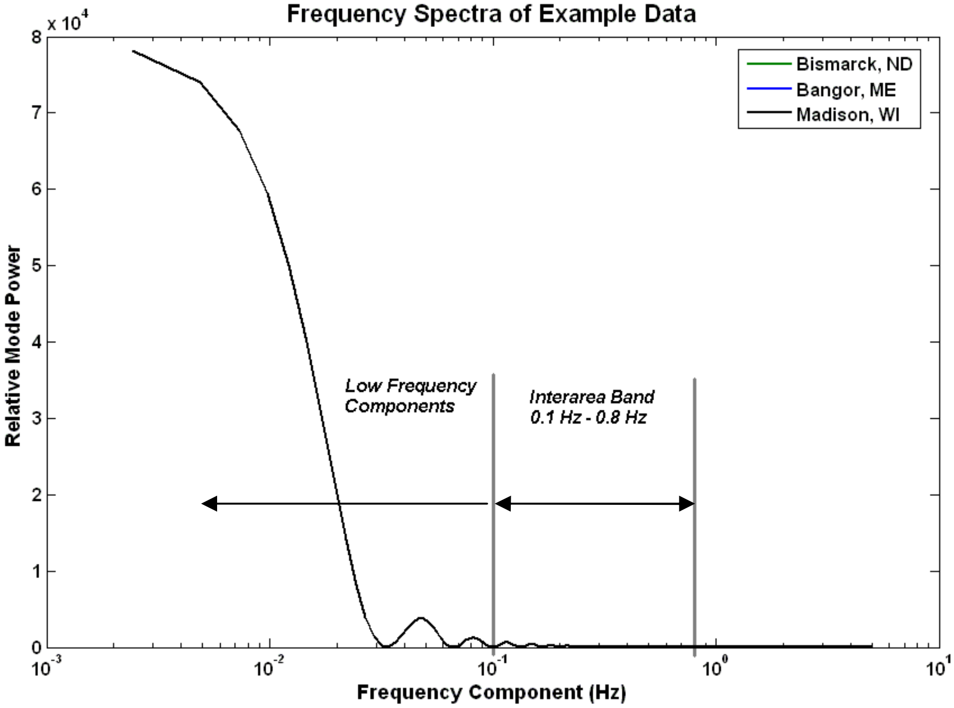


Figure 5.2: Frequency Spectra of Raw Example Data

The results of Figure 5.2 are expected as the frequency measurements are centered on 60 Hz and any deviations are significantly smaller. The oscillatory content of interest here generally demonstrates peak to peak amplitudes of around 20 mHz, when compared to the DC component at 60 Hz this value is insignificant and is completely overshadowed.

To resolve this problem the DC component needs to be mathematically removed from the signal. A simple solution to this problem is to subtract the median from the original signal, doing this detrends the data and centers the distribution of data points around zero. With the signal centered on zero the DC component is effectively removed as demonstrated in Figure 5.3. In this plot the magnitude of the low frequency components is drastically reduced as compared with Figure 5.3. Although the lower frequency components still dominate the frequency spectra the situation has been improved and the components within the interarea can now be observed within the Fourier Transform results. The remaining unwanted low frequency components will be further attenuated by a non-linear band pass filter at a later stage.

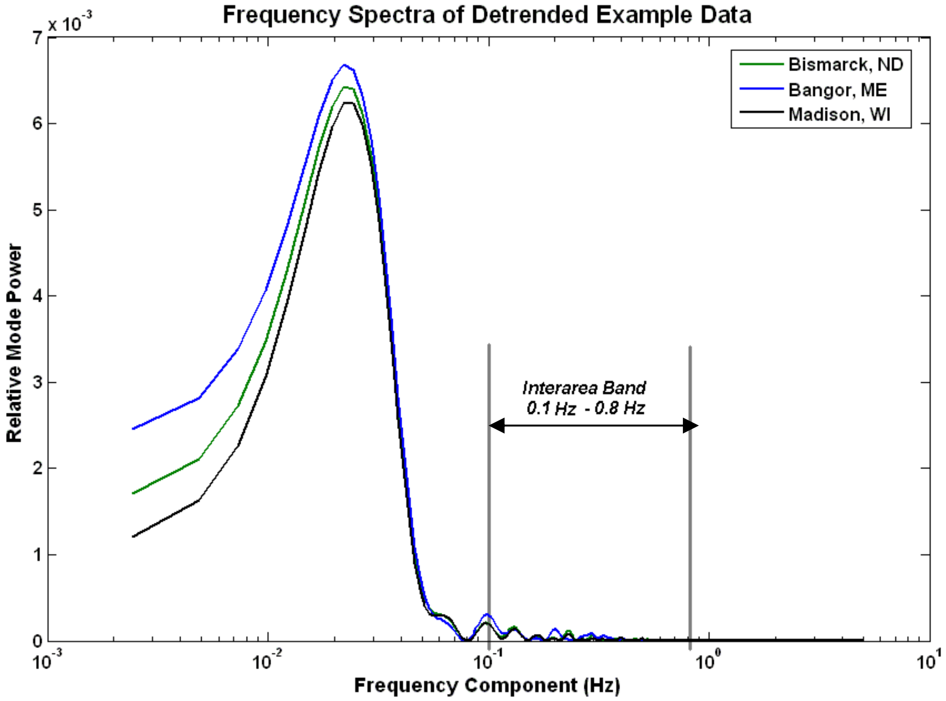


Figure 5.3: Frequency Spectra of Detrended Example data

At this point the noise content of the input signals will be considered. Observing Figure 5.3 it is seen that a noticeable amount of noise is seen in the example data. From previous work [22] the nature of noise in the FNET system has been characterized as being Laplacian in nature. This indicates it contains a higher number of outliers than that expected of a signal containing Gaussian distributed white noise. It has also been shown that a moving median filter provides an optimal filter for these noise elements.

A moving median filter is implemented by replacing a data point by the median of its immediate neighbors, resulting in a data window that slides across the vector of measurements [23],[24]. The size of this data window defines the sensitivity of the filter. Although no rigorous mathematical definition for the transfer function of this filter exists, it effectively implements a low pass filter. The break frequency of the filter is governed by the selected window size and sampling frequency of the discrete signal.

The magnitude spectrum of the moving median filter for various windows sizes is given in Figure 5.4. These characteristics were numerically derived and are based on a discrete sampling rate of 10 Hz. Each point on this plot was created by generating a discrete sinusoid of the test frequency sampled at 10 Hz. The generated sinusoid was then filtered with a moving median filter. This filtered signal was then compared to the original sinusoid to determine the filter gain for that specific input frequency and window size. This procedure was repeated across the frequency band from 0 to 5 Hz and for various filter window sizes. In a post processing step the higher frequency sidebands resulting from the discrete nature of the signal were removed.

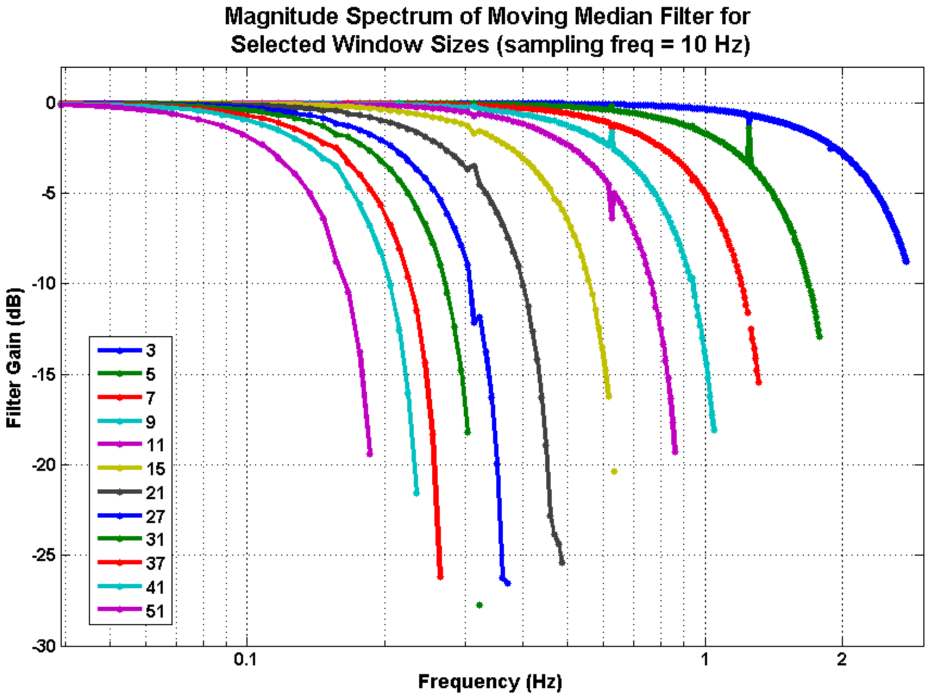


Figure 5.4: Moving Median Filter Transfer Function

Figure 5.4 demonstrates that as the window size of the moving median filter is increased the break frequency of the resulting low pass filter decreases. It is observed that the minimum window size of 3 points produces a filter that begins rolling off at 1 Hz and has a -5 dB gain at about 2.05 Hz. Conversely a very large data window of 51 points demonstrates a -5 dB gain at about 0.15 Hz.

To improve the performance of the modal extraction procedure the noise needs to be filtered out of the signal while preserving as much of the oscillatory content as possible. The source data in this case is FNET measurement vectors which have a sampling frequency of 10 Hz. By the Nyquist criteria the maximum observable frequency is one half of this sampling frequency, or 5 Hz. Realistically this limitation will be somewhat lower than 5 Hz. For all practical purposes everything above 2 Hz can be considered measurement noise in the FNET data.

Since the interarea oscillation components occupy the 0.1 Hz to 0.8 Hz band the selected low pass filter should pass these frequencies with minimal distortion. Thus the low pass filter used to denoise the FNET data for this application should have a break frequency greater than 0.8 Hz and less than 2 Hz. Based on the response of Figure 5.4 a moving median filter with a window size of 5 points was selected to satisfy these requirements. This filter demonstrates a gain of about -1 dB at 0.8 Hz, thus all low frequency components up to the top end of the interarea band are passed with minimal distortion. After 1 Hz the 5 point moving median filter demonstrates a step roll-off, achieving a theoretical -20 dB gain at about 2 Hz.

Application of the described detrending method and moving median based low pass filter to the example data set yields the filtered signals of Figure 5.5. When compared to the original signals of Figure 5.1 the level to which the noise elements have been attenuated becomes obvious. Figure 5.5 demonstrates very little high frequency noise content and the oscillatory content is much clearer. In the range after the event drop (times greater than 140) a smaller amplitude oscillation can be seen in all three signals. In the original data this same oscillation was present but obscured by the noise. Additionally the signals are now centered near zero, reducing the magnitude of the unwanted DC component as detailed above.

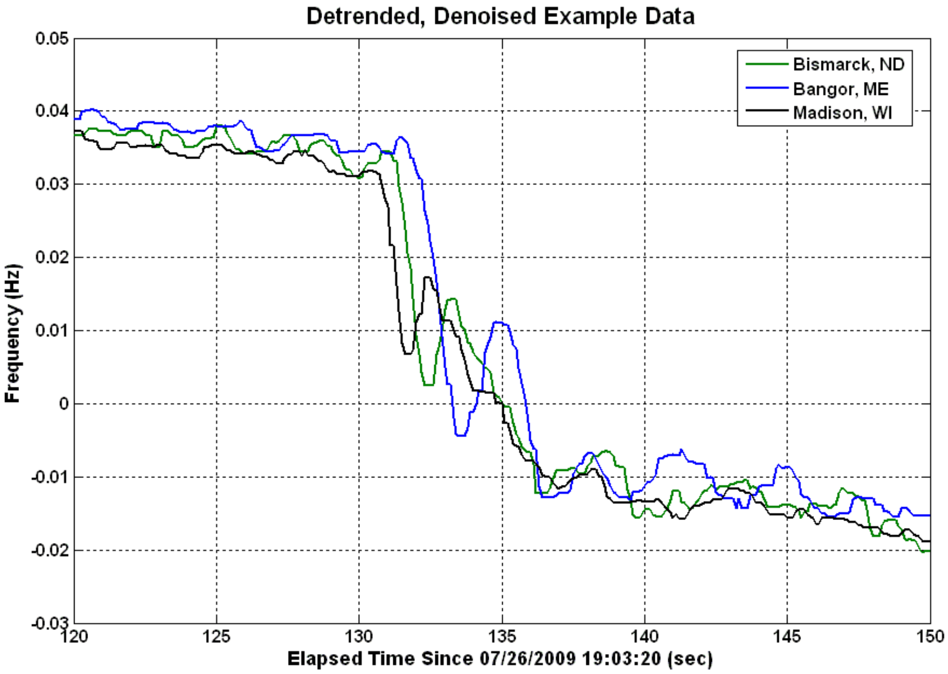


Figure 5.5: Detrended, Denoised, Example Data

The filtering process up to this point has helped to condition the input data and to isolate the oscillation frequencies of interest. Despite this the data signal still contains a large low frequency component which masks the inter area band as seen in Figure 5.3. It is wished to isolate only those frequencies within the interarea band so that the dominant oscillation mode can be extracted. This was achieved through the implementation of a non-linear band-pass filter. The filter would be primarily based on an Empirical Mode Decomposition (EMD) of the detrended and denoised input signal. The EMD and associated Hilbert-Huang Transform have been recently proposed as methods for isolating interarea modes in power systems [25],[26].

Empirical Mode Decomposition is a data driven method that decomposes a signal in a set of Intrinsic Mode Functions (IMF). Each IMF is an oscillatory signal which consists of a subset of frequency components from the original signal. As opposed to Fourier, Wavelet, Matrix Pencil and similar methods EMD constructs these component signals directly from the data by identifying local extrema and setting envelops around the signal in an iterative process. A fit is then performed on the local extrema to create an IMF. After creation of an IMF it is subtracted from the original signal and the process repeats to identify the next IMF. This identification and removal process continues until the original signal has been completely described by the set of IMFs. The output of the EMD process is the set of IMFs, generally this set is a small number of signals (usually less than 10 for the data sets considered here) that when summed together completely match the original signal. The number and spectral span of the calculated IMFs is defined by the input signal and the specified stopping criteria of the algorithm. The EMD algorithm employed does not explicitly compute oscillation frequencies, amplitudes or phase angles as with other signal decomposition techniques, instead the IMFs are derived directly from the input signal based on its local extrema. A complete mathematical description of the EMD algorithm is beyond the scope of this document but can be found in [27],[28],[29],[30].

Performing an Empirical Mode Decomposition on the Bangor trace of Figure 5.5 extracts the seven Intrinsic Mode Functions given in Figure 5.6. The first and second IMFs extracted in this process are given by the blue and green traces of Figure 5.6; these capture the high frequency components and noise of the input signal. The next three IMFs given by the red, cyan, and violet traces capture the middle frequencies present in the signal. Finally the last two IMFs extracted, those represented by the mustard and black lines, define the low frequency components of the input, representing the event drop itself in this case. This result serves to break apart the original signal into several signals, each focusing on a specific frequency band of the input. Since the EMD algorithm does not compute any specific oscillation frequencies they remain unknown but are now isolated from each other in the differing IMFs. It is important to note here that a summation of the seven IMFs of Figure 5.6 results in an exact match of the input signal and thus no approximation error has been incurred by the decomposition.

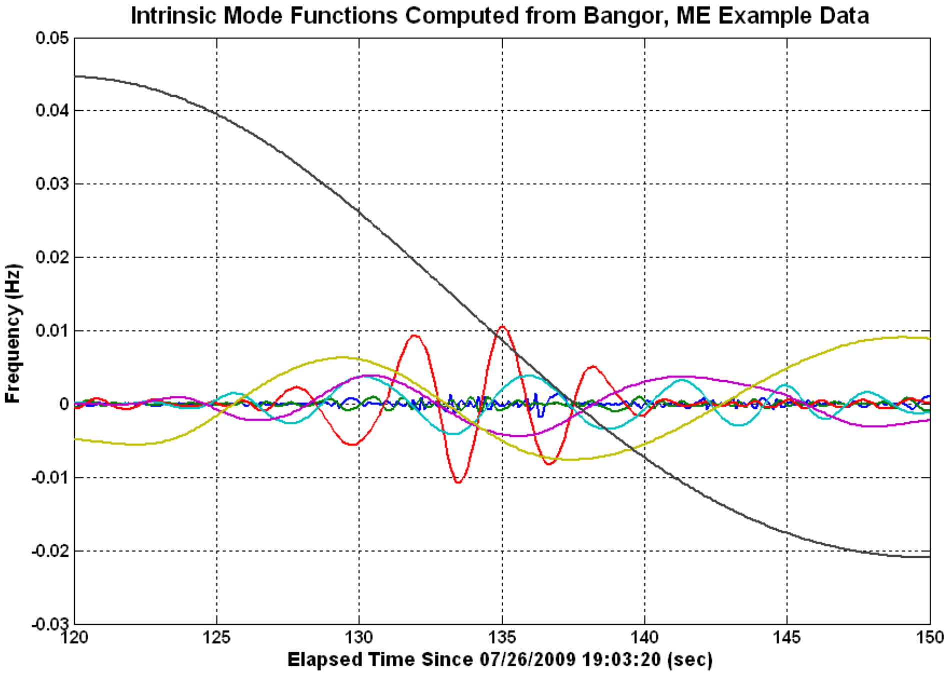


Figure 5.6: Intrinsic Mode Functions of Bangor Measurement Vector

The Fourier Transform of the IMF signals in Figure 5.6 is given in Figure 5.7. As before the frequency variable is plotted on a log scale to better demonstrate the frequencies in the lower range. Additionally the individual FFTs have been scaled by their maximum values so that the low frequency components do not dominate the independent variable scale as they did in Figure 5.3. Inspection of Figure 5.7 confirms that each IMF is capturing one specific band of the original signal. The first IMF extracted, which is plotted in blue is centered around 3 to 4 Hz and each subsequent one picks up successively lower frequency components, with the last IMF (the black one) capturing the lowest frequencies, located around 0.02 Hz.

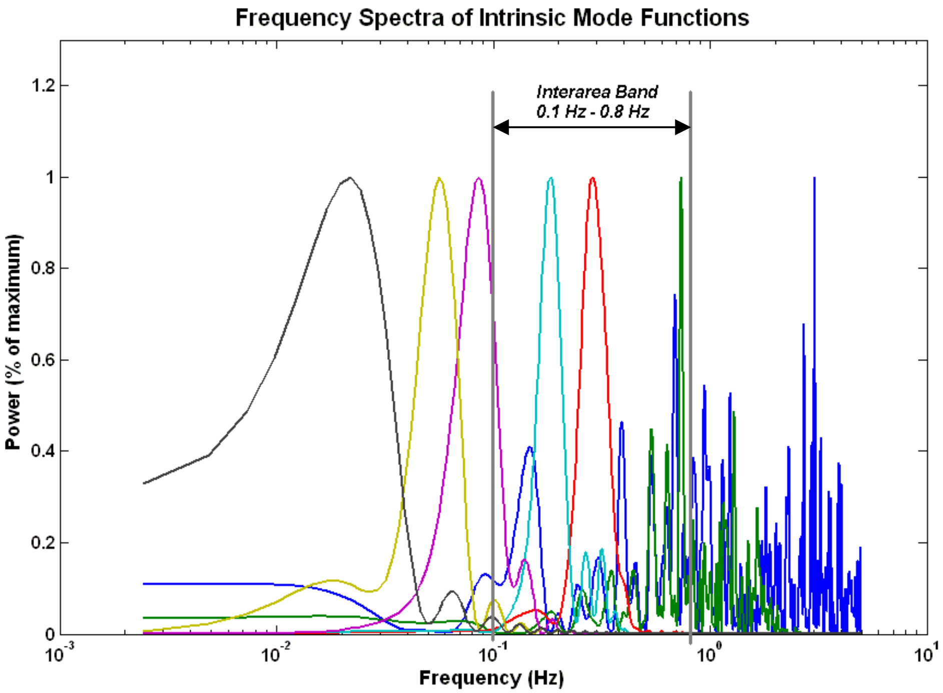
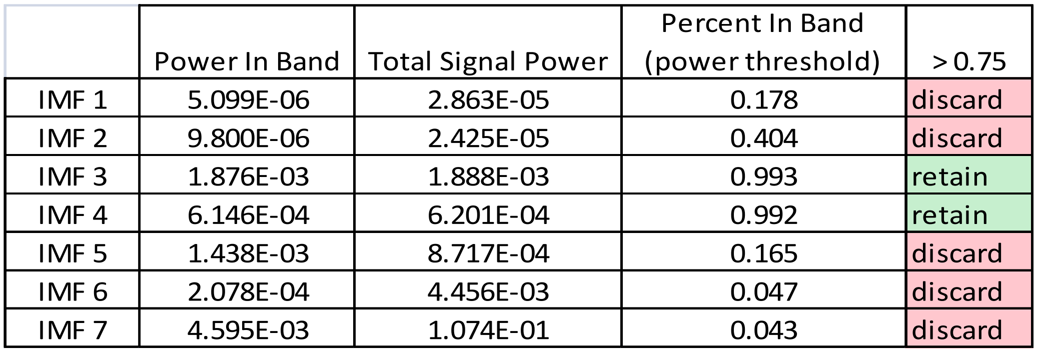


Figure 5.7: Frequency Spectra of Bangor IMFs

Since the interarea band is 0.1 Hz to 0.8 Hz it is wished to preserve only those IMFs which have a significant portion of their power in this band and discard the others. The final implementation of this filter computes the IMFs then performs a Fourier Transform of each one. Using the Fourier Transform results the percentage of total signal power within the interarea band is computed for each IMF. If this percentage exceeds a given threshold the IMF is retained, otherwise it is discarded. The final filter output is the summation of the retained IMFs. This effectively implements a band pass filter with three set points, the low frequency, the high frequency and the power threshold. In this implementation the threshold governs the steepness of the filter roll-off. As the threshold approaches 1 fewer IMFs are retained and the bandwidth of the output is limited.

Through testing on several data sets it was determined that cutoff frequencies of 0.1 Hz and 0.8 Hz with a power threshold of 0.75 provided the best response. These settings gave the best preservation of the interarea band while removing most of the other frequency components. The results for this filter on the example data for the Bangor measurement vector is given in Table 5.1, IMF 3 and IMF 4 were the only signals that exceeded the 0.75 power threshold requirement and thus the output is composed by summing them together.



	Power In Band	Total Signal Power	Percent In Band (power threshold)	> 0.75
IMF 1	5.099E-06	2.863E-05	0.178	discard
IMF 2	9.800E-06	2.425E-05	0.404	discard
IMF 3	1.876E-03	1.888E-03	0.993	retain
IMF 4	6.146E-04	6.201E-04	0.992	retain
IMF 5	1.438E-03	8.717E-04	0.165	discard
IMF 6	2.078E-04	4.456E-03	0.047	discard
IMF 7	4.595E-03	1.074E-01	0.043	discard

Table 5.1: EMD Based Band Pass Filter Results for Bangor Data

The total input filtering process for this modal identification application consists of three stages; first the median detrending stage, followed by a moving median filter and finally the EMD based filter. This process serves to isolate only the frequency components within the interarea band so that further processing can extract specified modes within this region. Applying this multistage filtering process to the set of example data results in the plots of Figure 5.8. Comparing this plot with that of Figure 5.1 it is seen that the low frequency trend is completely removed, leaving a zero centered signal. Additionally the higher frequency components and noise have also been removed from the raw data vectors. The oscillation in the interarea band observed during the event drop is now the most prominent feature of the data, making it easier to extract from a computational viewpoint.

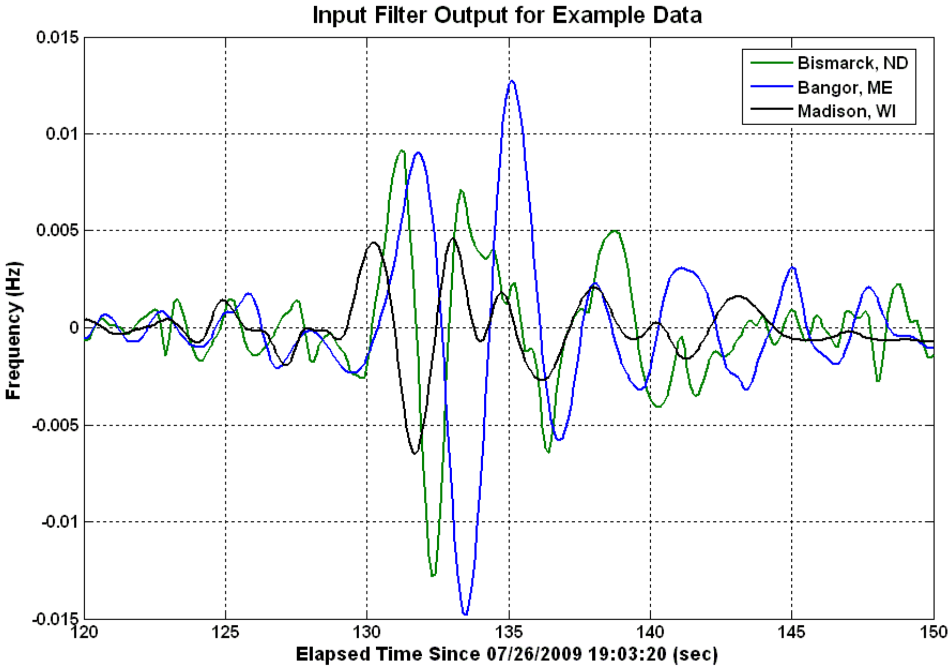


Figure 5.8: Output of Full Three Stage Filter for Example Data

The input filtering method employed here worked well under all test cases, properly extracting the interarea frequency band from the other components of the raw input data. Additionally it functioned well for several types of data. It is demonstrated here on frequency measurements derived from the FNET system but it performs similarly for FNET angle measurements and PMU data sets. Given any of these various types of data the filtering process returns a zero centered signal with the isolated interarea modes similar to that of Figure 5.8. The ability of this filtering process to work for various different types of data sets makes the final modal extraction procedure compatible with all these forms of input data. The remainder of the modal extraction procedure is tuned to handle data vectors similar to those of Figure 5.8, thus any data set that the input filter can reduce to that form can be processed.

The MATLAB code implementing this filtering process is presented in 0; Sections II, III, and IV.

5.2 EXTRACTION OF DOMINANT OSCILLATORY MODE

The focus of the modal extraction procedure detailed throughout this chapter is to create a description of a single oscillatory mode over time. One oscillation frequency is selected and the properties of that mode will be extracted over time for each measurement vector. In this fashion the evolution of the mode can be tracked. For historic data analysis this extraction will be focused around some particular event containing an oscillation. The dominant frequency in this oscillation will be extracted then tracked over time for a larger data set around the event.

In order to determine the dominant mode in the event, data from the corresponding time span is extracted. Once the necessary time span is determined the data for each measurement vector is extracted into a reduced data set. This data set is then filtered according to the procedure of Section 5.1. Doing this produces a set of signals that are zero centered and dominated by the oscillatory mode. Care should be taken during this step to specify the event time span such that the oscillation occupies the entire reduced data set. This serves to improve the estimation of the dominant mode.

With the event data extracted and filtered the vectors demonstrating the largest oscillation magnitude are selected. Since the oscillation and thus its amplitude has not yet been identified this is done by assessing the spread of the data points for each vector. As the vectors are now zero centered the extrema of the dominant mode will dictate the spread of the data. Here the spread of each vector is determined by difference between the 10% and 90% quantiles. Those vectors with the largest spread will be used to identify the dominant mode. Further reducing the data set in this fashion serves to ensure that measurement vectors which don't observe a prominent oscillation do not corrupt the results; additionally the data set and thus the necessary computational time is reduced. During development it was determined that the optimal number of vectors for mode extraction is between 5 and 10 depending on the nature of the data and size of the oscillation. For the example data seven FDRs demonstrating the largest spread were selected for extraction of the dominant mode. These seven filtered vectors are given in Figure 5.9. In this plot the interarea oscillation can be seen in the sinusoid wave at the beginning of the time span. It is also observed that Winnipeg, Bismarck, and Duluth appear to form a coherent group as they are oscillating in phase. Additionally the Carmel and Grand Rapids units form a coherent group oscillating against the Winnipeg group.

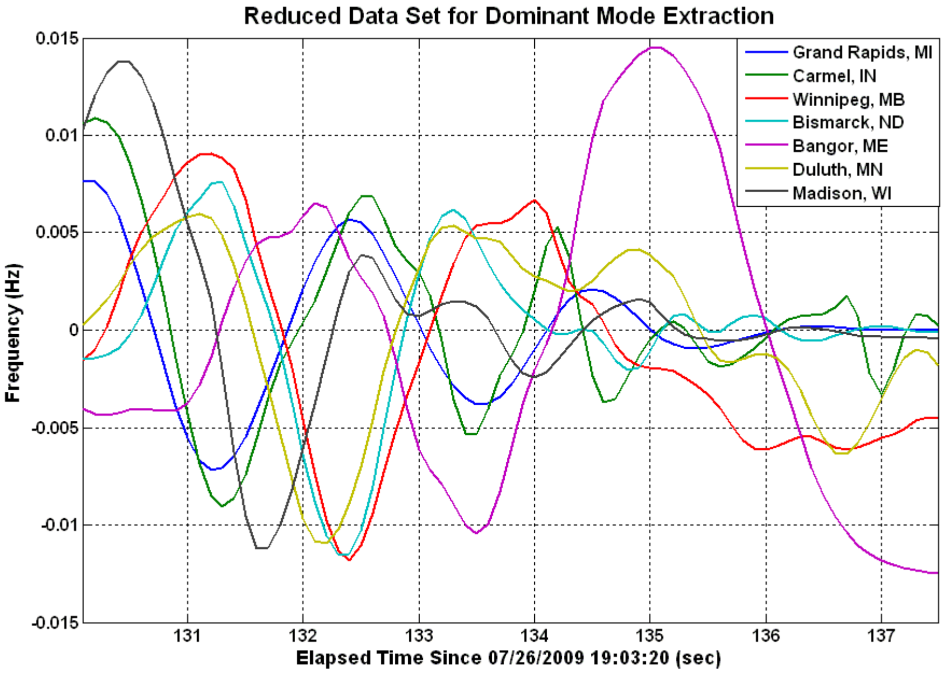


Figure 5.9: Filtered, Reduced Data Set during Oscillatory Event

The data of Figure 5.9 is derived from FNET measurements and as such the sampling frequency is 10 Hz. To improve the results of the upcoming Matrix Pencil algorithm the data needs to be resampled at a higher frequency. This was accomplished by interpolating additional data points to achieve a higher time resolution. For the example data set the sampling frequency was increased from 10 Hz to 100 Hz, by using a cubic spline fit to interpolate the new data points.

With the event data properly conditioned and with the time resolution increased the extraction of the dominant modal frequency can begin. In order to extract this mode a Matrix Pencil [31],[32] based analysis was employed across a sliding data window. Matrix Pencil analysis is a methodology for fitting a set of damped sinusoids to a signal. As the window is progressed across the data an independent Matrix Pencil procedure is performed for each measurement vector and each data window. The results for one of the Matrix Pencil fits on the example data set are given in Figure 5.10 and Table 5.2.

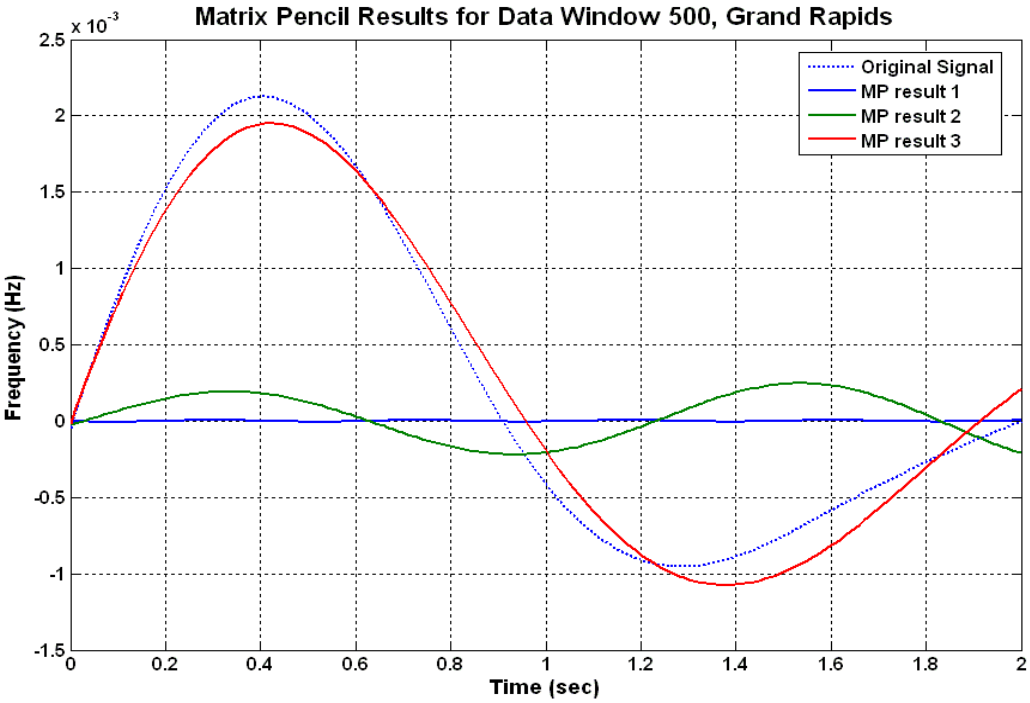
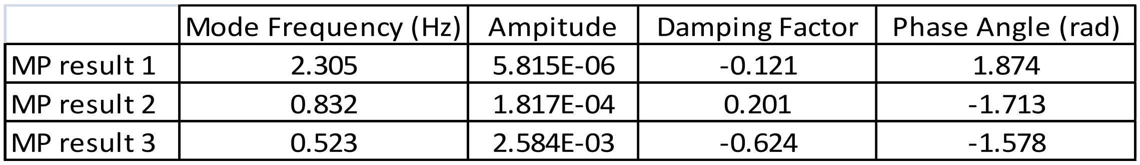


Figure 5.10: Matrix Pencil Decomposition Results



	Mode Frequency (Hz)	Amplitude	Damping Factor	Phase Angle (rad)
MP result 1	2.305	5.815E-06	-0.121	1.874
MP result 2	0.832	1.817E-04	0.201	-1.713
MP result 3	0.523	2.584E-03	-0.624	-1.578

Table 5.2: Matrix Pencil Decomposition Modal Parameters

Inspection of Figure 5.10 shows that one of the computed modes, MP result 3, closely matches the oscillation and captures the majority of the original signal. This result is typical of all the tested cases and is primarily due to the data detrending of the input filter. In all of the observed cases one of the computed modes followed the original signal very closely, with the remaining Matrix Pencil results accounting for the difference between the two. Due to the filtering process the interarea oscillations are now the most prominent components of the data signals, thus the computed mode which captures most of the signal trend also gives the frequency of the interarea oscillation for the associated window and measurement vector.

As seen in Figure 5.10 the third Matrix Pencil result matches the original signal most closely while the other two computed modes pick up the remaining residuals. This observation implies that the third result will necessarily have most energy and the first two with have smaller amplitudes and thus smaller energies by comparison as they account for error signatures. Hence the dominant mode for this window is given by the mode frequency that carries the most energy. The expression for a modal component is given by (5.1) where *A* is the amplitude, *α* is the damping, *f* is the frequency and *θ* is the phase angle.

$$y = Ae^{\alpha t}\cos(2\pi ft + \theta) \tag{5.1}$$

The power of this signal is (5.2); note that this equation does not give a true electrical power as it is derived from a frequency signal instead of a current or voltage.

$$P_y = y^2 \tag{5.2}$$

The total energy of (5.1) can then be expressed as the summation over time of the signal power (5.3). Once again this is not a true physical energy quantity, merely an analogous metric of the data signal.

$$E_y = \sum_t P_y \tag{5.3}$$

Whenever the Matrix Pencil method is performed on a measurement vector over a data window this energy metric is assessed for each resulting mode. The mode with the largest energy value is then selected as the representative mode for that data set. This process is repeated for each measurement vector and for each data window within the span of the oscillation event. Since a representative mode is determined for each of these iterations, giving many possible modal frequencies.

Use of the Matrix Pencil method and selection of the mode with the most energy in the signal has determined a list of candidates for the dominant mode. Each of these candidates represents one oscillatory frequency which prevailed during a specific data window on one of the measurement vectors. If there is a prevalent mode within the data it should appear many times in the set of candidates. This is reinforced by the fact that there is a large amount of overlap in the data windows meaning that consecutive windows should identify similar modes. The set of candidate modal frequencies can thus be treated as a probability distribution describing the location of the dominant mode. The distribution of candidate modes for the example data set is given in Figure 5.11. This distribution is typical of the Matrix Pencil results; most of the distribution mass is focused around the interarea band with other frequencies ranging up into the 3 Hz range. As there are no computed frequencies below 0 Hz this distribution represents an asymmetrical probability density function. From this data distribution it is wished to determine the most likely value which in this case is the mode frequency most often seen within the data set. Because of this asymmetrical nature the optimal estimator of position is the statistical mode [33].

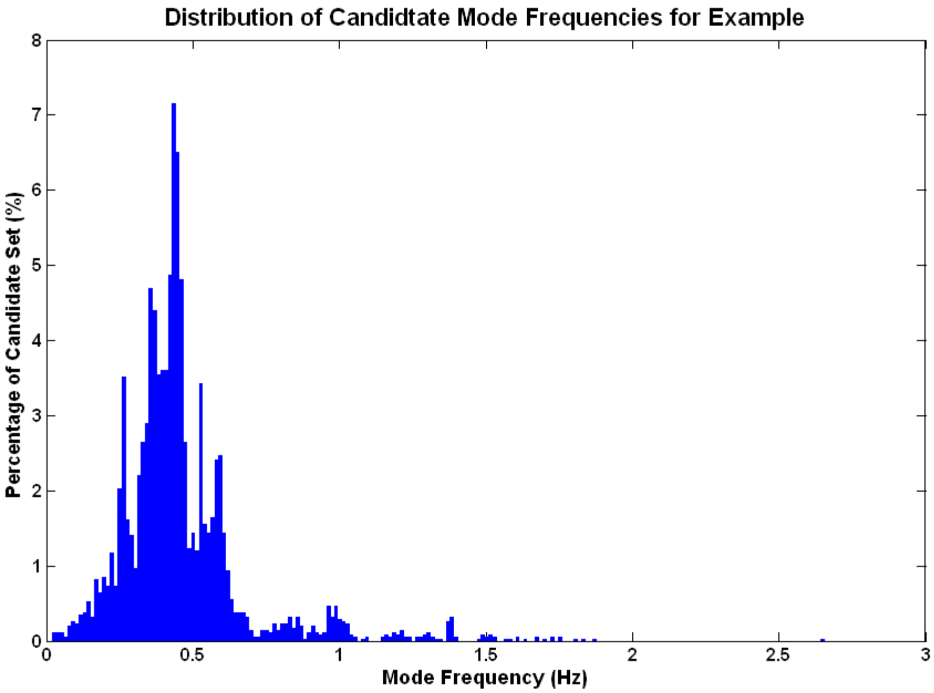
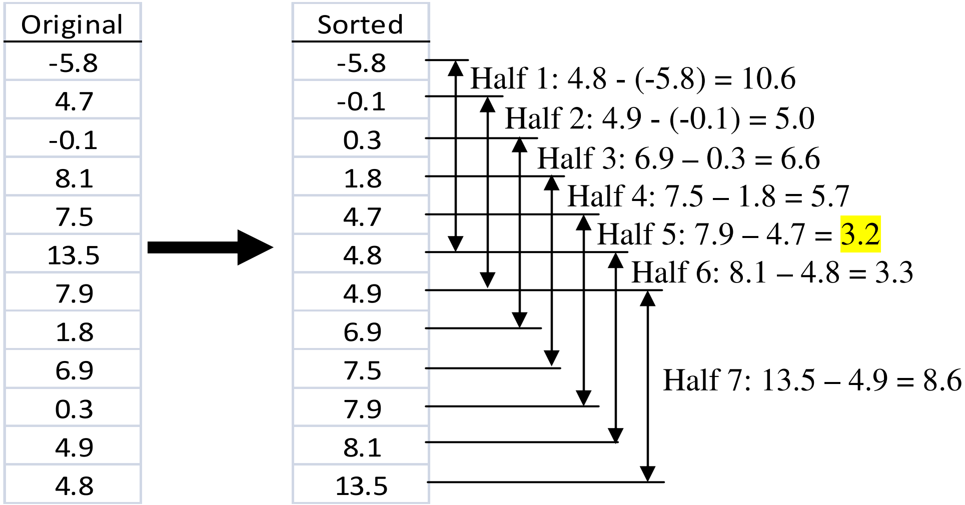


Figure 5.11: Histogram of Candidate Mode Frequencies

Determining the true statistical mode of a distribution requires a complete description of the probability density function, as this is unknown an estimator is needed. Several estimators of the statistical mode have been proposed based on the shortest half [34]. In order to determine the shortest half the data points are sorted then the shortest span which includes half of the data points is assessed. This procedure is demonstrated on an example data set in Figure 5.12. For this data set the shortest half of the data is Half 5 as it has the shortest width.



Original	Sorted
-5.8	-5.8
4.7	-0.1
-0.1	0.3
8.1	1.8
7.5	4.7
13.5	4.8
7.9	4.9
1.8	6.9
6.9	7.5
0.3	7.9
4.9	8.1
4.8	13.5

Figure 5.12: Example Computation of Shortest Half

This shortest half calculation was applied in an iterative fashion by first determining the shortest half of the existing distribution, this shortest half was then treated as the new data set and the process was repeated. Thus through each iteration half of the remaining candidate points were removed from the distribution, leaving only those centered around the most likely frequency value of the dominant mode. The iterative process was halted when the number of remaining candidates was less than four. The dominant mode was than selected as the average of these remaining points. In this fashion the most likely candidate for the dominant modal frequency during the event was determined. Executing this procedure on the set of example data determined a modal frequency of 0.431 Hz. This value is within the desired interarea band and when compared with the original data of Figure 5.1 it is consistent with the observed oscillation frequency.

The MATLAB code implementing the extraction and identification of the dominant mode is presented in 0; Sections V and VI.

5.3 WINDOWING FIT OF DOMINANT MODE TO FULL DATA SET

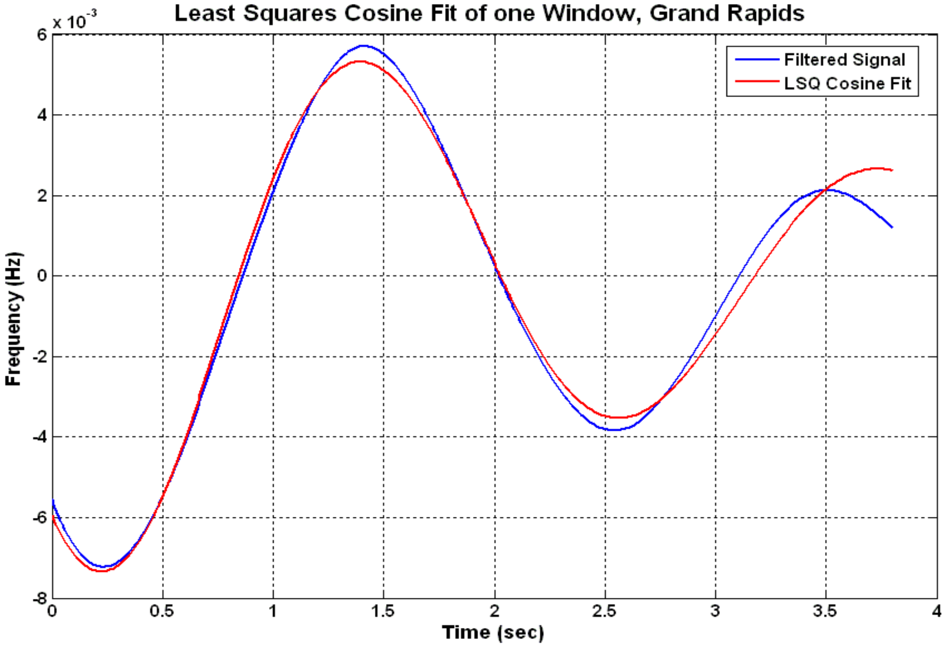
Now that the dominant oscillatory frequency has been determined it will be applied across the full length of the data set. Once again a moving data window will be employed to fit the mode across time. For each measurement vector and window instance the damping, angle and phase are derived which provide a best fit to the given data.

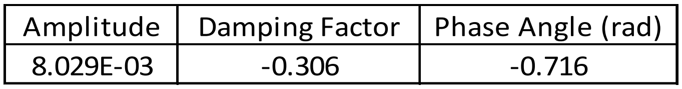
The moving data window is sized such that it covers approximately one period of the dominant oscillation frequency. The data window then starts at the first data point to be included in the output. It then begins moving across the full data set shifting one timestamp at a time until the end time is reach. For each instance of the window each measurement vector is filtered according to the process of Section 5.1. After filtering the data vectors are resampled using a cubic spline fit to increase the discrete sampling rate by a factor of ten. As before this resampling serves to increase the stability and accuracy of the final fit.

Once these conditioning steps have been performed the fit is ready to be executed. In this case it is wished to fit a damped sinusoid of a specified frequency (the dominant modal frequency determined previously). Because of this the Matrix Pencil approach becomes inappropriate as it cannot take a frequency value as an input. Instead a least squares fit of a damped sinusoid function was performed. This function is of the form in (5.4). Here the variable fit parameters are the amplitude, A, the damping factor, α, the phase, θ, and a DC offset, C. In (5.4) f_0 is the dominant modal frequency determined previously. Finally, t is the resampled time vector and y is the filtered, resampled data vector.

$$y = Ae^{\alpha t}\cos\left(2\pi f_0 t + \theta\right) + C \tag{5.4}$$

For this application MATLAB's *lsqcurvefit* function from the optimization toolbox was used [35]. This function provides a simple interface for performing a least squares fit to any type of known functional form. Performing this fit results in an optimal least squares solution to the four fit parameters. This solution gives the amplitude, damping and phase for the oscillatory mode of interest. An example of this least squares fit result is given in Figure 5.13.





Amplitude	Damping Factor	Phase Angle (rad)
8.029E-03	-0.306	-0.716

Figure 5.13: Least Squares Fit Example

The example of Figure 5.13 demonstrates one of the better results from this fitting procedure but in general the least squares fit produced good results considering it was inherently limited by only trying to fit a single sinusoid of a set frequency. As the sinusoidal frequency is derived to fit the majority of cases there are obviously a few outliers that do not fit well at the dominant modal frequency value.

As this filtering, resampling, and fitting procedure is iterated across the time and measurement vectors an evolution of the dominant mode is captured. The amplitude, damping and phase angles are computed with respect to the known, constant, mode frequency. These values may then be tracked both spatially and temporally to gain an understanding of the particular mode and the system response in general.

The MATLAB code implementing the damping cosine fitting procedure is presented in 0; Section VII.

5.4 IDENTIFICATION OF COHERENT GROUPS

With the mode angle and phase determined for each measurement point it is desired to estimate those units that form coherent groups. A coherent group here refers to a set of machines that are oscillating together and against one or more other groups of machines. Given a set of phasors computed from a mode the coherent groups can be identified by clustering the phasors according to their phase angle. If one group is oscillating in opposition to one other group then their phasors should be 180 degrees out of phase with each other. In the case of several different groups participating in the oscillation there will be several groups of phasors with each group occupying a distinct angle range on the unit circle.

In order to identify any coherent groups within the mode phasors a clustering algorithm based on the *k*-Means approach was implemented. Each cluster identified by this procedure will be treated as a coherent group. The *k*-Means procedure is described in [36] and repeated here. Given a data set of *n* points to be clustered the basic algorithm is:

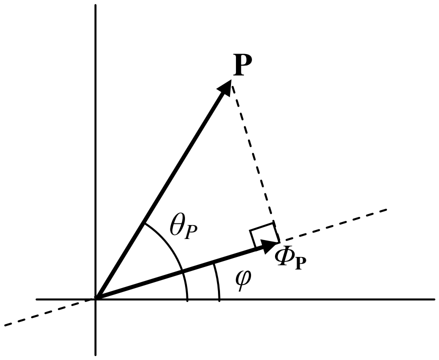
```
Initialize $k$, $\varphi_1$, $\varphi_2$, $\varphi_3$, … , $\varphi_k$
Do
    Classify $n$ samples according to nearest $\varphi_\mathrm{i}$
    Recompute $\varphi_\mathrm{i}$
Until no change in $\varphi_\mathrm{i}$
Return $\varphi_1$, $\varphi_2$, $\varphi_3$, … , $\varphi_k$
```

This algorithm assumes that the number of clusters within the data set, k, is known and picks an initial cluster centroid, φ_1, φ_2, φ_3, … , φ_k, for each of these clusters. It then assigns each point in the data set to the nearest cluster. The centroids are then recomputed to match the cluster assignment. This assignment and centroid recomputation process is repeated until the cluster centroids stabilize at their final value.

The algorithm described above is a general description of the procedure, the terms 'nearest' and 'recompute' need to be defined with respect to the data set and desired solution. For this application the coherent groups responding to an interarea mode needed to be identified. As each group will be oscillating against the other group(s) in the system the different groups should demonstrate differing phase angles. In other words measurement points from the same group should have similar modal phase angles and measurements from different groups should be out of phase with each other. The desired cluster centroids in this case become phase angles defining the center of each coherent group. For this reason a clustering method based on the mode phase angle was selected.

Due to the numerical nature in which the modal content was computed implementation of a clustering algorithm based entirely on the phase angle, will skew the results. A phasor will still be computed for any point in the system not participating in the interarea mode, but it will have small amplitude relative to other parts of the system. If the clustering is based solely on phase angle all measurements receive the same weighting in determination of the coherent groups. In order to produce a more appropriate fit those areas demonstrating stronger modal content should receive greater weight in the clustering process. For this reason the vector projection was selected over phase angle as clustering criteria because it includes the phasor amplitude acting as a weighting factor.

Given a phasor, **P**, which belongs to a cluster with centroid *φ*, the projection of **P** onto *φ*, $\Phi_{\mathbf{P}}$, is given by (5.5).



$$\Phi_{\mathbf{P}} = \left|\mathbf{P}\right| \cos(\theta_P - \varphi) \tag{5.5}$$

In the ideal case a phasor should have the same phase angle as its associated cluster centroid, giving zero classification error. When this happens the angle difference is zero and the cosine term is maximized which in turn maximizes (5.5). This projection function will be used by the *k*-Means classification step to determine to which cluster a phasor is nearest to by assigning it to the cluster onto which it produces the largest projection. Additionally the recompute step of the *k*-Means algorithm needs to select the centroid of a cluster in a manner which maximizes the projections of its component phasors.

Defining the total projection of a cluster, Φ_{TOT}, as the summation of projections from each of the member phasors, {$\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, …, $\mathbf{P}_M$}, gives (5.6).

$$\Phi_{TOT} = \sum_{i=1}^{M} \Phi_{\mathbf{P}_i} = \sum_{i=1}^{M} \left|\mathbf{P}_i\right| \cos(\theta_i - \varphi) \tag{5.6}$$

Selecting *φ* to maximize (5.6) yields the cluster centroid which provides a best fit to all of the member phasors. Since this is a non-linear equation dependant on *φ*, determining the location of the global maximum becomes an iterative solution.

Using Newton's Method to find the stationary points of (5.6) takes the form of (5.7).

$$\varphi_{n+1} = \varphi_n - \frac{\Phi'_{TOT}(\varphi_n)}{\Phi''_{TOT}(\varphi_n)} \tag{5.7}$$

Substituting (5.6) into (5.7) yields (5.8).

$$\varphi_{n+1} = \varphi_n - \frac{\dfrac{\partial}{\partial \varphi_n}\left(\sum_{i=1}^{M} \left|\mathbf{P}_i\right| \cos(\theta_i - \varphi_n)\right)}{\dfrac{\partial^2}{\partial {\varphi_n}^2}\left(\sum_{i=1}^{M} \left|\mathbf{P}_i\right| \cos(\theta_i - \varphi_n)\right)} \tag{5.8}$$

Carrying the derivatives in (5.8) through produces (5.9)

$$\varphi_{n+1} = \varphi_n + \frac{\sum_{i=1}^{M} \left|\mathbf{P}_i\right| \sin(\theta_i - \varphi_n)}{\sum_{i=1}^{M} \left|\mathbf{P}_i\right| \cos(\theta_i - \varphi_n)} \tag{5.9}$$

Where φ_n is an estimated location of a stationary point of Φ_{TOT} and φ_{n+1} is a more accurate estimate. (5.9) can be used in an iterative process with the new estimate computed in one iteration loop used as the input to the successive loop. This process continues until the change in φ is sufficiently small. Iterating in this fashion finds the closest local extrema to the given starting point. With an initial guess sufficiently close, the location of the global maximum can be determined. This location is then assigned as the centroid of the cluster consisting of the phasors $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \ldots, \mathbf{P}_M\}$.

Using the phasor projections to determine the nearest cluster and using Newton's Method to recompute the cluster centroids refines the above *k*-Means algorithm as follows:

```
Initialize $k$, $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_k$
Do
    For each phasor
        Assign phasor to $\varphi_i$ that produces a maximum projection (5.5)
    EndFor
    For each cluster centroid, $\varphi_i$ of $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_k$
        Determine new $\varphi_i$ using Newton's Method with (5.9)
    EndFor
Until no change in $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_k$
Return $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_k$
```

The number of clusters identified by this process up to this point has been a fixed input value thus the preceding algorithm is incapable of handling cases with an unknown number of clusters. Meaning that the number of coherent groups must be known before computing the location of the groups and to which group each phasor belongs. This limitation hinders the analysis for the interarea oscillations which may involve an unknown number of generator groups. Extensions will now be made to the algorithm to resolve this issue.

A basic principle of a *k*-Means style algorithm is that as the number of clusters is increased the classification improves. With a greater number of clusters and a data set of constant size each cluster is required to match a smaller number of points. As the number of points in a cluster decreases the classification error between each point and the cluster centroid decreases as well. This principle extends up to the case where the number of clusters is equal to the number of data points. When this happens each data point is assigned its own cluster, producing zero classification error with respect to the training set. The implemented method for identification of an unknown number of clusters exploits this principle.

The clustering algorithm starts by determining two clusters within the data using the *k*-Means procedure above. Once the two clusters stabilize to their final values the quality of each cluster is assessed by attempting to split it into two subclusters using the *k*-Means procedure again. If the cluster with the largest angle difference between its subclusters exceeds a predefined criterion then it is split in two, effectively incrementing the number of clusters that are identified. The new set of clusters is then solved and the process is repeated for the next value of *k*. This iteration continues until all clusters produce an acceptable subcluster angle difference. The preceding process is given in the algorithm below which extends the previous one.

```
Initialize endLoop = unset, $k = 2$, $\varphi_1$, $\varphi_2$
Do
    For each phasor
        Assign phasor to $\varphi_i$ that produces a maximum projection (5.5)
    EndFor
    For each cluster centroid, $\varphi_i$
        Determine new $\varphi_i$ using Newton's Method with (5.9)
    EndFor
    If no change in $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_k$ Then
        For each cluster, $\varphi_i$
            Run $k$-Means algorithm on $\varphi_i$ phasors with $k$ fixed at 2
            Store resulting centroids as subclusters 1 and 2
            Determine angle difference between subclusters 1 and 2
        EndFor
        Find cluster, $\varphi_j$, with largest angle difference
        If angle difference of $\varphi_j$ exceeds tolerance Then
            Increment k (adding another data cluster)
            Set $\varphi_j$ to subcluster 1 centroid
            Set $\varphi_k$ to subcluster 2 centroid
        ElseIf angle difference of $\varphi_j$ is within tolerance Then
            Set endLoop flag
        EndIf
    EndIf
Until endLoop flag is set
Return $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_k$
```

This procedure will now be demonstrated through a series of figures that illustrate how it determines clusters. The set of phasors used for this example was derived from the response to a generation trip occurring in the eastern interconnect. This set of phasors gives the amplitude and phase of a 0.638 Hz mode for several measurement points in the system. This mode falls into the range of Kundur's second interarea definition and as such it is expected to demonstrate several coherent groups. The phasor directions are distributed around the unit circle with a few arcs containing a higher concentration of phasors. The set of phasors partitioned in the example is given in Figure 5.14.

Figure 5.14 also gives the initialization steps on the clustering process, the number of clusters, *k*, is set at two and a centroid is selected for each cluster. The two resulting centroids are given by the red and blue dashed lines. Next the *k*-Means process continues by assigning each phasor to the nearest cluster and recalculating the cluster centroids. Figure 5.15 shows the results from the first *k*-Means iteration. The heavily dashed lines are the original centroids for cluster 1 (blue) and cluster 2 (red). Each phasor was assigned to the nearest cluster as indicated by the color coding. The centroids for each cluster were then recomputed using the Newton's Iteration procedure described above, producing the new centroids indicated by the lines with the lighter dashing.

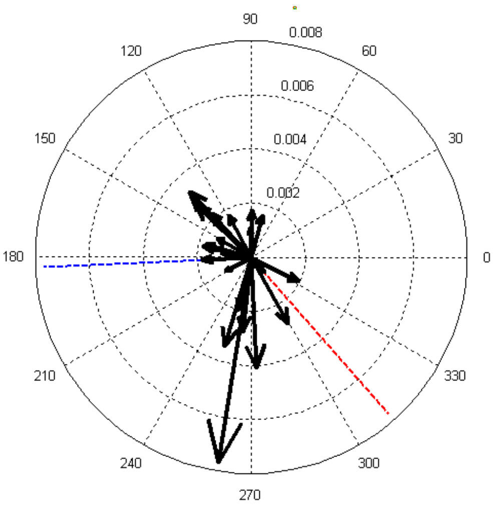


Figure 5.14: Initialization of k = 2 and Cluster Centroids

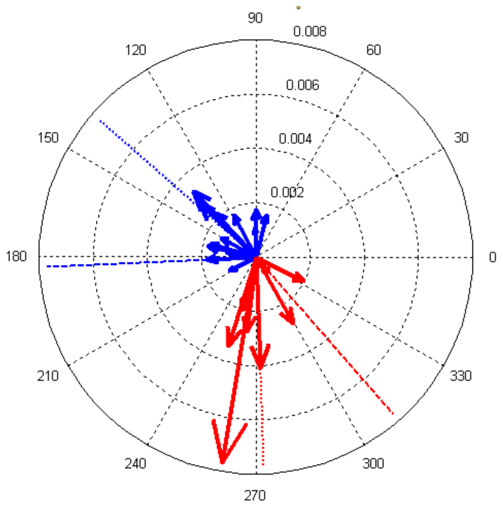


Figure 5.15: Assignment of Phasors to Clusters and Recalculation of Centroids

With the centroid repositioning of Figure 5.15 some of the phasors changed clusters thus additional iteration were required to compute the new centroids. Within a few iterations the cluster assignments stabilized and the centroids obtained their final values as shown in Figure 5.16. The subclustering analysis was then performed, it was determined that the red cluster had the greatest definition between subclusters. The identified subclusters are given by the lightly dashed red line and the lightly dashed black line (near 270 degrees). The difference between the subcluster centroids exceeded the tolerance of 45 degrees so the red group of Figure 5.16 needed to be split. This split was performed in Figure 5.17 as the black cluster was split out of the red one based on proximity to the two identified subclusters.

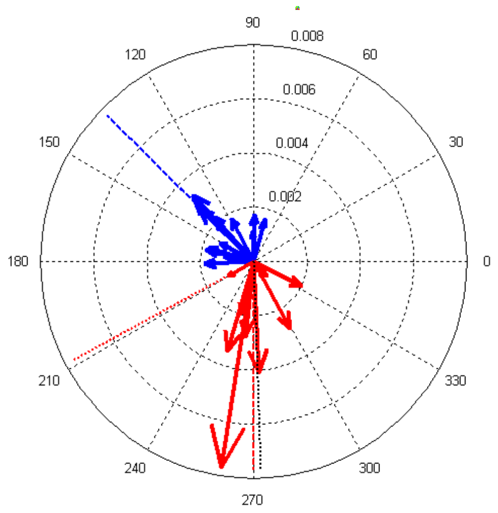


Figure 5.16: Partitioning of Red Cluster into Subclusters

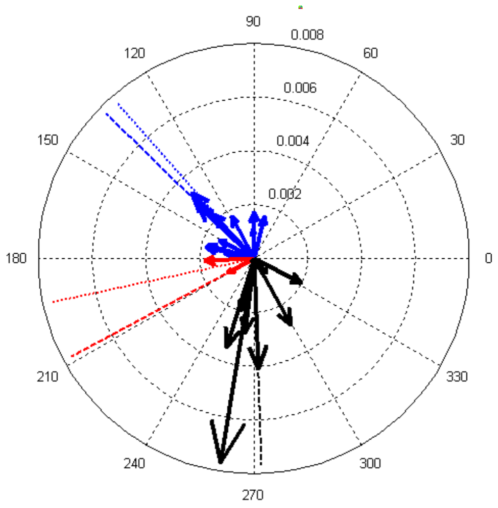


Figure 5.17: Creation of New Cluster from Subclusters and Reassignment

Performing this split increased the value of *k* from 2 to 3. The *k*-Means algorithm was then executed with this new set of clusters. Throughout this set of iterations the red cluster picked up some phasors from the blue one as seen in the progression from Figure 5.17 to Figure 5.18 and finally to the stable values in Figure 5.19, in each case the heavier dashed line represents the previous centroid and the lighter dashed line gives succeeding centroid. Once the stable point had been reach the quality of each cluster was assessed again and it was determined that the black cluster exceeded the tolerance and needed to be split. With the proposed additional cluster located at the green line of Figure 5.19.

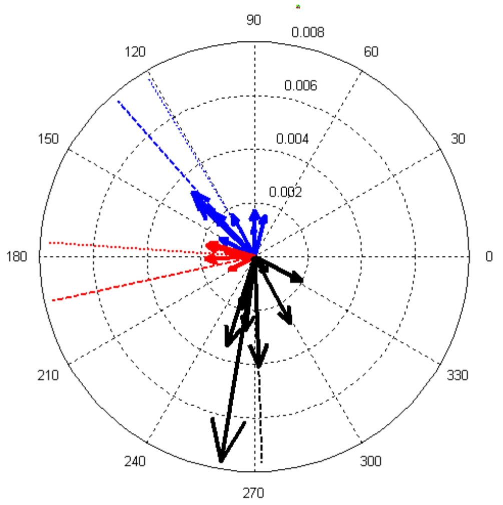


Figure 5.18: Iterating with k = 3 to Determine Cluster Centroids

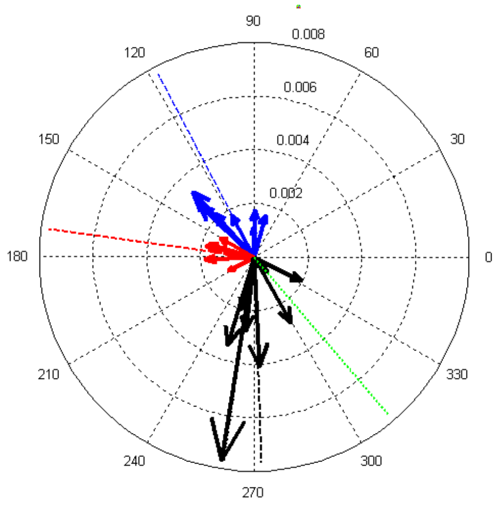


Figure 5.19: Partitioning of Black Cluster after Stabilization with k = 3

After a split of the black cluster and subsequent *k*-Means iterations the cluster stabilized to Figure 5.20. Once again it was determined that a cluster needed to be split, in this case the blue one. The resulting set of five clusters stabilized to the final positions of Figure 5.21, at this point it was found that the quality of all the clusters was within acceptable tolerance and the iteration procedure could be stopped.

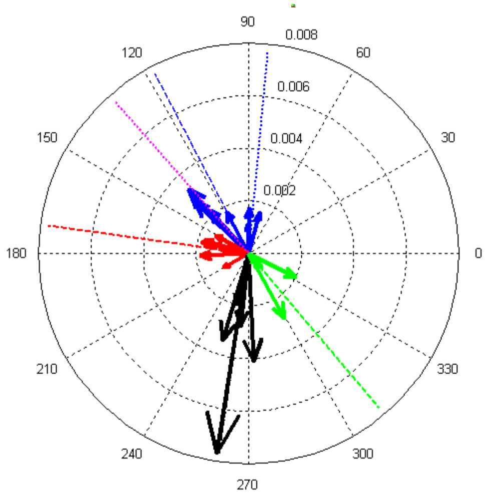


Figure 5.20: Partitioning of Blue Cluster after Stabilization with k = 4

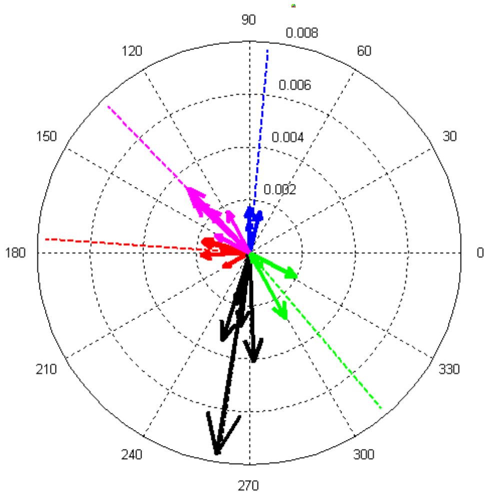


Figure 5.21: Final Cluster Assignment and Associated Phasors

The MATLAB code which enacts this clustering algorithm is given in 0; Section VIII.

5.5 RENDERING AND VISUALIZATION OF RESULTS

A visualization package was developed to display the results from the preceding modal identification procedure. This visualization is based around a geographic display which gives the mode amplitude, damping and phase at each measurement point. These phasors are placed on the map so that the coherent groups can be observed spatially. Two other plots are included in this visualization a phasor diagram and a time series of the source data. The phasor diagram places all of the phasors from the map at the same origin giving a clearer comparison of their amplitudes and the relative directions of the coherent groups. The time series plot at the top of the window gives an overview of the oscillation event and the time position of the phasor displays.

This process was wrapped around the developments of Sections 5.1 through 5.4 in order to present the results. In the first step the raw input data was filtered according to procedure outlined in Section 5.1. Given this filtered data set and an event region the identification of the dominant mode proceeds according to the process outlined in Section 5.2. At this point the most likely candidate for the dominant mode has been identified. The flow chart for the first few steps of the process is given in Figure 5.22. The remainder of the full process is given in Figure 5.23. This flow chart depicts the steps involved with fitting the mode frequency, clustering the resulting phasors and plotting the results. Taking Figure 5.22 and Figure 5.23 together the entire process for modal extraction and visualization is described.

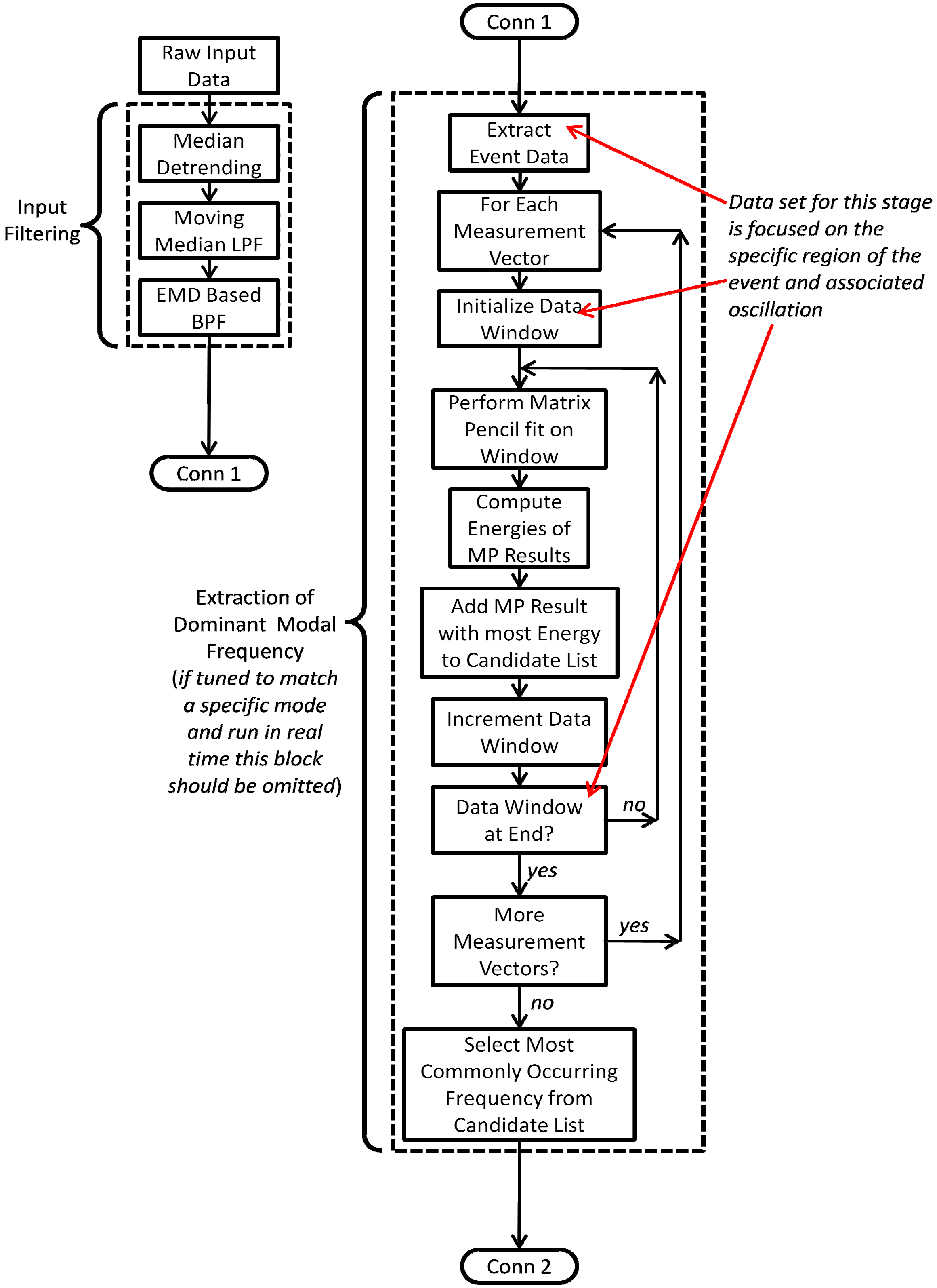

Raw Input Data
Median Detrending
Moving Median LPF
EMD Based BPF
Input Filtering
Conn 1
Conn 1
Extract Event Data
For Each Measurement Vector
Initialize Data Window
Perform Matrix Pencil fit on Window
Compute Energies of MP Results
Add MP Result with most Energy to Candidate List
Increment Data Window
Data Window at End?
no
yes
More Measurement Vectors?
yes
no
Select Most Commonly Occurring Frequency from Candidate List
Conn 2
Data set for this stage is focused on the specific region of the event and associated oscillation
Extraction of Dominant Modal Frequency (if tuned to match a specific mode and run in real time this block should be omitted)

Figure 5.22: Flow Chart for Filtering and Mode Identification Process

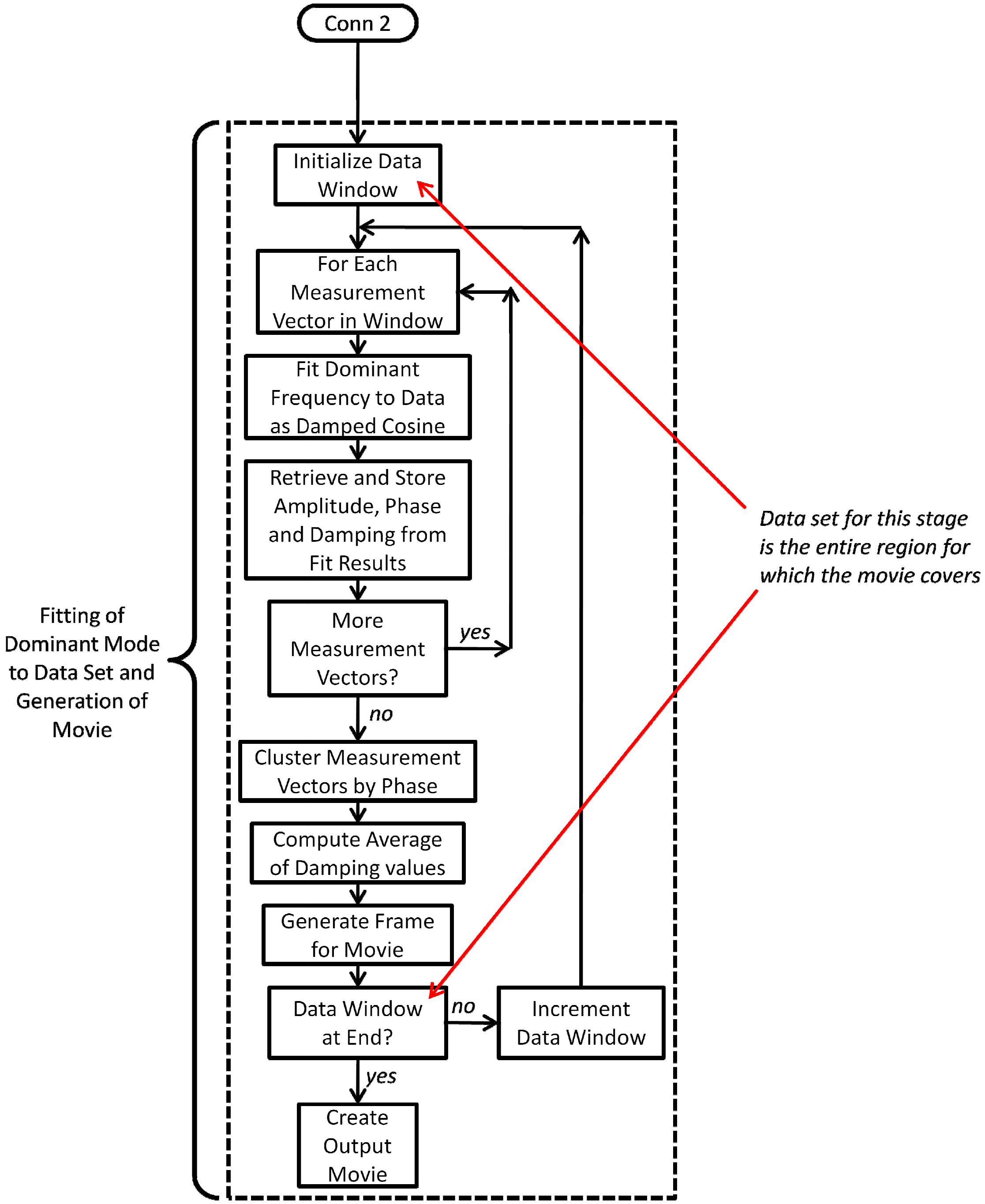

Conn 2
Initialize Data Window
For Each Measurement Vector in Window
Fit Dominant Frequency to Data as Damped Cosine
Retrieve and Store Amplitude, Phase and Damping from Fit Results
More Measurement Vectors?
yes
no
Cluster Measurement Vectors by Phase
Compute Average of Damping values
Generate Frame for Movie
Data Window at End?
no
Increment Data Window
yes
Create Output Movie
Fitting of Dominant Mode to Data Set and Generation of Movie
Data set for this stage is the entire region for which the movie covers

Figure 5.23: Flow Chart Describing Fitting and Movie Generation

This visualization architecture was used to generate several movies which run across a given oscillation to present the graphics over time such that the temporal evolution of the mode can be observed. One such movie is represented below by a series of frame captures. This movie was constructed from a data set centered on the example case used throughout this chapter. As stated previously, this example is drawn from FDR measurements which capture the system response to a generation trip at the Donald C. Cook nuclear plant located in southwestern Michigan. This event occurred at about 19:05:30 UTC (15:05:30 EDT) on July 26, 2009.

Figure 5.24 gives a movie frame from before the event occurred and demonstrates a region with very little oscillatory content. The plot at the top of the frame gives the raw measurement data for the seven FDRs demonstrating the largest magnitude oscillations. The time range highlighted in red is the region for which the cosine fit has been performed. This plot is intended to provide a timeline of the oscillation event under study as well as giving the time point which the computed phasors correspond to.

The main body of the frame of Figure 5.24 is made up of the geographic region spanned by the power system with each mode phasor plotted at the location of its corresponding measurement point. The phasor is plotted as an arrow with the length indicating the amplitude and the direction giving the phase angle. In this example all of these mode phasors have been referenced against the unit located in Bangor, ME to prevent them from rotating as a group and to keep their direction relatively stationary. Bangor was selected as the reference in this instance because it demonstrated a strong oscillation throughout the event and thus the fit results and its phase angle were more consistent throughout the time span of interest. The phasors in this frame are also color coded by the coherent group to which they belong.

In the lower left corner of Figure 5.24 the modal frequency being extracted is given along with the average damping factor seen across the measurement points of the system. The damping values used to compute the average are drawn directly from the results of the cosine fitting operation. For a negative value of damping factor the oscillation is decaying on average at the metered points in the system. If this value is positive the oscillation magnitude is growing throughout the system. Thus if the mean damping factor is positive for an extended period of time the system will go unstable as the oscillations are growing without bound. To the left of the average damping read out is a traffic light style indicator. When the damping is greater than zero this light turns red specifying an alarm condition as the oscillation is growing. If the oscillation is lightly damped ($-0.1 < \alpha \leq 0$) the light turns yellow. Finally this indicator is green when the oscillation is heavily damped ($\alpha < -0.1$).

The final feature of Figure 5.24 is a phasor diagram in the lower right-hand corner which plots all of the mode phasors on the same origin so that the clustering and relative magnitudes are better demonstrated. In this diagram the dashed lines represent the centroids of the computed clusters. Once again the color coding represents the identified coherent groups.

As indicated by the location of highlighted time span Figure 5.24 is drawn from a time point before the inciting event. At this stage the system is in steady state and not experiencing an interarea oscillation. Because of this all of the mode phasors have a very small magnitude and register as nearly zero in the diagrams.

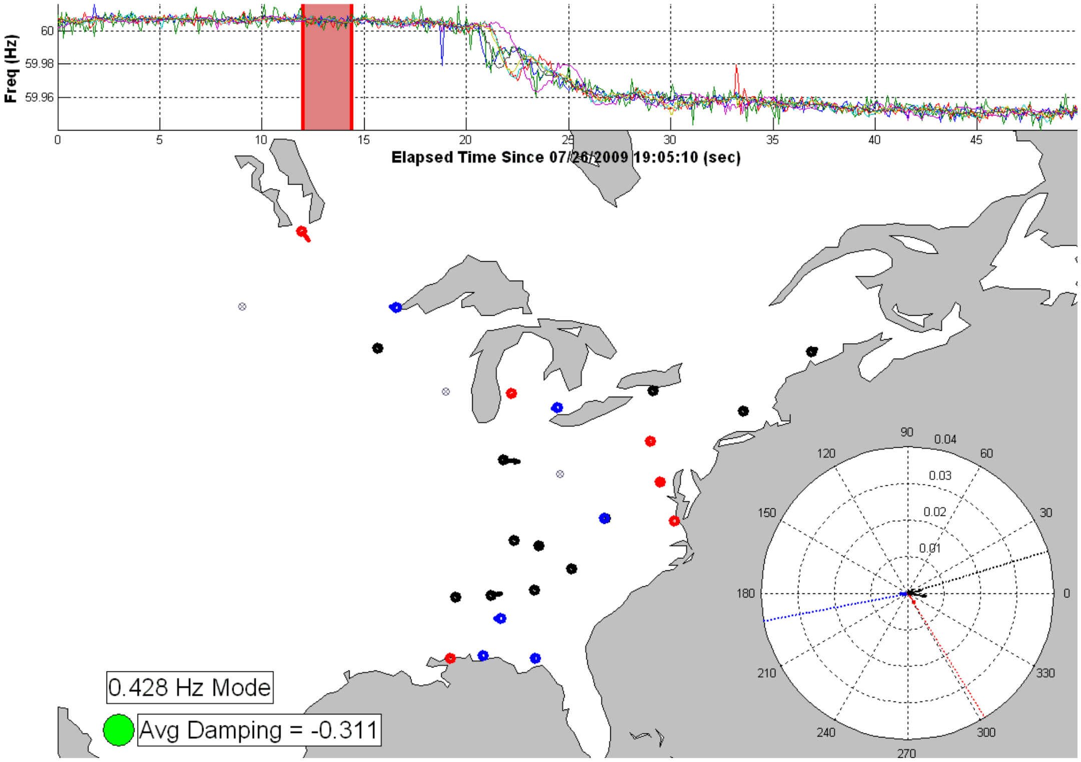


Figure 5.24: Pre-Event Condition

Figure 5.25 was captured near the time at which the inciting event occurred. As seen by the highlighted time span the oscillation is beginning to grow with most of the FDRs seeing the first quarter cycle of the interarea oscillation. There is no oscillation present at the beginning of the time frame under inspection so the amplitudes of the fitted cosine functions are still small. However, the end of the data window demonstrates a sizeable oscillation and as a result most of the computed damping factors are positive. This trend is seen at nearly all of the measurement points as their phasor have small amplitude. The positive damping trend is also indicated by the average damping factor value of +0.719. Since this damping factor is greater than zero the indicator light has turned red. At this point the coherent groups are also taking shape with New England and the Southeast oscillating against the Great Lakes Region.

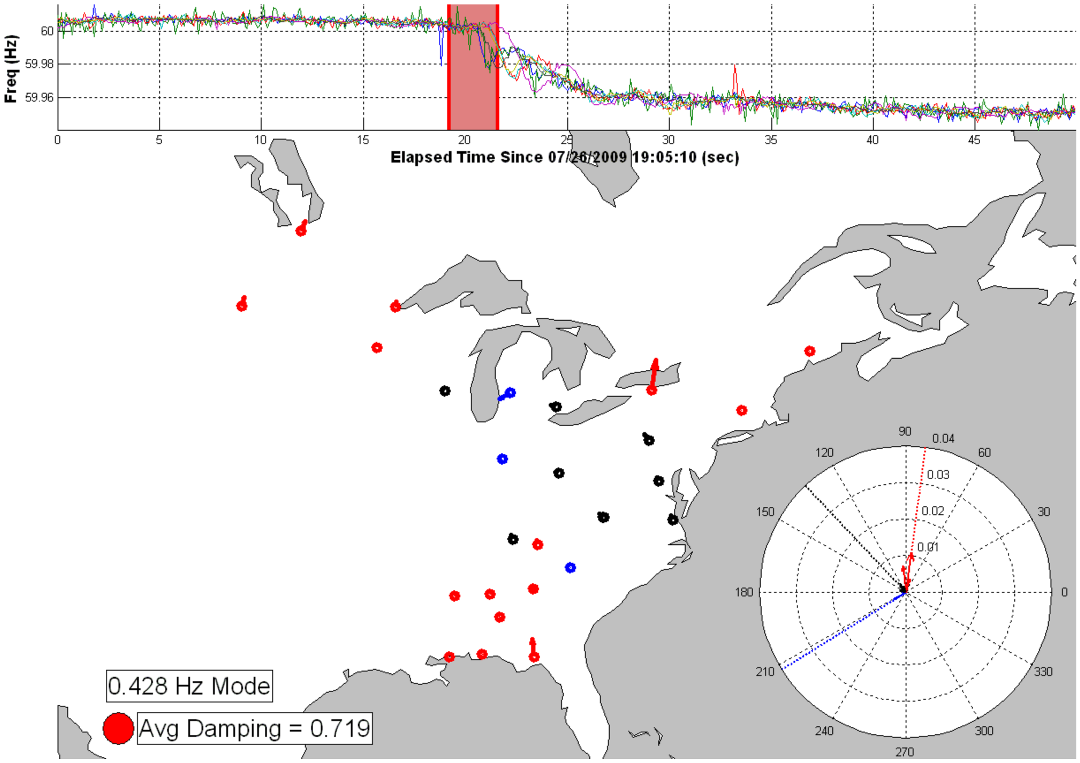


Figure 5.25: Beginning of Oscillation

As time progresses to Figure 5.26 the interarea oscillation is fully developed. Some measurement points begin registering significant amplitude values for their mode phasors. These higher amplitudes are seen by the Great Lakes units as well as the unit located in Bangor. Many of the other units have just barely started to see oscillations and as such they are registering low amplitude and positive damping as in Figure 5.25. Because of this disparity the average damping across the system is still positive but is less than half of that computed in Figure 5.26. The clustering algorithm has identified two major coherent groups at this point, the first is the two New England units in red and the second covers the Great Lakes regions in blue. For the most part these two groups are oscillating 180 degrees out of phase. The other units are of sufficiently small amplitude such that the classification procedure becomes unreliable and they are classified throughout the two groups and into a third and fourth group (the black and green ones).

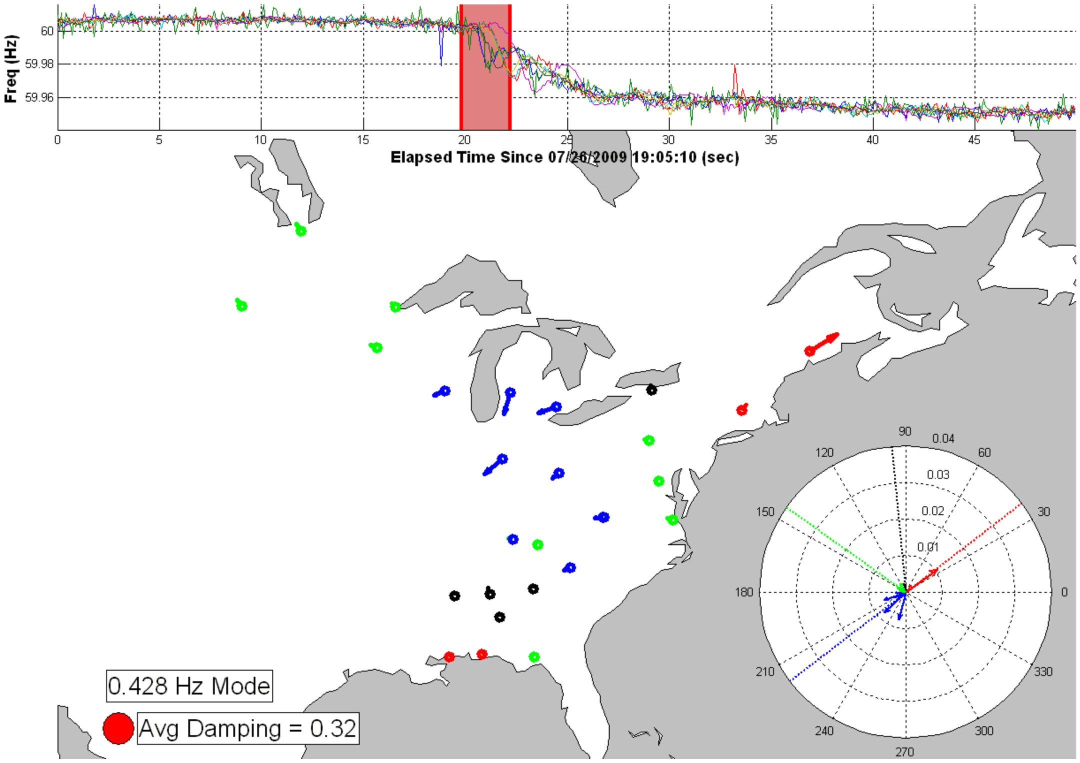


Figure 5.26: Initial Cycle of Oscillation

The interarea oscillation has grown to its largest magnitude by Figure 5.27, here nearly all of the measurement points are registering significant amplitude and the phasors are all observable on the map. Also at this point the oscillation magnitude has begun to decay, indicating damping across the system. The average damping factor is -0.902 indicating that the oscillation is decaying sharply throughout the system. At this point the two primary groups are defined by the Bangor, ME unit and the Bismarck, ND units which are oscillating nearly 180 degrees out of phase. New England and the South (the black group) are oscillating together against the rest of the system, primarily the Great Lakes and Northern Plains Regions (red group). A few other units are distributed about halfway between these two groups (green and blue units).

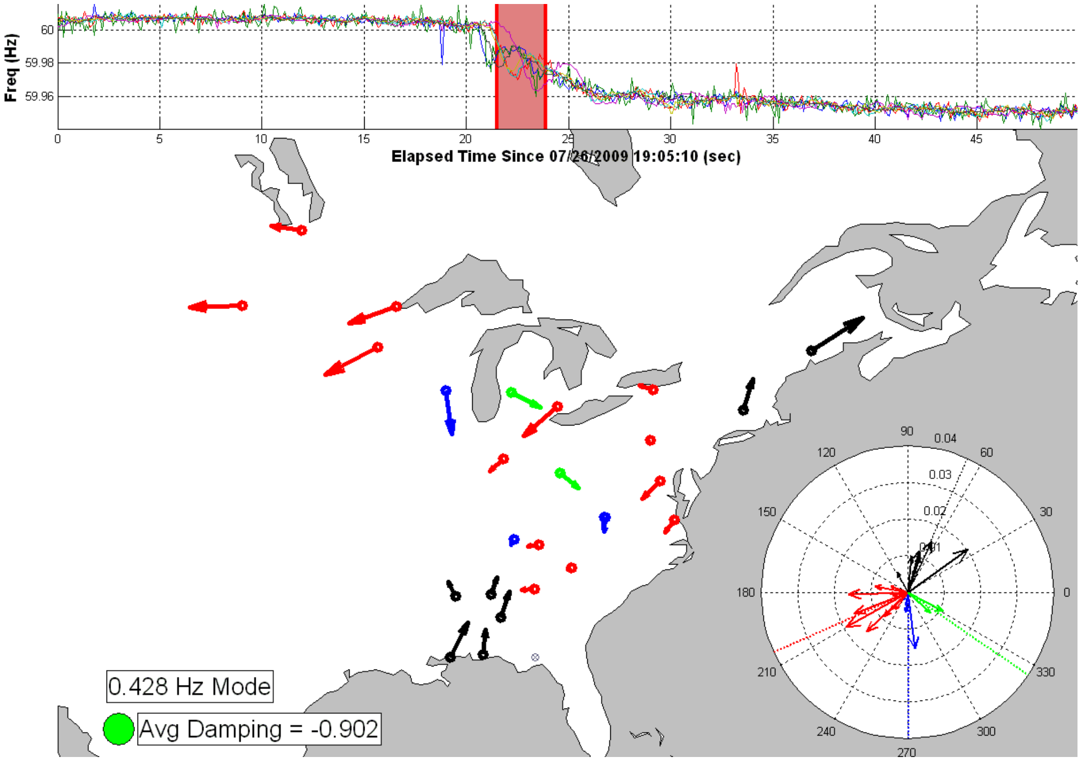


Figure 5.27: Oscillation Reaches Largest Magnitude

As time progresses through to Figure 5.28 the oscillation continues to decay. As a result the majority of the phasors in this capture are smaller than they were in Figure 5.27. The average system damping factor here has achieved -0.636 which for this case should completely damp the oscillation in a few cycles. The New England units are still forming a coherent group with the Southern US as characterized by the blue and pink groups. This group is now oscillating against the rest of the system most notably the Northern Plains area as seen by the black group of units. A third group oscillation halfway between the other two has been identified by the clustering algorithm and is shown in red and green. A side note here is that the coloring of the groups is not consistent between frames, for example the black group of Figure 5.27 is now covered by the blue and pink groups.

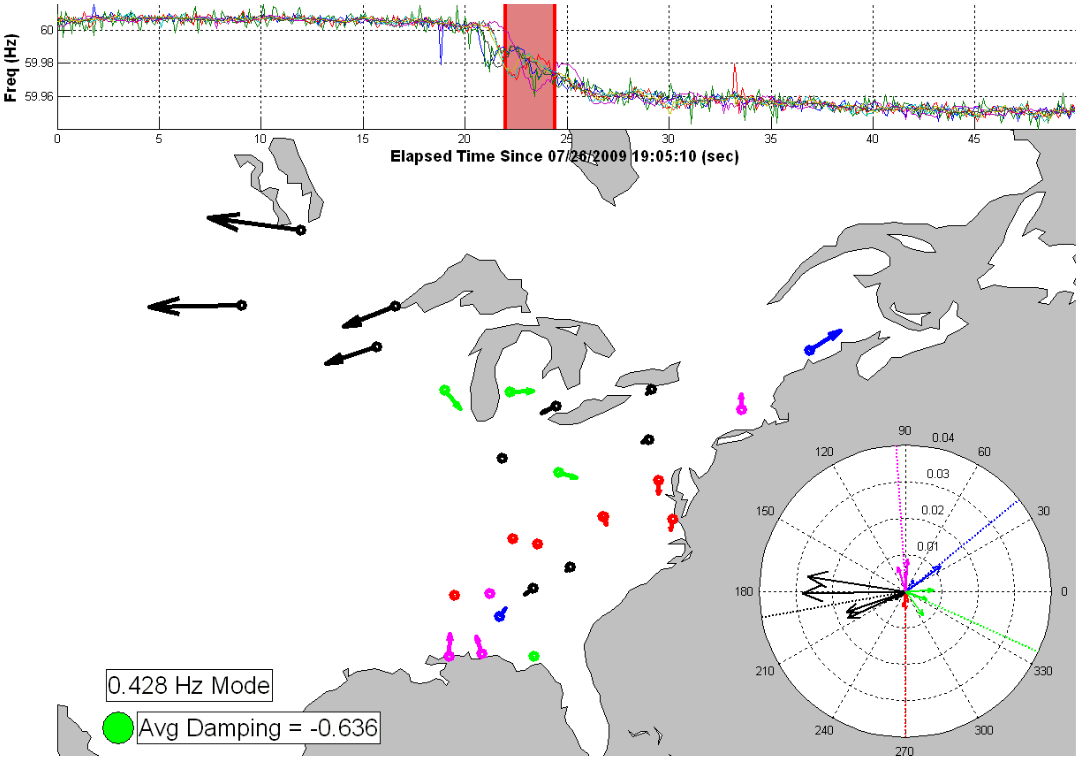


Figure 5.28: Continued Damping of Oscillation

The interarea oscillation continues to die out through Figure 5.29. By this point most of the phasors across the central portion of the US are demonstrating little to no amplitude. The two New England units in the black group are still oscillating with a relatively large magnitude. They are now swinging against the blue group whose most prominent member is the Bismarck, ND unit. The rest of the units in the system have low amplitude oscillations and as such they are classified into several unimportant groupings. As with Figure 5.28 the average damping factor indicates that this trend will continue and the oscillation will decay until the system approaches its steady-state condition.

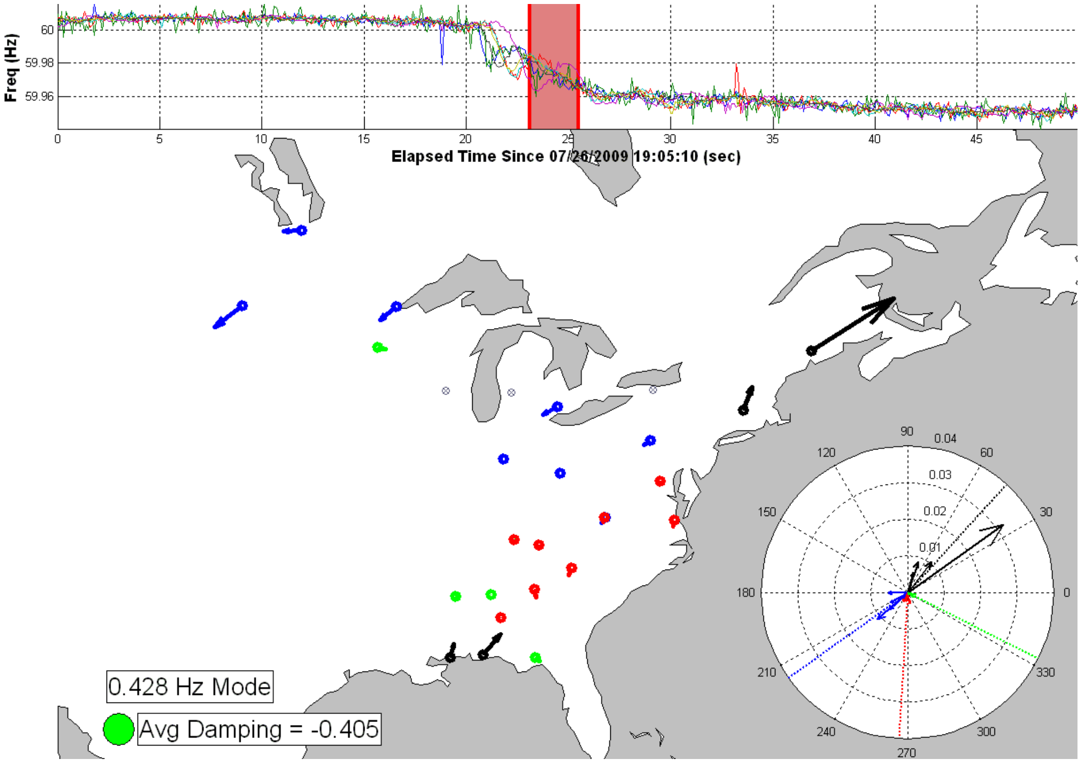


Figure 5.29: Final Decay of Oscillation

The oscillation has almost completely died out by the time that the movie approaches the timespan of Figure 5.30. At all of the measurement points the mode phasors have little to no amplitude. Additionally the computed dampings do not expect them to be growing substantially over the next cycle. The average system damping is still negative but approaching zero, this is not necessarily an alarm but due mainly to the fact that the system is not demonstrating any appreciable oscillation. From this it is obvious that the oscillation has run its course and the system has achieved its new steady-state value.

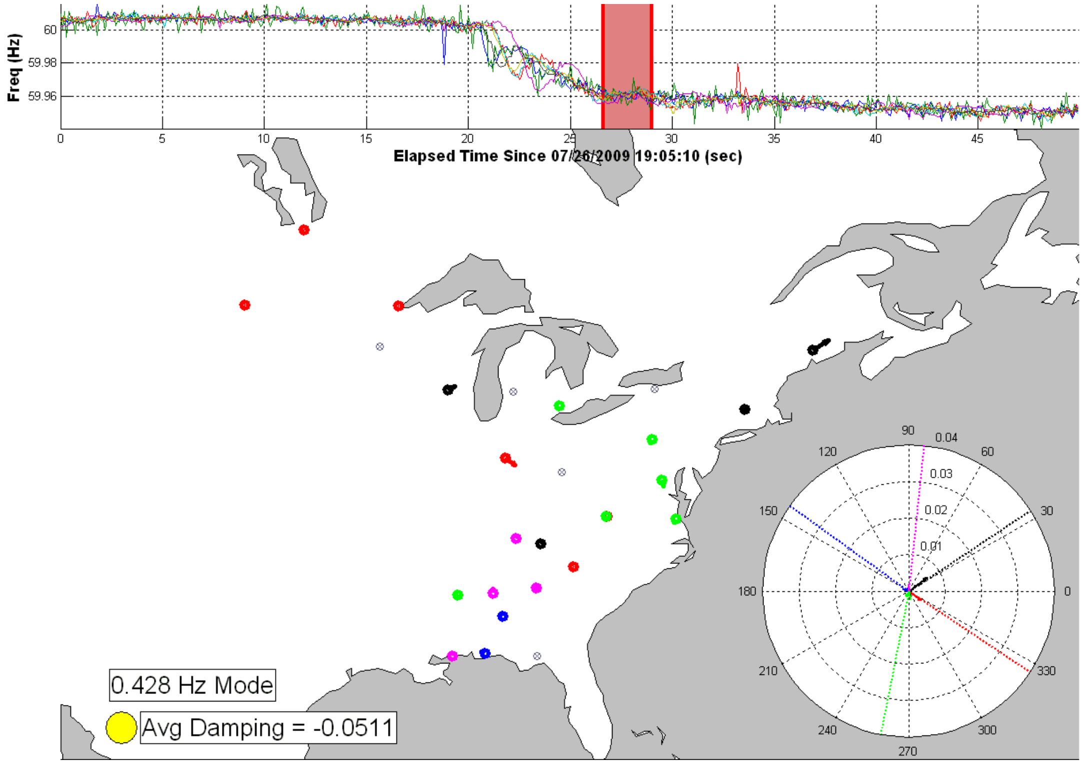


Figure 5.30: System Approaches Steady State

The visualizations and movie creation described in this section was performed in MATLAB, the supporting code is a part of the main modal extraction process and is contained within the code of 0; Section I. The individual frames are rendered as MATLAB figures and then imported into to an *.avi file to create the movie. The geographic mapping and coastlines of the displays were achieved with the use of M_Map [37], which is a set of MATLAB functions implementing a mapping toolbox. M_Map is freely available at http://www.eos.ubc.ca/~rich/map.html.

5.6 CONCLUSIONS

Throughout this chapter a procedure for identifying and tracking an interarea mode has been established. Given a set of wide area measurement data this algorithm is capable of identifying the dominant mode and creating a visualization based on the results. The input data is first detrended and filtered to isolate the interarea frequency band. With this accomplished the most likely modal frequency is extracted from the filtered data. Next this frequency is fitted as a damped cosine waveform to each measurement vector and data window across the time span of interest. The results of these successive fitting operations produce the amplitude, phase angle and damping of the mode for each measurement at each time point within the data window. These results then feed a clustering algorithm which attempts to determine the coherent groups within the system. All of these results are then used to generate a movie which depicts the evolution of the interarea mode over time.

An example data set has been presented along with a detailed, step by step depiction of the entire process, from raw data through generation of the movie output. For the example data set an oscillation of 0.428 Hz was identified. This oscillation demonstrated an interarea mode with New England swinging against the rest of the Eastern Interconnect. This New England mode has been estimated in several previous studies [21] from distinct data sets. These studies generally place the mode frequency in the 0.35 Hz to 0.45 Hz band depending on the prevailing system conditions and measurement characteristics. The computed value of 0.428 falls within this band validating the dominant mode extraction procedure.

The implemented algorithm was tested with several other data sets to verify its stability; frames from some of the resulting movies are presented in APPENDIX D. Data sets based on voltage angle measurements were also tested as input. It was found that the input filtering process was capable of reducing the raw angle measurements to the form expected by the remainder of the algorithm. This indicates that this same code should be able to monitor both oscillations in phasor angle as well as frequency. Since the frequency is essentially the derivative of voltage angle and the same modes are encountered (with a 90 degree phase shift between the two types of data sets). This was verified through comparison of results from frequency and voltages angle measurements of the same oscillatory event.

Several possible improvements can be made to this extraction and visualization process. Firstly the rendering and display format can be refined. The movies and captures presented here were design mainly to convey the results of the data analysis; improvements can be made to enhance the overall display. Additionally, the current method of coloring the groups for the display is inconsistent across time frames and should be addressed in order to produce a better visualization package. The damped cosine fitting procedure produces unstable results for a non-trivial number of the attempted fit operations (about 10%), especially when the interarea mode is not a prominent feature of the data. Future research opportunities exist to improve the method in which the dominant modal frequency is fitted and the manner in which the results are processed. Other areas which may benefit from additional research include tuning of the input filter and the group identification procedure.

5.7 POTENTIAL APPLICATION: REAL-TIME OSCILLATION MONITORING

This process was designed with an eye toward a final application as real-time oscillation monitoring tool. The basic principle is that one or more oscillatory modes are known for the system to be monitored. The dominant mode identification step can then be omitted and the fitting, clustering and visualization can be tuned to monitor a given modal frequency that is endemic to the power system in question. As data is read in real time this frequency is fitted to the data and the results are presented to the user. In this manner a specific mode can be continuously monitored. If it begins to flare up it can be seen in real time and any necessary control action can be taken if it persists for an extended period of time or begins to grow in an unstable fashion. Multiple oscillatory modes can then be tracked through several instances of the same application with each one tuned on a specific interarea frequency.

Development of this real-time application would require several changes to the algorithm presented here. Firstly, as mentioned the dominant mode extract procedure can be removed as the target frequency is already known. The input filtering process will also need refinement as the EMD is using future values which would not be available in real-time. Without these future values the EMD results are adversely affected and the filtering process may need modification to counter act this effect.

CONCLUSIONS AND FINAL REMARKS

The properties of propagating electromechanical waves in power systems were investigated thought the research detailed here. The ultimate goal is to achieve a better understanding of the electromechanical properties of a power system and how they affect different portions of that system. A basic description of this phenomenon was introduced along with some preliminary observations from measurement data. This work was continued through computer simulations of simple systems. From these simulations several observations were made about the effects of different system parameters on wave propagation speeds. Although these studies provided useful insight they were inherently limited as they involved extremely simple system architectures which failed to capture the large scale effects of the process under study.

To remedy this limitation a simplified modeling architecture was developed based on the primary physical equations which provided an easily scalable system. With this development the investigative process was extended to power systems containing hundred of busses. Stepping up to these larger systems allowed a more complete picture of the wave propagation. With this tool in hand the effect of system parameters on wave propagation could be more completely detailed. It was found that increased inertia and line impedance served to slow a wavefront as it moves across a large system. Additionally it was found that increased machine damping serves to limit the magnitude of an electromechanical wave but does not affect its propagation speed.

An application of this modeling tool was then developed. Using measurement data from an existing system the simplified dynamic model was applied. The objective here is to develop a reduced order model which provided a best fit to the measurement data and thus attempted to approximate the original system. By predefining a system architecture the parameters of the required model could be estimated through a least squares fit to the source measurement data. The necessary equations for this process were developed by solving the swing equation and DC power flows in reverse, i.e. solving for the system parameters instead on the state variables. This process was found to be extremely effective when used to reduce the simplified model architecture.

In order to increase the usability of this reduction technique it was then applied to a data set drawn from a PSS/E simulation. As this data represents an AC system it provided a more appropriate test case to asses the results when reduction of a real-world system is attempted. Additionally a prediction process was implemented for this system. The prediction procedure uses the known system state and the reduced order model to identify upcoming system conditions. This prediction was found to work relatively well especially under steady-state conditions. The largest errors were encountered in transient regions where an event was taking place on the system and the predictor did not have enough up to date information available to it. As expected increasing the prediction time frame was found to exacerbate these errors. Considering this the predictor performed well but has room for improvement including more accurate modeling strategies incorporating AC power flow equations and various load models. It was found that dynamically adjusting the parameters of the reduced system should provide benefits.

Depending on the accuracy of the final predictive modeling procedure this work has several possible applications which can provide benefits to system security and stability. One benefit is increased situational awareness and prediction of system dynamic response for the operators. Additionally this process can help to feed advanced closed loop control systems in which the control actuators begin working to arrest a situation before it becomes a problem. This would rely on the predictive model estimating system conditions several timesteps ahead of time and looking for unstable swings, violation of equipment ratings, underdamped modes, voltage collapse and other unstable conditions within the predicted values. When these unstable cases are identified the necessary control actions can be initiated further in advance to limit the severity of any consequences.

The final topic presented covers the identification and extraction of interarea modes. The primary feature of this process is a non-linear filtering procedure which serves to isolate an interarea oscillation from the other data features. After filtering the most commonly occurring interarea frequency is identified and then tracked over time. Next the determination of amplitude, phase and damping for the oscillatory mode becomes a simple fit to a sinusoid. These parameters are computed for each of a series of data windows. In this method the evolution of the mode is tracked over time and visualization is developed which gives the properties of the mode at each point on a map and identifies any existing coherent groups.

Several examples of this extraction procedure are presented and in all tested case it produced effective results. The goal of this research is to use this algorithm with a large catalog of historic measurement data to establish the dominant modes for a given power system. Through this analysis the most frequently occurring modal frequencies can be identified along with their associated coherent groups. Additionally this process has potential applications as a real-time situational awareness and stability assessment tool.

REFERENCES

[1] P. Kundur, "*Power System Stability and Control*", McGraw-Hill Inc., 1994.

[2] J.S. Thorp, C.E. Seyler, and A.G. Phadke, "*Electromechanical Wave propagation in Large Electric Power Systems*", IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications, Vol. 45, No. 6, June 1998.

[3] R.M. Gardner, J.K. Wang, and Y. Liu, "*Power System Event Location Analysis Using Wide-Area Measurements*", IEEE Power Engineering Society General Meeting, 2006. 18-22 June 2006.

[4] A.J. Arana, J.N. Bank, R.M. Gardner, Y. Liu, "*Estimating Speed of Frequency Disturbance Propagation through Transmission and Distribution Systems*", IEEE Power Systems Conference and Exposition, 2006. 29 Oct – 1 Nov 2006.

[5] J.N. Bank, R.M. Gardner, J.K. Wang, A.J. Arana, Y. Liu, "*Generator Trip Identification Using WAMS and Historical Data Analysis*", IEEE Power Systems Conference and Exposition, 2006. 29 Oct – 1 Nov 2006.

[6] J.N. Bank, R.M. Gardner, S.S. Tsai, K.S. Kook, Y. Liu, "*Visualization of Wide-Area Frequency Measurement Information*", IEEE Power Engineering Society, General Meeting, 2007. 25 June – 28 June 2006.

[7] B. Qiu, L. Chen, et.al. "*Internet based frequency monitoring network (FNET)*", IEEE Power Engineering Society, Winter Meeting, 2001. Vol. 3, 28 Jan.-1 Feb. 2001 pp 1166 - 1171.

[8] J.K. Wang, R.M. Gardner, Y. Liu, "*Analysis of System Oscillations Using Wide-Area Measurements*", IEEE Power Engineering Society, General Meeting, 2007. 25 June – 28 June 2006.

[9] Z. Zhong, C. Xu, et.al. "*Power System Frequency Monitoring Network (FNET) Implementation*", IEEE Trans. on Power Systems, Vol. 20, Issue 4, Nov. 2005, pp 1914 - 1921.

[10] C.L. Phillips, and J.M. Parr, "*Signals, Systems, and Transforms*", Prentice Hall, New Jersey, 1999.

[11] J.P. Yang, G.H Cheng, and Z. Xu, "*Dynamic Reduction of Large Power System in PSS/E*", IEEE/PES Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005.

[12] R. Belhomme, H. Zhao, M. Pavella, "*Power System Reduction Techniques for Direct Transient Stability Methods*", IEEE Transactions on Power Systems, Vol. 8, No. 2, May 1993.

[13] P.V. Kokotovic, and P.W. Sauer, "*Integral Manifold as a Tool for Reduced-Order Modeling of Nonlinear Systems: A Synchronous machine Case Study*", IEEE Transactions on Circuits and Systems, Vol. 36 No. 3, March 1989.

[14] D.C. Chaniotis, and M.A. Pai, "*Model reduction in Power Systems Using Krylov Subspace Methods*", IEEE Transactions on Power Systems, Vol. 20, No. 2, May 2005.

[15] J.N. Franklin, "*Matrix Theory*", Dover Publications Inc., Mineola, New York, 1993.

[16] D.T. Lee, and B.J. Schachter, "*Two Algorithms for Constructing a Delaunay Triangulation*", International Journal of Computer and Information Sciences, Vol. 9, No. 3, 1980, pp. 219-242.

[17] B.C. Pal, "*Robust Damping of Interarea Oscillations with Unified Powerflow Controller*", IEE Proceedings- Generation, Transmission and Distribution, vol. 149, pp. 733-738, Nov. 2002.

[18] P. Kundur, M. Klein, G.J. Rogers, and M.S. Zywno, "*Application of Power System Stabilizers for Enhancement of Overall System Stability*", IEEE Transactions on Power Systems, 1989, 4, (2), pp.6 14-626.

[19] J.F. Hauer, "*Applications of Prony Analysis to the Determination of Modal Content and Equivalent Models for Measured Power System Response*", IEEE Transactions on Power Systems, Vol. 6, No. 3, August 1991.

[20] J.F. Hauer, W.A. Mittelstadt, K.E. Martin, et. al., "*Use of the WECC WAMS in Wide-Area Probing Tests for Validation of System Performance and Modeling*", IEEE Transactions on Power Systems, Vol. 24, No. 1, February 2009.

[21] J.H. Chow, L. Vanfretti, A. Armenia, et. al. "*Preliminary Synchronized Phasor Data Analysis of Disturbance Events in the US Eastern Interconnect*", IEEE Power System Conference and Exposition, 2009, IEEE PES 2009, 15-18 March 2009, pp. 1-8.

[22] J. Dong, T. Xia, Y. Zhang, L. Wang, Y. Liu, L. Beard, T. Bilke, "*Wide-Area Measurements of Three North America Interconnections at Distribution Level*", IEEE Power Systems Conference and Exposition, 15-18 March 2009, pg. 1-8.

[23] C.G. Boncelet, Jr. , "*Order Statistic Distributions with Multiple Windows*", IEEE Transactions on Information Theory, Vol. 37, Issue 2, March 1991, pg 436 – 442.

[24] E. Ataman, V.K. Aatre, K.M. Wong, "*Some Statistical Properties of Median Filters*", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 29, Issue 5, Oct 1981, pg 1073 – 1075.

[25] D.S. Laila, A.R. Messina, B.C. Pal, "*A Refined Hilbert-Huang Transform With Applications to Interarea Oscillation Monitoring*", IEEE Transactions on Power Systems, Vol. 24, No. 2, May 2009.

[26] N. Senroy, "*Generator Coherency Using the Hilbert-Huang Transform*", IEEE Transactions on Power Systems, Vol. 23, No. 4, Nov 2008.

[27] N.E. Huang, Z. Shen, S.R. Long, M.C. Wu, S.H. Shih, Q. Zheng, C.C. Tung, and H.H. Liu. "*The empirical mode decomposition method and the Hilbert spectrum for non-stationary time series analysis*", Proc. Roy. Soc. London, A454:903-995, 1998.

[28] H. Liang, Z. Lin, and R.W. McCallum. "*Artifact reduction in electrogastrogram based on the empirical model decomposition method*", Med. Biol. Eng. Comput., 38:35-41, 2000.

[29] N.E. Huang, M.-L. Wu, W. Qu, S.R. Long, and S.P. Shen. "*Applications of Hilbert–Huang transform to non-stationary financial time series analysis*", Appl. Stochastic Models Bus. Ind., 19:245-268, 2003.

[30] S.T. Quek, P.S. Tua, and Q. Wang. "*Detecting anomalies in beams and plate based on the Hilbert–Huang transform of real data*", Smart Material Structure 12:447-460, 2003.

[31] T. K. Sarkar and O. Pereira, "*Using the Matrix Pencil Method to Estimate the Parameters of a Sum of Complex Exponentials*", IEEE Antennas and Propagation Magazine, Vol. 37, No. 1, pp 48-55, 1995.

[32] Y. Hua and T. K. Sarkar, "*Matrix Pencil Method for Estimating parameters of exponentially damped/undamped sinusoids in noise*", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 36, No. 2, pp. 228-240, 1988.

[33] J.S. Milton, J.C. Arnold, "*Introduction to Probability and Statistics*", Fourth Edition, McGraw-Hill Inc., 2003.

[34] D.F. Andrews, P.J. Bickel, F.R. Hamper, P.J. Huber, W.H. Rogers, J.W. Tukey, "*Robust Estimates of Location: Survey and Advances*", Princeton University Press, Princeton, New Jersey, 1972.

[35] The MathWorks Inc., *MATLAB help page for lsqcurvefit function*, MATLAB Version 7.6.0.325 (R2008a), Optimization Toolbox Version 4.0.

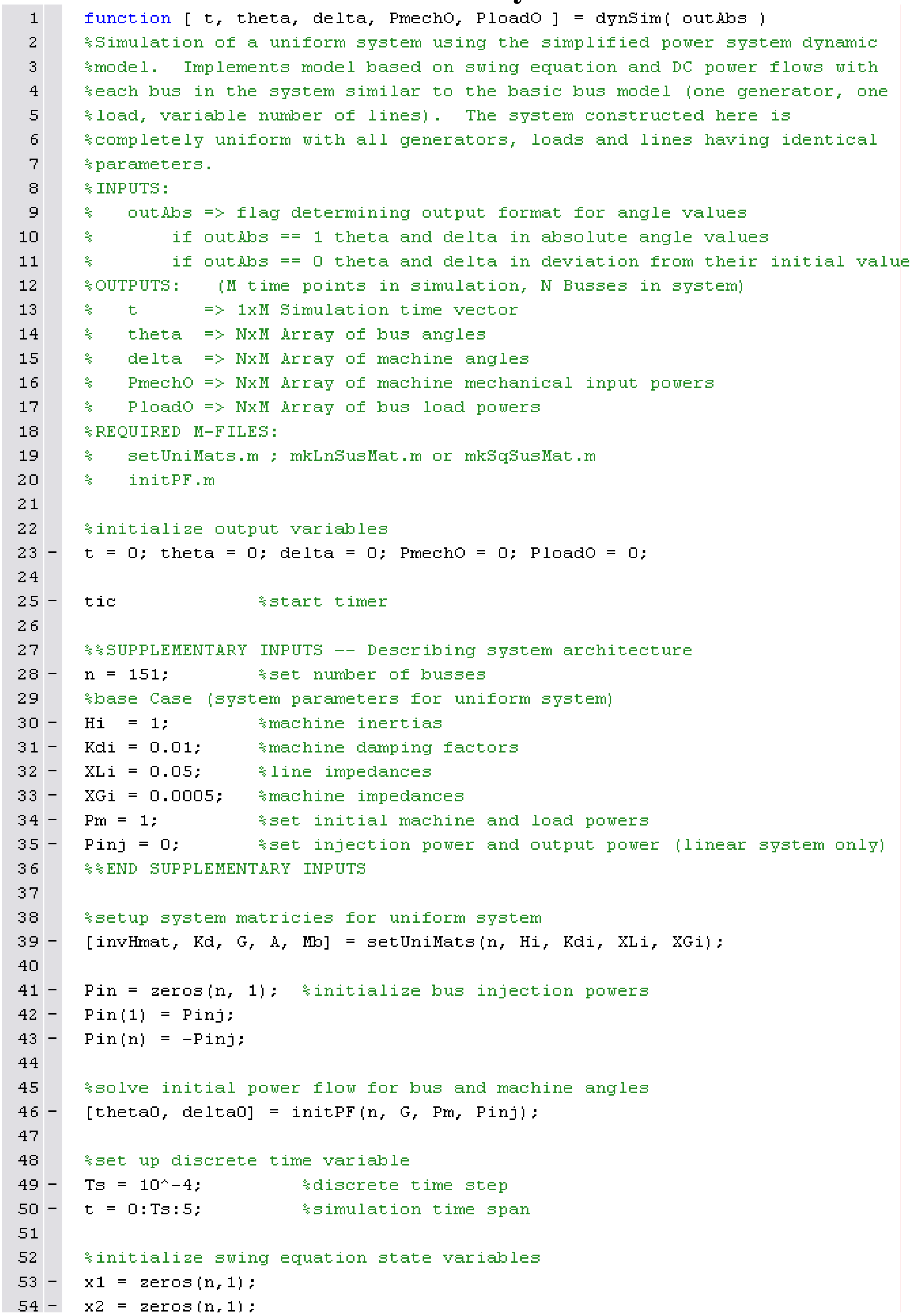
[36] R.O. Duda, P.E. Hart and D.G. Stork, "*Pattern Classification*", John Wiley & Sons Inc., 2001.

[37] R. Pawlowicz, "*M_Map: A Mapping Package for Matlab*", http://www.eos.ubc.ca/~rich/map.html, Ver 1.4e, Oct 2009.

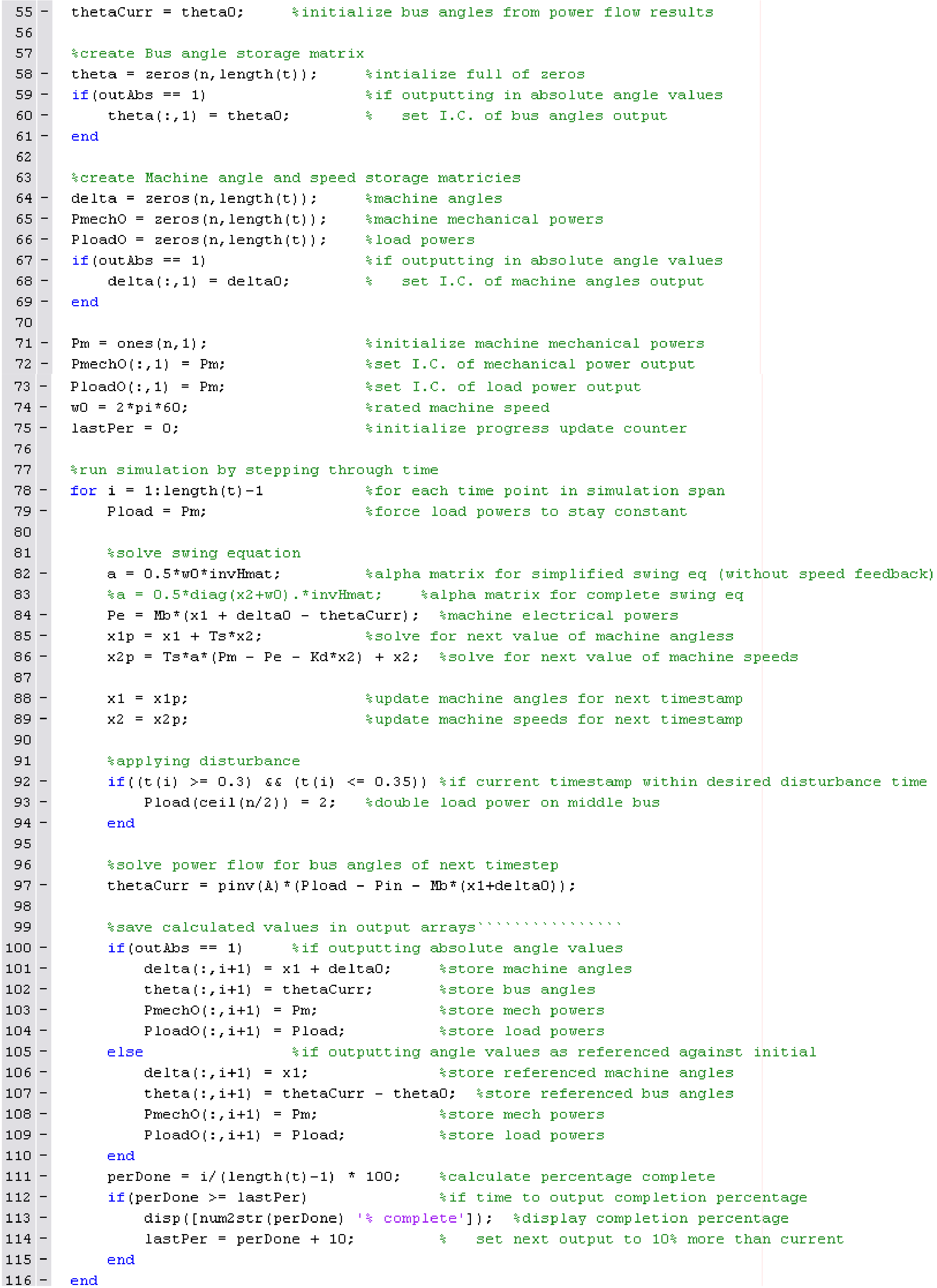
APPENDIX A

MATLAB CODE IMPLEMENTING SIMPLIFIED POWER SYSTEM DYNAMIC MODEL

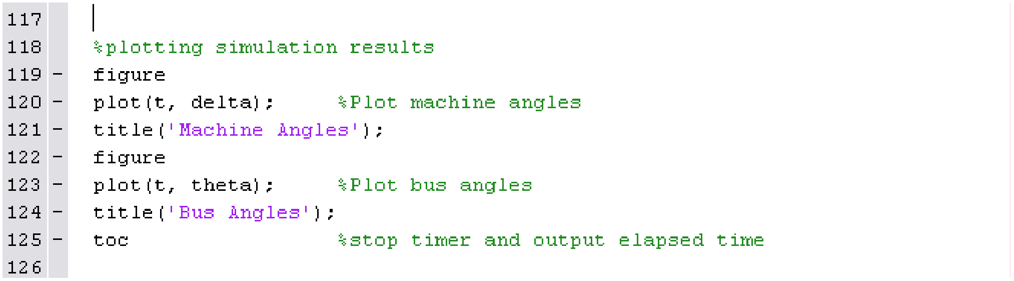
I MAIN SIMULATOR PROCESS – dynSim.m



```matlab
function [ t, theta, delta, Pmech0, Pload0 ] = dynSim( outAbs )
%Simulation of a uniform system using the simplified power system dynamic
%model.  Implements model based on swing equation and DC power flows with
%each bus in the system similar to the basic bus model (one generator, one
%load, variable number of lines).  The system constructed here is
%completely uniform with all generators, loads and lines having identical
%parameters.
%INPUTS:
%   outAbs => flag determining output format for angle values
%       if outAbs == 1 theta and delta in absolute angle values
%       if outAbs == 0 theta and delta in deviation from their initial values
%OUTPUTS:   (M time points in simulation, N Busses in system)
%   t      => 1xM Simulation time vector
%   theta  => NxM Array of bus angles
%   delta  => NxM Array of machine angles
%   Pmech0 => NxM Array of machine mechanical input powers
%   Pload0 => NxM Array of bus load powers
%REQUIRED M-FILES:
%   setUniMats.m ; mkLnSusMat.m or mkSqSusMat.m
%   initPF.m

%initialize output variables
t = 0; theta = 0; delta = 0; Pmech0 = 0; Pload0 = 0;

tic             %start timer

%%SUPPLEMENTARY INPUTS -- Describing system architecture
n = 151;        %set number of busses
%base Case (system parameters for uniform system)
Hi  = 1;        %machine inertias
Kdi = 0.01;     %machine damping factors
XLi = 0.05;     %line impedances
XGi = 0.0005;   %machine impedances
Pm = 1;         %set initial machine and load powers
Pinj = 0;       %set injection power and output power (linear system only)
%%END SUPPLEMENTARY INPUTS

%setup system matricies for uniform system
[invHmat, Kd, G, A, Mb] = setUniMats(n, Hi, Kdi, XLi, XGi);

Pin = zeros(n, 1);  %initialize bus injection powers
Pin(1) = Pinj;
Pin(n) = -Pinj;

%solve initial power flow for bus and machine angles
[theta0, delta0] = initPF(n, G, Pm, Pinj);

%set up discrete time variable
Ts = 10^-4;         %discrete time step
t = 0:Ts:5;         %simulation time span

%initialize swing equation state variables
x1 = zeros(n,1);
x2 = zeros(n,1);
```

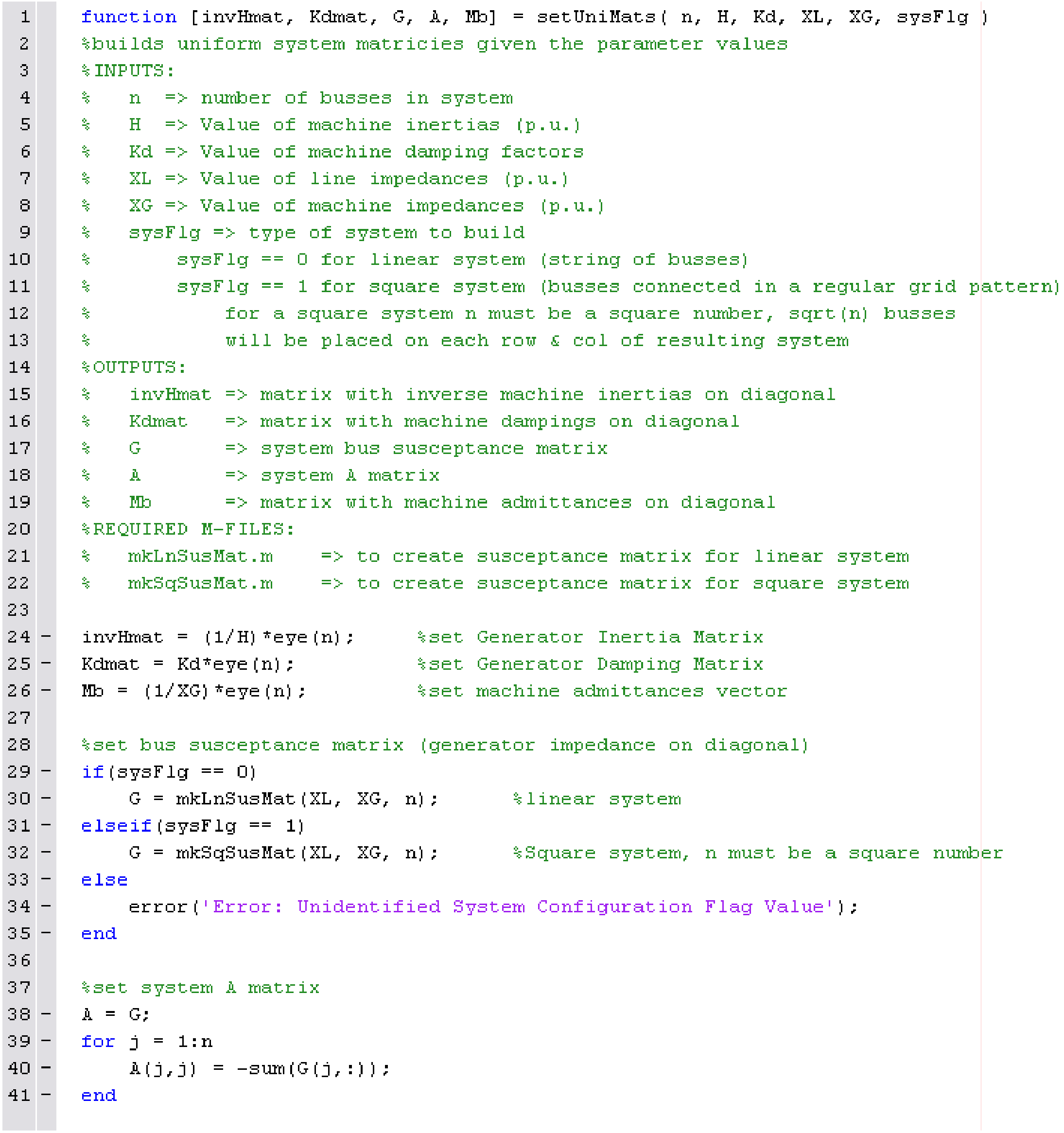


```
thetaCurr = theta0;     %initialize bus angles from power flow results

%create Bus angle storage matrix
theta = zeros(n,length(t));     %intialize full of zeros
if(outAbs == 1)                 %if outputting in absolute angle values
    theta(:,1) = theta0;        %   set I.C. of bus angles output
end

%create Machine angle and speed storage matricies
delta = zeros(n,length(t));     %machine angles
Pmech0 = zeros(n,length(t));    %machine mechanical powers
Pload0 = zeros(n,length(t));    %load powers
if(outAbs == 1)                 %if outputting in absolute angle values
    delta(:,1) = delta0;        %   set I.C. of machine angles output
end

Pm = ones(n,1);                 %initialize machine mechanical powers
Pmech0(:,1) = Pm;               %set I.C. of mechanical power output
Pload0(:,1) = Pm;               %set I.C. of load power output
w0 = 2*pi*60;                   %rated machine speed
lastPer = 0;                    %initialize progress update counter

%run simulation by stepping through time
for i = 1:length(t)-1           %for each time point in simulation span
    Pload = Pm;                 %force load powers to stay constant

    %solve swing equation
    a = 0.5*w0*invHmat;         %alpha matrix for simplified swing eq (without speed feedback)
    %a = 0.5*diag(x2+w0).*invHmat;    %alpha matrix for complete swing eq
    Pe = Mb*(x1 + delta0 - thetaCurr);  %machine electrical powers
    x1p = x1 + Ts*x2;           %solve for next value of machine angless
    x2p = Ts*a*(Pm - Pe - Kd*x2) + x2;  %solve for next value of machine speeds

    x1 = x1p;                   %update machine angles for next timestamp
    x2 = x2p;                   %update machine speeds for next timestamp

    %applying disturbance
    if((t(i) >= 0.3) && (t(i) <= 0.35)) %if current timestamp within desired disturbance time
        Pload(ceil(n/2)) = 2;   %double load power on middle bus
    end

    %solve power flow for bus angles of next timestep
    thetaCurr = pinv(A)*(Pload - Pin - Mb*(x1+delta0));

    %save calculated values in output arrays```````````````
    if(outAbs == 1)     %if outputting absolute angle values
        delta(:,i+1) = x1 + delta0;     %store machine angles
        theta(:,i+1) = thetaCurr;       %store bus angles
        Pmech0(:,i+1) = Pm;             %store mech powers
        Pload0(:,i+1) = Pload;          %store load powers
    else                %if outputting angle values as referenced against initial
        delta(:,i+1) = x1;              %store referenced machine angles
        theta(:,i+1) = thetaCurr - theta0;  %store referenced bus angles
        Pmech0(:,i+1) = Pm;             %store mech powers
        Pload0(:,i+1) = Pload;          %store load powers
    end
    perDone = i/(length(t)-1) * 100;    %calculate percentage complete
    if(perDone >= lastPer)              %if time to output completion percentage
        disp([num2str(perDone) '% complete']);  %display completion percentage
        lastPer = perDone + 10;         %   set next output to 10% more than current
    end
end
```



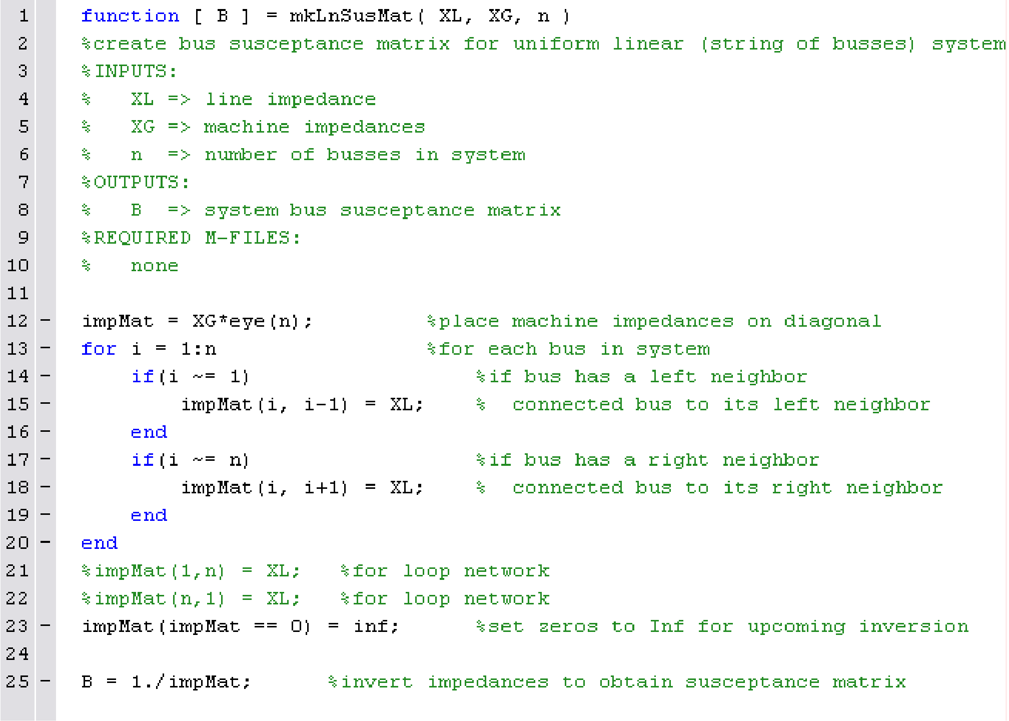
```
%plotting simulation results
figure
plot(t, delta);     %Plot machine angles
title('Machine Angles');
figure
plot(t, theta);     %Plot bus angles
title('Bus Angles');
toc                 %stop timer and output elapsed time
```

II SYSTEM MATRIX GENERATION SUB-PROCESS – setUniMats.m



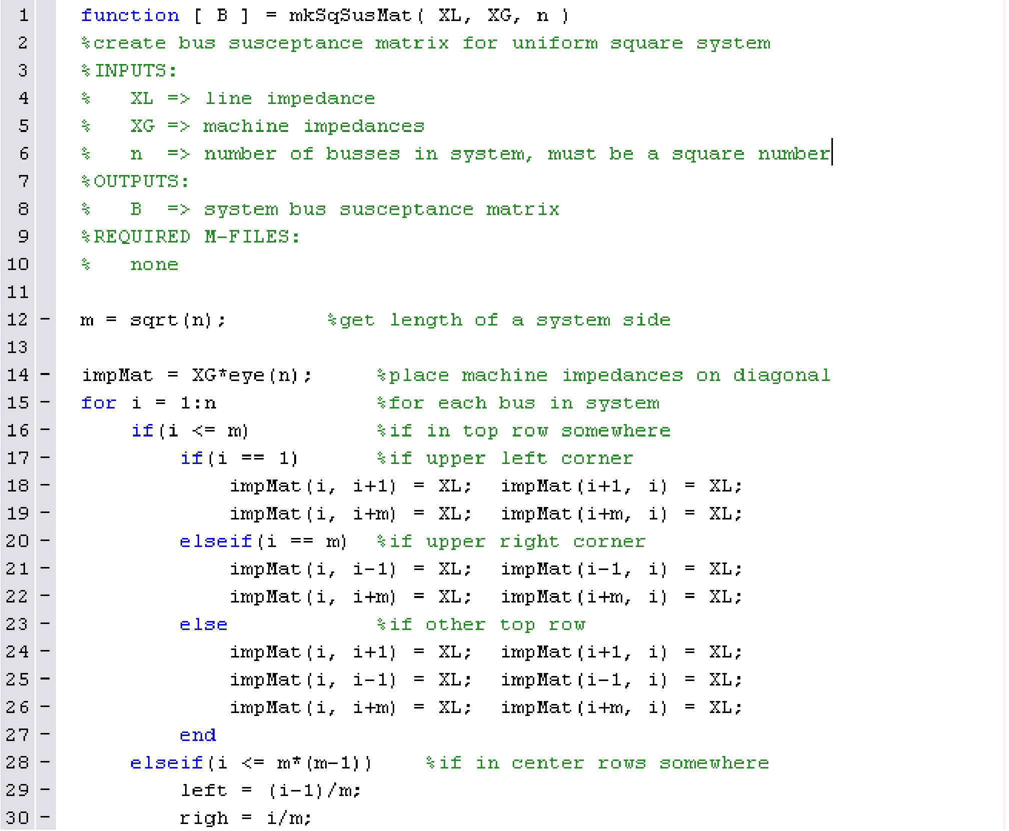
```
function [invHmat, Kdmat, G, A, Mb] = setUniMats( n, H, Kd, XL, XG, sysFlg )
%builds uniform system matricies given the parameter values
%INPUTS:
%   n  => number of busses in system
%   H  => Value of machine inertias (p.u.)
%   Kd => Value of machine damping factors
%   XL => Value of line impedances (p.u.)
%   XG => Value of machine impedances (p.u.)
%   sysFlg => type of system to build
%       sysFlg == 0 for linear system (string of busses)
%       sysFlg == 1 for square system (busses connected in a regular grid pattern)
%           for a square system n must be a square number, sqrt(n) busses
%           will be placed on each row & col of resulting system
%OUTPUTS:
%   invHmat => matrix with inverse machine inertias on diagonal
%   Kdmat   => matrix with machine dampings on diagonal
%   G       => system bus susceptance matrix
%   A       => system A matrix
%   Mb      => matrix with machine admittances on diagonal
%REQUIRED M-FILES:
%   mkLnSusMat.m     => to create susceptance matrix for linear system
%   mkSqSusMat.m     => to create susceptance matrix for square system

invHmat = (1/H)*eye(n);     %set Generator Inertia Matrix
Kdmat = Kd*eye(n);          %set Generator Damping Matrix
Mb = (1/XG)*eye(n);         %set machine admittances vector

%set bus susceptance matrix (generator impedance on diagonal)
if(sysFlg == 0)
    G = mkLnSusMat(XL, XG, n);      %linear system
elseif(sysFlg == 1)
    G = mkSqSusMat(XL, XG, n);      %Square system, n must be a square number
else
    error('Error: Unidentified System Configuration Flag Value');
end

%set system A matrix
A = G;
for j = 1:n
    A(j,j) = -sum(G(j,:));
end
```

III LINEAR SYSTEM TOPOLOGY SUB-PROCESS – mkLnSusMat.m

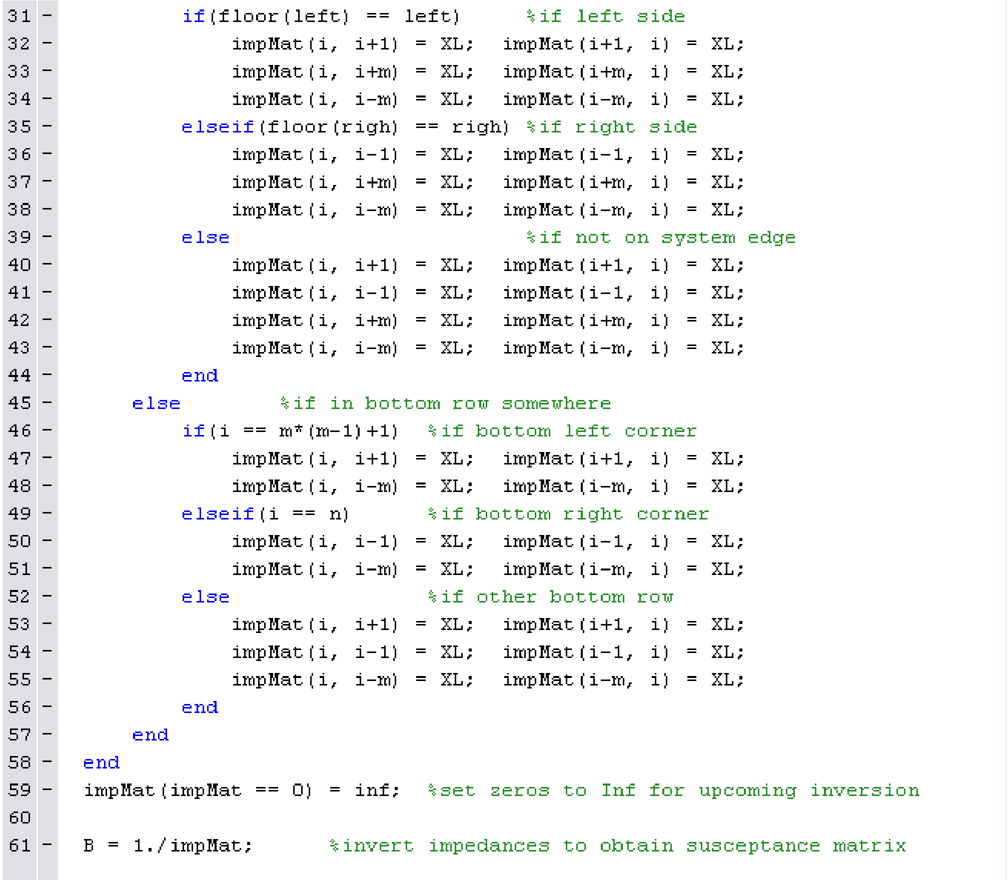


```
function [ B ] = mkLnSusMat( XL, XG, n )
%create bus susceptance matrix for uniform linear (string of busses) system
%INPUTS:
%   XL => line impedance
%   XG => machine impedances
%   n  => number of busses in system
%OUTPUTS:
%   B  => system bus susceptance matrix
%REQUIRED M-FILES:
%   none

impMat = XG*eye(n);         %place machine impedances on diagonal
for i = 1:n                 %for each bus in system
    if(i ~= 1)                  %if bus has a left neighbor
        impMat(i, i-1) = XL;    %  connected bus to its left neighbor
    end
    if(i ~= n)                  %if bus has a right neighbor
        impMat(i, i+1) = XL;    %  connected bus to its right neighbor
    end
end
%impMat(1,n) = XL;   %for loop network
%impMat(n,1) = XL;   %for loop network
impMat(impMat == 0) = inf;      %set zeros to Inf for upcoming inversion

B = 1./impMat;      %invert impedances to obtain susceptance matrix
```

IV SQUARE SYSTEM TOPOLOGY SUB-PROCESS – mkSqSusMat.m

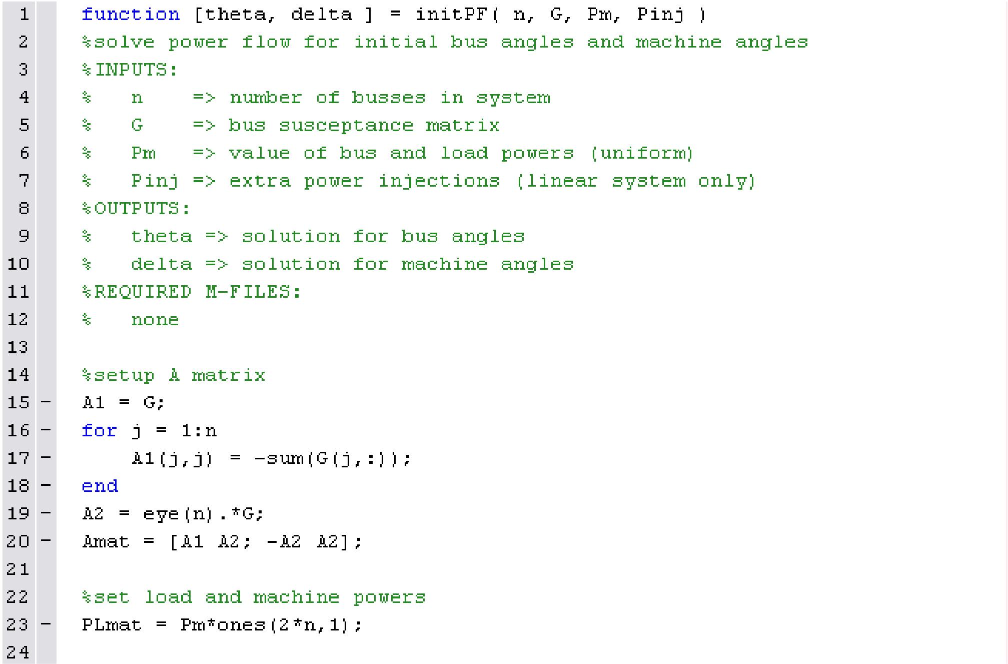


```
function [ B ] = mkSqSusMat( XL, XG, n )
%create bus susceptance matrix for uniform square system
%INPUTS:
%   XL => line impedance
%   XG => machine impedances
%   n  => number of busses in system, must be a square number
%OUTPUTS:
%   B  => system bus susceptance matrix
%REQUIRED M-FILES:
%   none

m = sqrt(n);        %get length of a system side

impMat = XG*eye(n);     %place machine impedances on diagonal
for i = 1:n             %for each bus in system
    if(i <= m)          %if in top row somewhere
        if(i == 1)      %if upper left corner
            impMat(i, i+1) = XL;  impMat(i+1, i) = XL;
            impMat(i, i+m) = XL;  impMat(i+m, i) = XL;
        elseif(i == m)  %if upper right corner
            impMat(i, i-1) = XL;  impMat(i-1, i) = XL;
            impMat(i, i+m) = XL;  impMat(i+m, i) = XL;
        else            %if other top row
            impMat(i, i+1) = XL;  impMat(i+1, i) = XL;
            impMat(i, i-1) = XL;  impMat(i-1, i) = XL;
            impMat(i, i+m) = XL;  impMat(i+m, i) = XL;
        end
    elseif(i <= m*(m-1))    %if in center rows somewhere
        left = (i-1)/m;
        righ = i/m;
```

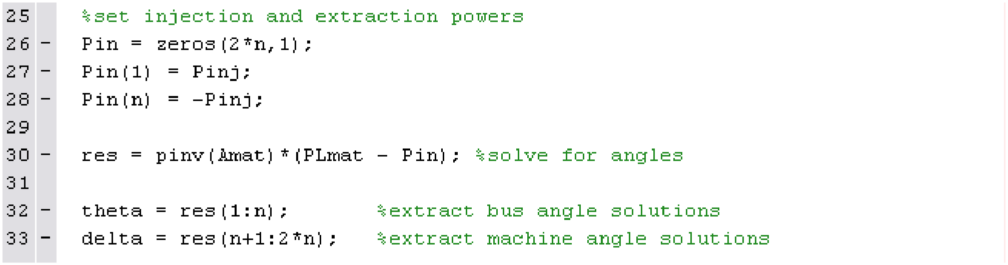


```
        if(floor(left) == left)     %if left side
            impMat(i, i+1) = XL;  impMat(i+1, i) = XL;
            impMat(i, i+m) = XL;  impMat(i+m, i) = XL;
            impMat(i, i-m) = XL;  impMat(i-m, i) = XL;
        elseif(floor(righ) == righ) %if right side
            impMat(i, i-1) = XL;  impMat(i-1, i) = XL;
            impMat(i, i+m) = XL;  impMat(i+m, i) = XL;
            impMat(i, i-m) = XL;  impMat(i-m, i) = XL;
        else                        %if not on system edge
            impMat(i, i+1) = XL;  impMat(i+1, i) = XL;
            impMat(i, i-1) = XL;  impMat(i-1, i) = XL;
            impMat(i, i+m) = XL;  impMat(i+m, i) = XL;
            impMat(i, i-m) = XL;  impMat(i-m, i) = XL;
        end
    else        %if in bottom row somewhere
        if(i == m*(m-1)+1)  %if bottom left corner
            impMat(i, i+1) = XL;  impMat(i+1, i) = XL;
            impMat(i, i-m) = XL;  impMat(i-m, i) = XL;
        elseif(i == n)      %if bottom right corner
            impMat(i, i-1) = XL;  impMat(i-1, i) = XL;
            impMat(i, i-m) = XL;  impMat(i-m, i) = XL;
        else                %if other bottom row
            impMat(i, i+1) = XL;  impMat(i+1, i) = XL;
            impMat(i, i-1) = XL;  impMat(i-1, i) = XL;
            impMat(i, i-m) = XL;  impMat(i-m, i) = XL;
        end
    end
end
impMat(impMat == 0) = inf;  %set zeros to Inf for upcoming inversion

B = 1./impMat;      %invert impedances to obtain susceptance matrix
```

V INITIAL POWER FLOW SOLVER SUB-PROCESS – initPF.m



```
function [theta, delta ] = initPF( n, G, Pm, Pinj )
%solve power flow for initial bus angles and machine angles
%INPUTS:
%   n    => number of busses in system
%   G    => bus susceptance matrix
%   Pm   => value of bus and load powers (uniform)
%   Pinj => extra power injections (linear system only)
%OUTPUTS:
%   theta => solution for bus angles
%   delta => solution for machine angles
%REQUIRED M-FILES:
%   none

%setup A matrix
A1 = G;
for j = 1:n
    A1(j,j) = -sum(G(j,:));
end
A2 = eye(n).*G;
Amat = [A1 A2; -A2 A2];

%set load and machine powers
PLmat = Pm*ones(2*n,1);

```

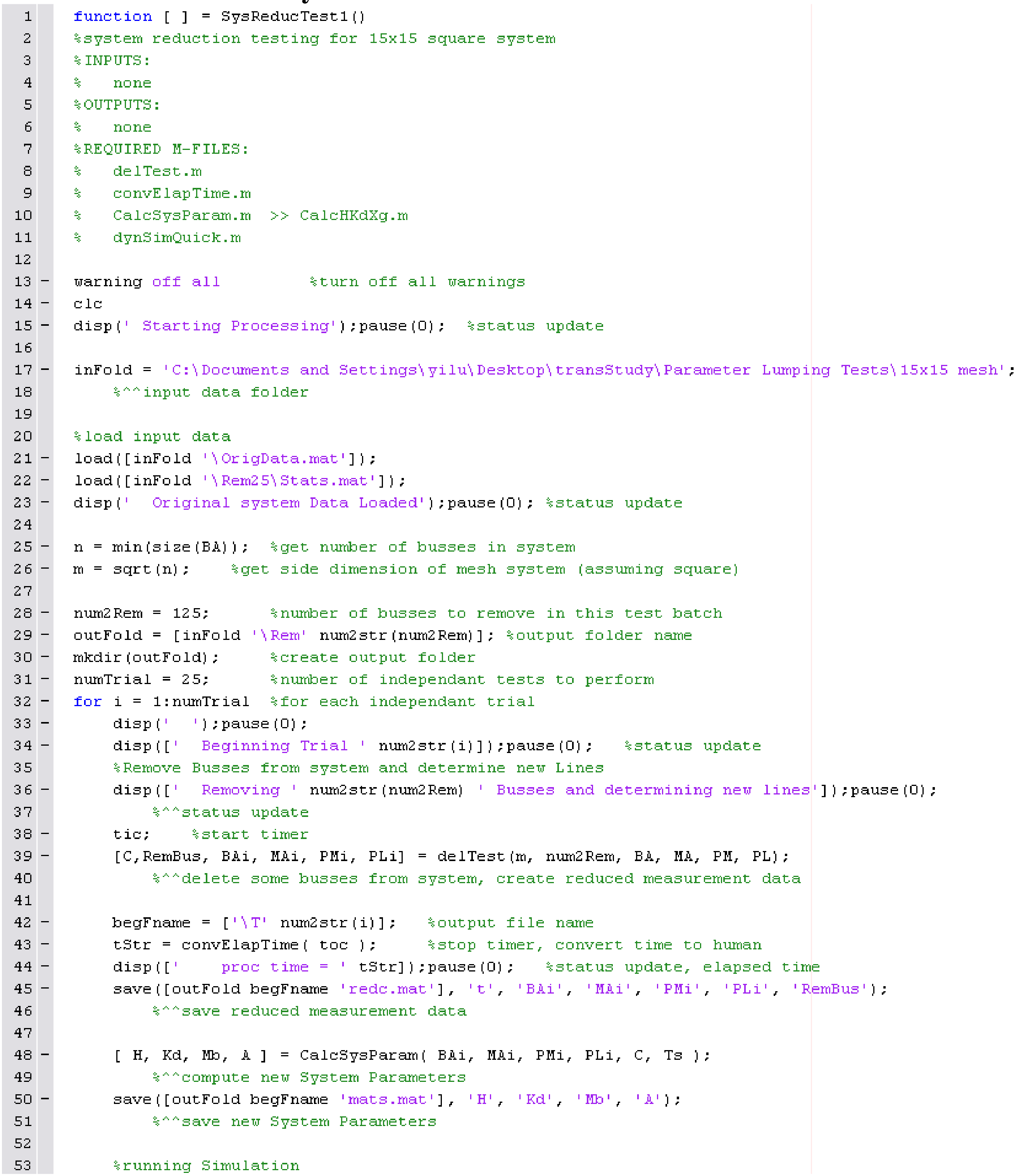


```
%set injection and extraction powers
Pin = zeros(2*n,1);
Pin(1) = Pinj;
Pin(n) = -Pinj;

res = pinv(Amat)*(PLmat - Pin); %solve for angles

theta = res(1:n);       %extract bus angle solutions
delta = res(n+1:2*n);   %extract machine angle solutions
```

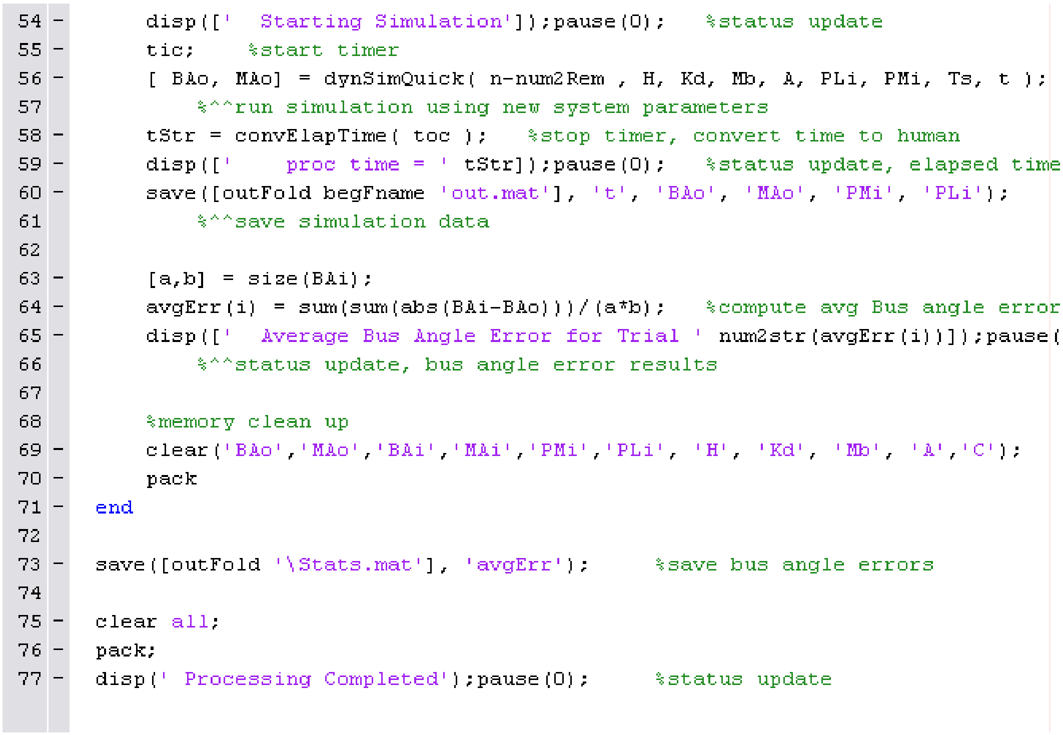
APPENDIX B

MATLAB CODE USED FOR SYSTEM REDUCTION TESTING

I MAIN PROCESS – SysReducTest1.m

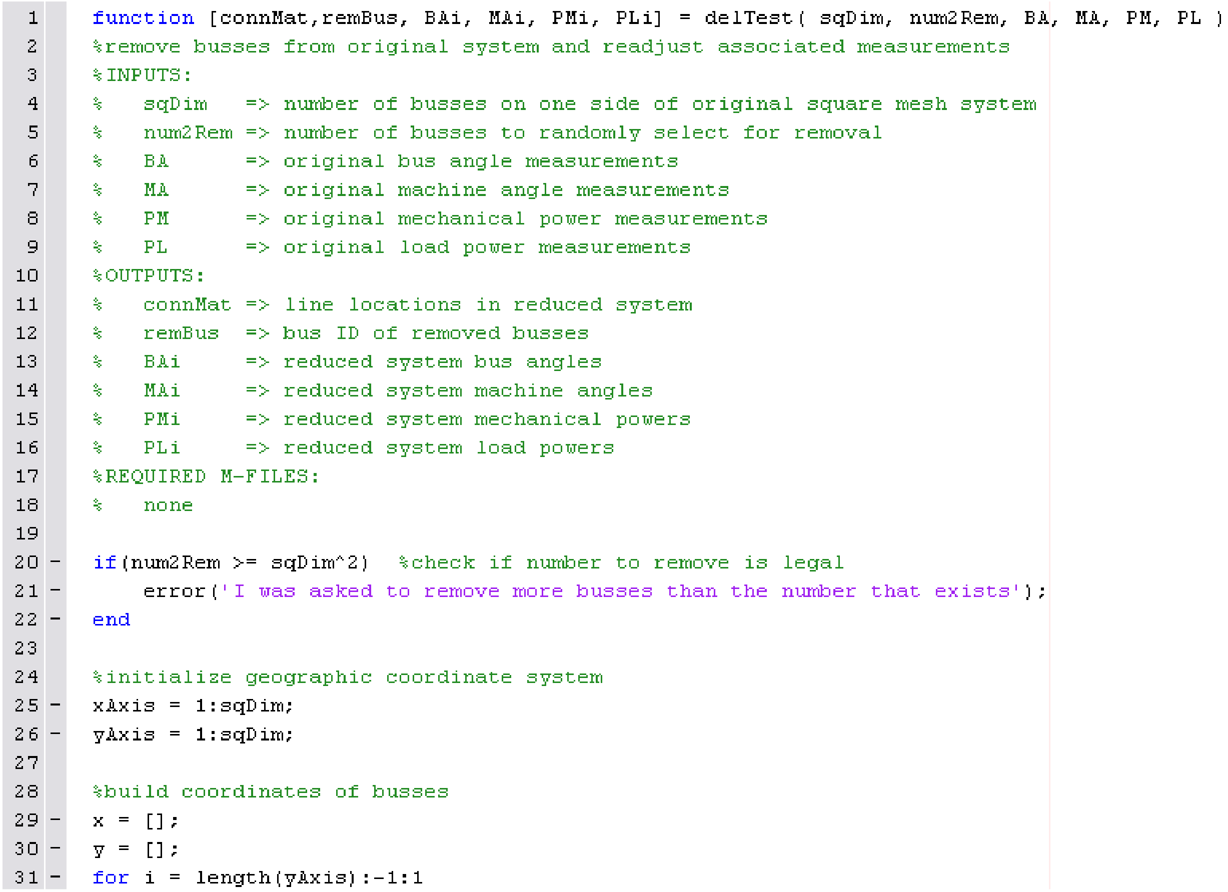


```
function [ ] = SysReducTest1()
%system reduction testing for 15x15 square system
%INPUTS:
%   none
%OUTPUTS:
%   none
%REQUIRED M-FILES:
%   delTest.m
%   convElapTime.m
%   CalcSysParam.m  >> CalcHKdXg.m
%   dynSimQuick.m

warning off all         %turn off all warnings
clc
disp(' Starting Processing');pause(0);  %status update

inFold = 'C:\Documents and Settings\yilu\Desktop\transStudy\Parameter Lumping Tests\15x15 mesh';
    %^^input data folder

%load input data
load([inFold '\OrigData.mat']);
load([inFold '\Rem25\Stats.mat']);
disp('  Original system Data Loaded');pause(0); %status update

n = min(size(BA));  %get number of busses in system
m = sqrt(n);    %get side dimension of mesh system (assuming square)

num2Rem = 125;      %number of busses to remove in this test batch
outFold = [inFold '\Rem' num2str(num2Rem)]; %output folder name
mkdir(outFold);     %create output folder
numTrial = 25;      %number of independant tests to perform
for i = 1:numTrial  %for each independant trial
    disp('  ');pause(0);
    disp(['  Beginning Trial ' num2str(i)]);pause(0);   %status update
    %Remove Busses from system and determine new Lines
    disp(['  Removing ' num2str(num2Rem) ' Busses and determining new lines']);pause(0);
        %^^status update
    tic;    %start timer
    [C,RemBus, BAi, MAi, PMi, PLi] = delTest(m, num2Rem, BA, MA, PM, PL);
        %^^delete some busses from system, create reduced measurement data

    begFname = ['\T' num2str(i)];   %output file name
    tStr = convElapTime( toc );     %stop timer, convert time to human
    disp(['    proc time = ' tStr]);pause(0);   %status update, elapsed time
    save([outFold begFname 'redc.mat'], 't', 'BAi', 'MAi', 'PMi', 'PLi', 'RemBus');
        %^^save reduced measurement data

    [ H, Kd, Mb, A ] = CalcSysParam( BAi, MAi, PMi, PLi, C, Ts );
        %^^compute new System Parameters
    save([outFold begFname 'mats.mat'], 'H', 'Kd', 'Mb', 'A');
        %^^save new System Parameters

    %running Simulation
```

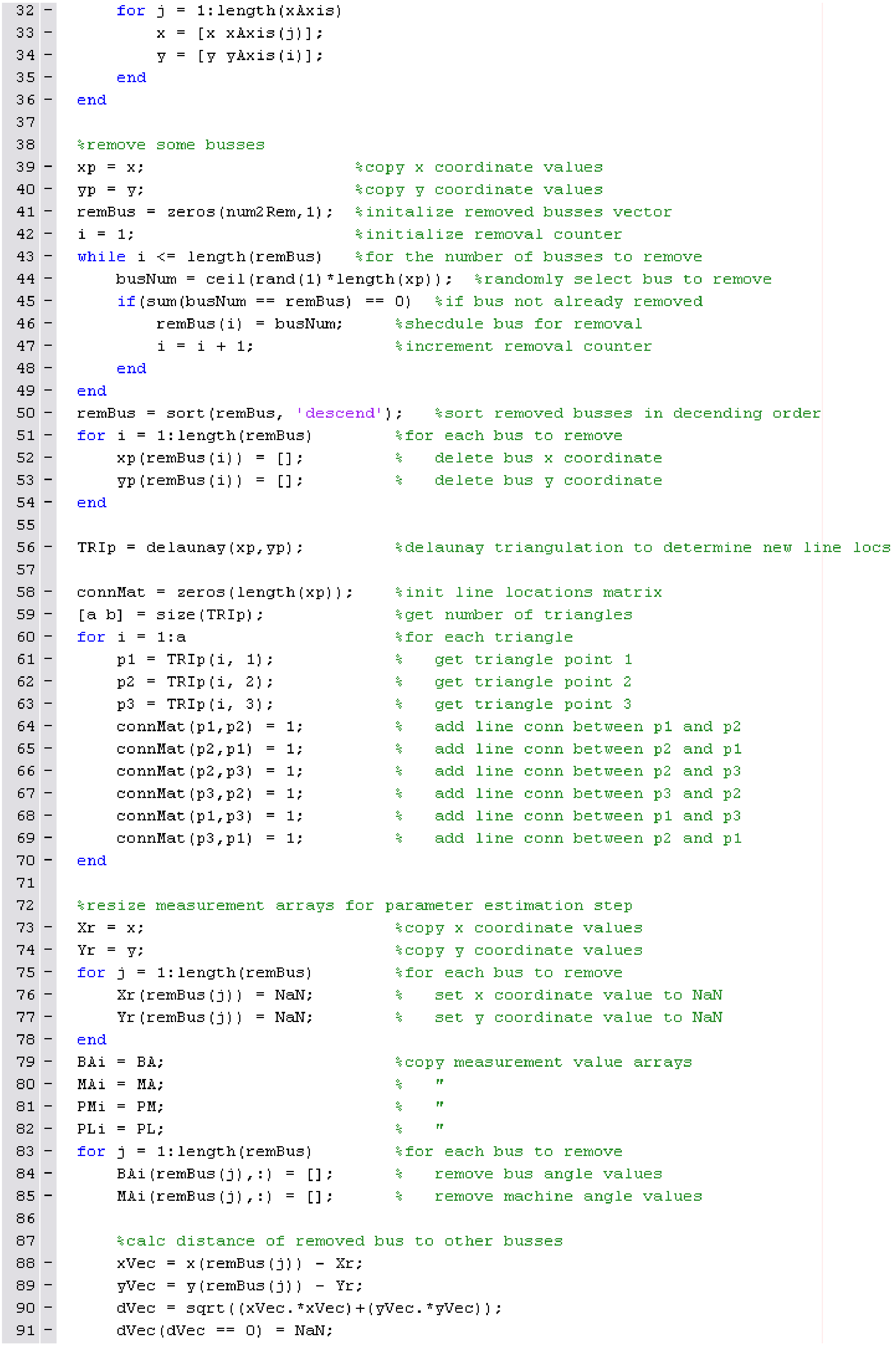


```
    disp(['  Starting Simulation']);pause(0);   %status update
    tic;    %start timer
    [ BAo, MAo] = dynSimQuick( n-num2Rem , H, Kd, Mb, A, PLi, PMi, Ts, t );
        %^^run simulation using new system parameters
    tStr = convElapTime( toc );   %stop timer, convert time to human
    disp(['    proc time = ' tStr]);pause(0);   %status update, elapsed time
    save([outFold begFname 'out.mat'], 't', 'BAo', 'MAo', 'PMi', 'PLi');
        %^^save simulation data

    [a,b] = size(BAi);
    avgErr(i) = sum(sum(abs(BAi-BAo)))/(a*b);   %compute avg Bus angle error
    disp(['  Average Bus Angle Error for Trial ' num2str(avgErr(i))]);pause(0);
        %^^status update, bus angle error results

    %memory clean up
    clear('BAo','MAo','BAi','MAi','PMi','PLi', 'H', 'Kd', 'Mb', 'A','C');
    pack
end

save([outFold '\Stats.mat'], 'avgErr');     %save bus angle errors

clear all;
pack;
disp(' Processing Completed');pause(0);     %status update
```

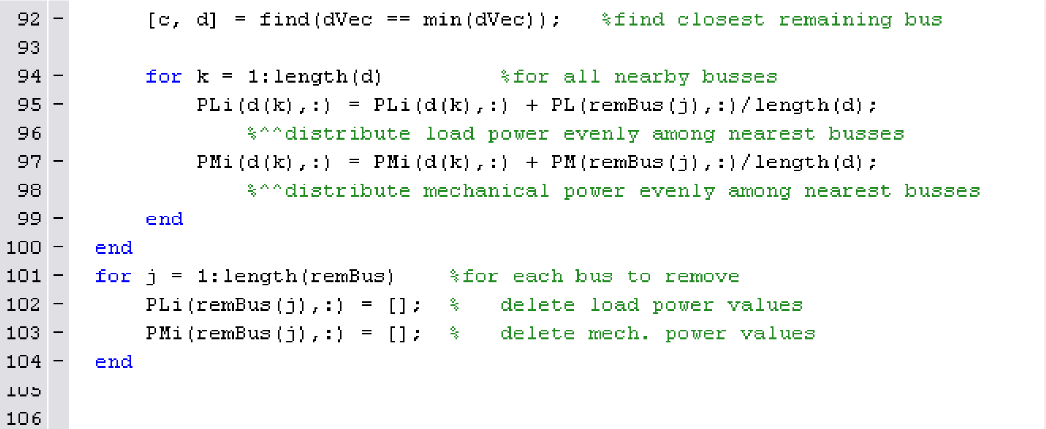
II BUSSES REMOVAL SUB-PROCESS – delTest.m



```
function [connMat,remBus, BAi, MAi, PMi, PLi] = delTest( sqDim, num2Rem, BA, MA, PM, PL )
%remove busses from original system and readjust associated measurements
%INPUTS:
%   sqDim   => number of busses on one side of original square mesh system
%   num2Rem => number of busses to randomly select for removal
%   BA      => original bus angle measurements
%   MA      => original machine angle measurements
%   PM      => original mechanical power measurements
%   PL      => original load power measurements
%OUTPUTS:
%   connMat => line locations in reduced system
%   remBus  => bus ID of removed busses
%   BAi     => reduced system bus angles
%   MAi     => reduced system machine angles
%   PMi     => reduced system mechanical powers
%   PLi     => reduced system load powers
%REQUIRED M-FILES:
%   none

if(num2Rem >= sqDim^2)  %check if number to remove is legal
    error('I was asked to remove more busses than the number that exists');
end

%initialize geographic coordinate system
xAxis = 1:sqDim;
yAxis = 1:sqDim;

%build coordinates of busses
x = [];
y = [];
for i = length(yAxis):-1:1
```

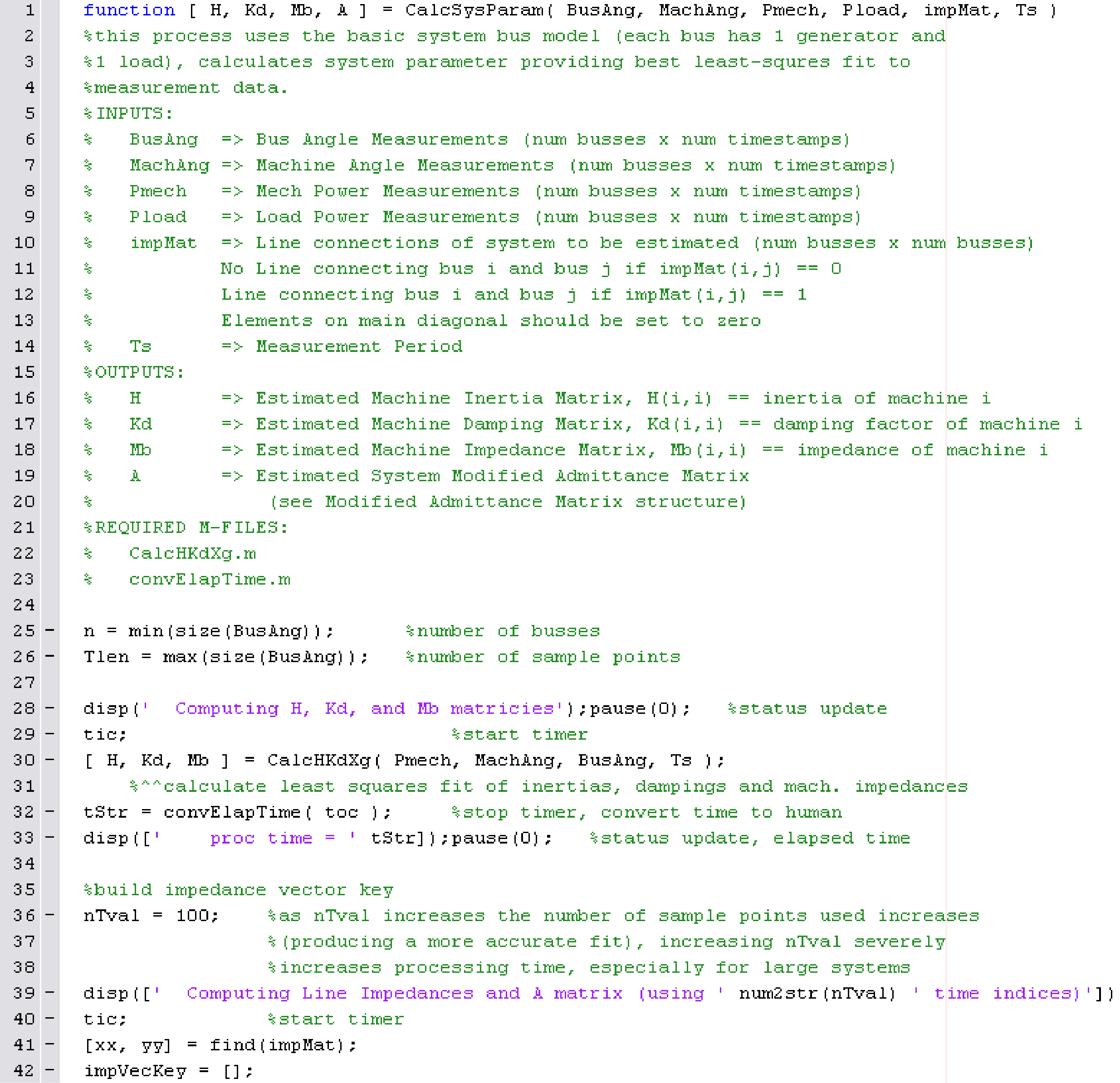


```
    for j = 1:length(xAxis)
        x = [x xAxis(j)];
        y = [y yAxis(i)];
    end
end

%remove some busses
xp = x;                     %copy x coordinate values
yp = y;                     %copy y coordinate values
remBus = zeros(num2Rem,1);  %initalize removed busses vector
i = 1;                      %initialize removal counter
while i <= length(remBus)   %for the number of busses to remove
    busNum = ceil(rand(1)*length(xp));  %randomly select bus to remove
    if(sum(busNum == remBus) == 0)  %if bus not already removed
        remBus(i) = busNum;     %shecdule bus for removal
        i = i + 1;              %increment removal counter
    end
end
remBus = sort(remBus, 'descend');   %sort removed busses in decending order
for i = 1:length(remBus)        %for each bus to remove
    xp(remBus(i)) = [];         %   delete bus x coordinate
    yp(remBus(i)) = [];         %   delete bus y coordinate
end

TRIp = delaunay(xp,yp);         %delaunay triangulation to determine new line locs

connMat = zeros(length(xp));    %init line locations matrix
[a b] = size(TRIp);             %get number of triangles
for i = 1:a                     %for each triangle
    p1 = TRIp(i, 1);            %   get triangle point 1
    p2 = TRIp(i, 2);            %   get triangle point 2
    p3 = TRIp(i, 3);            %   get triangle point 3
    connMat(p1,p2) = 1;         %   add line conn between p1 and p2
    connMat(p2,p1) = 1;         %   add line conn between p2 and p1
    connMat(p2,p3) = 1;         %   add line conn between p2 and p3
    connMat(p3,p2) = 1;         %   add line conn between p3 and p2
    connMat(p1,p3) = 1;         %   add line conn between p1 and p3
    connMat(p3,p1) = 1;         %   add line conn between p2 and p1
end

%resize measurement arrays for parameter estimation step
Xr = x;                         %copy x coordinate values
Yr = y;                         %copy y coordinate values
for j = 1:length(remBus)        %for each bus to remove
    Xr(remBus(j)) = NaN;        %   set x coordinate value to NaN
    Yr(remBus(j)) = NaN;        %   set y coordinate value to NaN
end
BAi = BA;                       %copy measurement value arrays
MAi = MA;                       %   "
PMi = PM;                       %   "
PLi = PL;                       %   "
for j = 1:length(remBus)        %for each bus to remove
    BAi(remBus(j),:) = [];      %   remove bus angle values
    MAi(remBus(j),:) = [];      %   remove machine angle values

    %calc distance of removed bus to other busses
    xVec = x(remBus(j)) - Xr;
    yVec = y(remBus(j)) - Yr;
    dVec = sqrt((xVec.*xVec)+(yVec.*yVec));
    dVec(dVec == 0) = NaN;
```

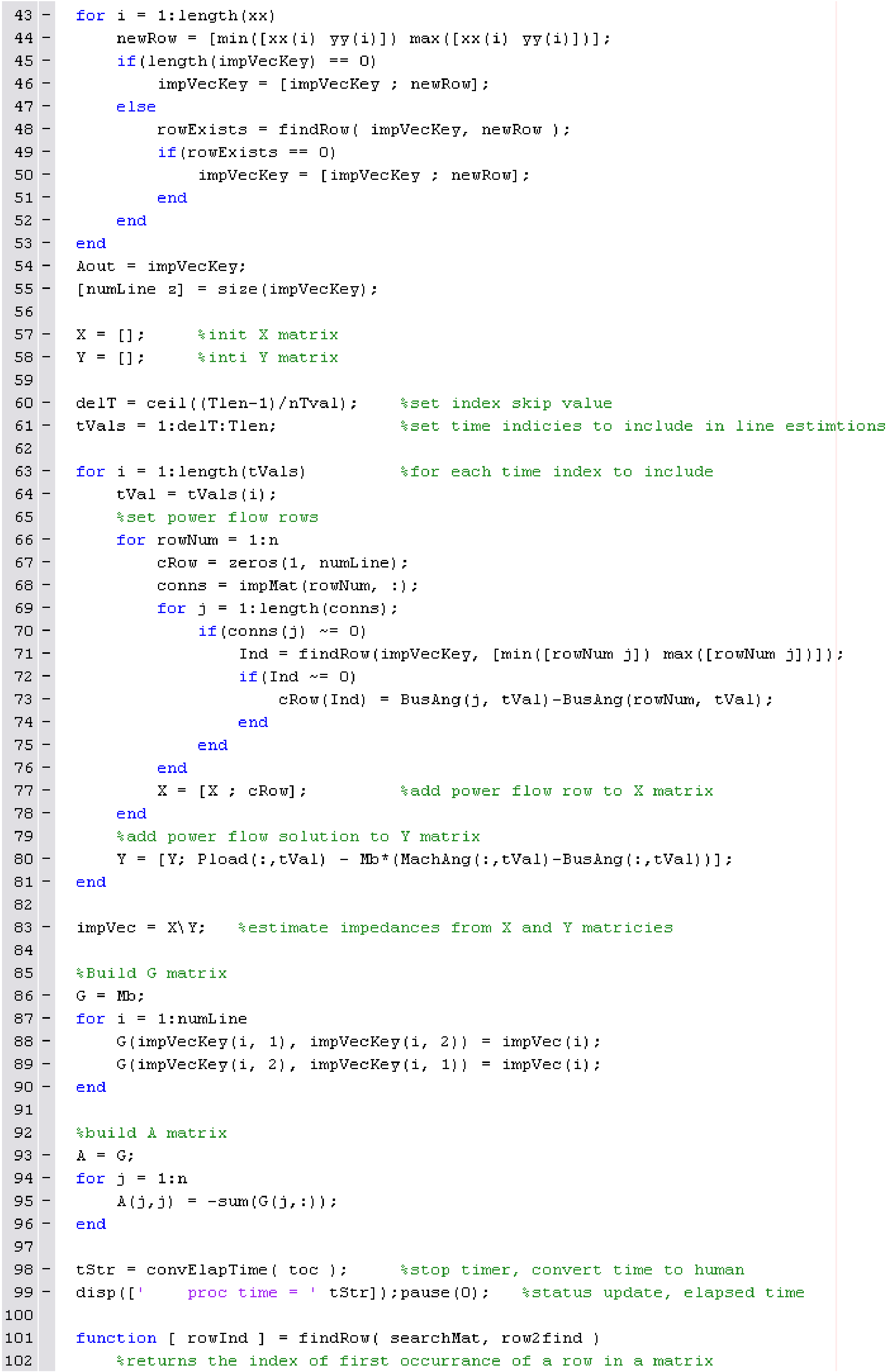


```
    [c, d] = find(dVec == min(dVec));   %find closest remaining bus

    for k = 1:length(d)          %for all nearby busses
        PLi(d(k),:) = PLi(d(k),:) + PL(remBus(j),:)/length(d);
            %^^distribute load power evenly among nearest busses
        PMi(d(k),:) = PMi(d(k),:) + PM(remBus(j),:)/length(d);
            %^^distribute mechanical power evenly among nearest busses
    end
end
for j = 1:length(remBus)    %for each bus to remove
    PLi(remBus(j),:) = [];  %   delete load power values
    PMi(remBus(j),:) = [];  %   delete mech. power values
end


```

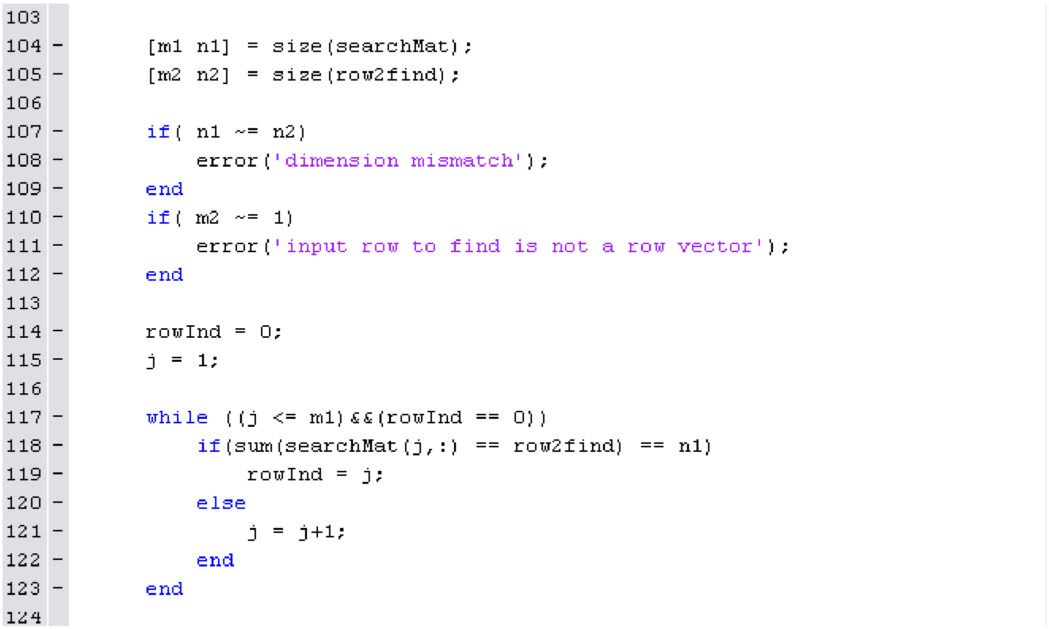
III System Estimation Sub-Process – CalcSysParam.m



```
function [ H, Kd, Mb, A ] = CalcSysParam( BusAng, MachAng, Pmech, Pload, impMat, Ts )
%this process uses the basic system bus model (each bus has 1 generator and
%1 load), calculates system parameter providing best least-squres fit to
%measurement data.
%INPUTS:
%   BusAng  => Bus Angle Measurements (num busses x num timestamps)
%   MachAng => Machine Angle Measurements (num busses x num timestamps)
%   Pmech   => Mech Power Measurements (num busses x num timestamps)
%   Pload   => Load Power Measurements (num busses x num timestamps)
%   impMat  => Line connections of system to be estimated (num busses x num busses)
%              No Line connecting bus i and bus j if impMat(i,j) == 0
%              Line connecting bus i and bus j if impMat(i,j) == 1
%              Elements on main diagonal should be set to zero
%   Ts      => Measurement Period
%OUTPUTS:
%   H       => Estimated Machine Inertia Matrix, H(i,i) == inertia of machine i
%   Kd      => Estimated Machine Damping Matrix, Kd(i,i) == damping factor of machine i
%   Mb      => Estimated Machine Impedance Matrix, Mb(i,i) == impedance of machine i
%   A       => Estimated System Modified Admittance Matrix
%                  (see Modified Admittance Matrix structure)
%REQUIRED M-FILES:
%   CalcHKdXg.m
%   convElapTime.m

n = min(size(BusAng));      %number of busses
Tlen = max(size(BusAng));   %number of sample points

disp('  Computing H, Kd, and Mb matricies');pause(0);   %status update
tic;                               %start timer
[ H, Kd, Mb ] = CalcHKdXg( Pmech, MachAng, BusAng, Ts );
    %^^calculate least squares fit of inertias, dampings and mach. impedances
tStr = convElapTime( toc );     %stop timer, convert time to human
disp(['    proc time = ' tStr]);pause(0);    %status update, elapsed time

%build impedance vector key
nTval = 100;    %as nTval increases the number of sample points used increases
                %(producing a more accurate fit), increasing nTval severely
                %increases processing time, especially for large systems
disp(['  Computing Line Impedances and A matrix (using ' num2str(nTval) ' time indices)'])
tic;            %start timer
[xx, yy] = find(impMat);
impVecKey = [];
```

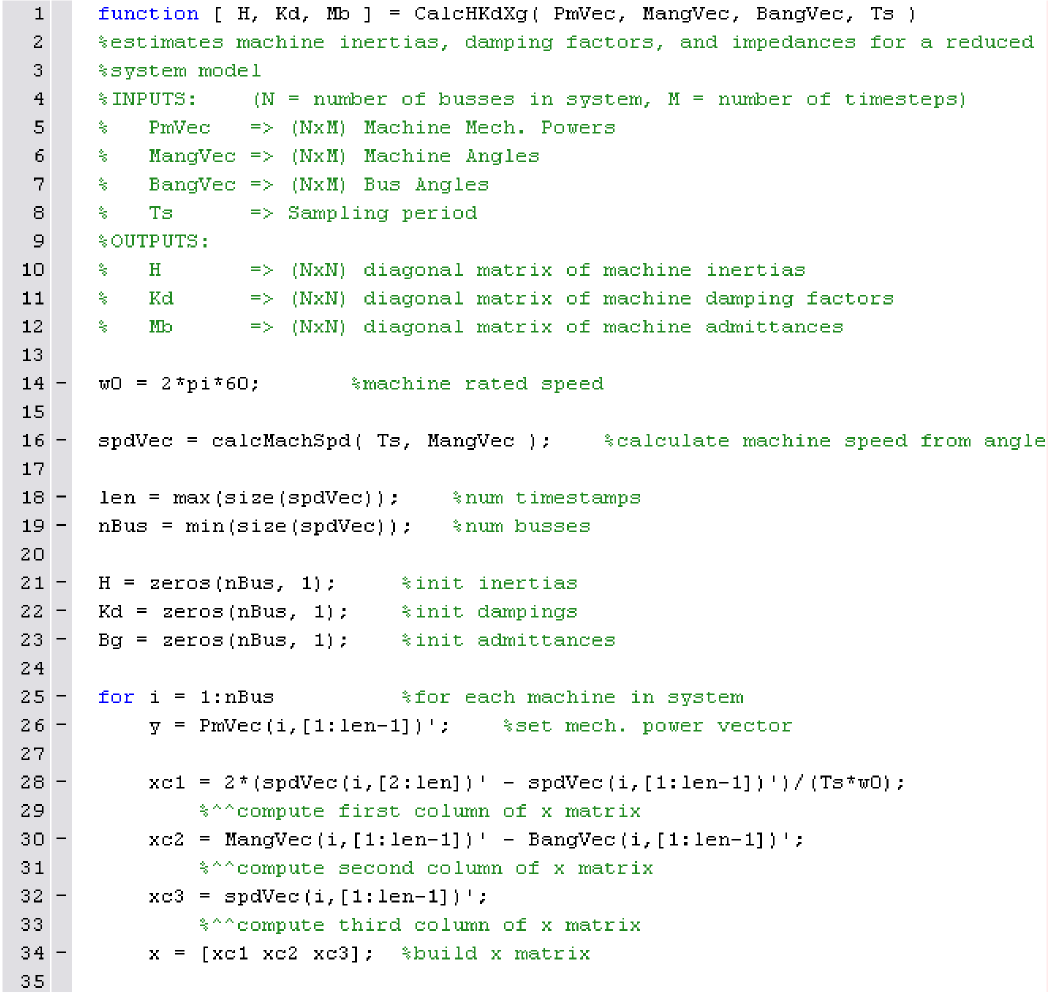


```
for i = 1:length(xx)
    newRow = [min([xx(i) yy(i)]) max([xx(i) yy(i)])];
    if(length(impVecKey) == 0)
        impVecKey = [impVecKey ; newRow];
    else
        rowExists = findRow( impVecKey, newRow );
        if(rowExists == 0)
            impVecKey = [impVecKey ; newRow];
        end
    end
end
Aout = impVecKey;
[numLine z] = size(impVecKey);

X = [];     %init X matrix
Y = [];     %inti Y matrix

delT = ceil((Tlen-1)/nTval);    %set index skip value
tVals = 1:delT:Tlen;            %set time indicies to include in line estimtions

for i = 1:length(tVals)         %for each time index to include
    tVal = tVals(i);
    %set power flow rows
    for rowNum = 1:n
        cRow = zeros(1, numLine);
        conns = impMat(rowNum, :);
        for j = 1:length(conns);
            if(conns(j) ~= 0)
                Ind = findRow(impVecKey, [min([rowNum j]) max([rowNum j])]);
                if(Ind ~= 0)
                    cRow(Ind) = BusAng(j, tVal)-BusAng(rowNum, tVal);
                end
            end
        end
        X = [X ; cRow];         %add power flow row to X matrix
    end
    %add power flow solution to Y matrix
    Y = [Y; Pload(:,tVal) - Mb*(MachAng(:,tVal)-BusAng(:,tVal))];
end

impVec = X\Y;   %estimate impedances from X and Y matricies

%Build G matrix
G = Mb;
for i = 1:numLine
    G(impVecKey(i, 1), impVecKey(i, 2)) = impVec(i);
    G(impVecKey(i, 2), impVecKey(i, 1)) = impVec(i);
end

%build A matrix
A = G;
for j = 1:n
    A(j,j) = -sum(G(j,:));
end

tStr = convElapTime( toc );     %stop timer, convert time to human
disp(['    proc time = ' tStr]);pause(0);   %status update, elapsed time

function [ rowInd ] = findRow( searchMat, row2find )
    %returns the index of first occurrance of a row in a matrix
```

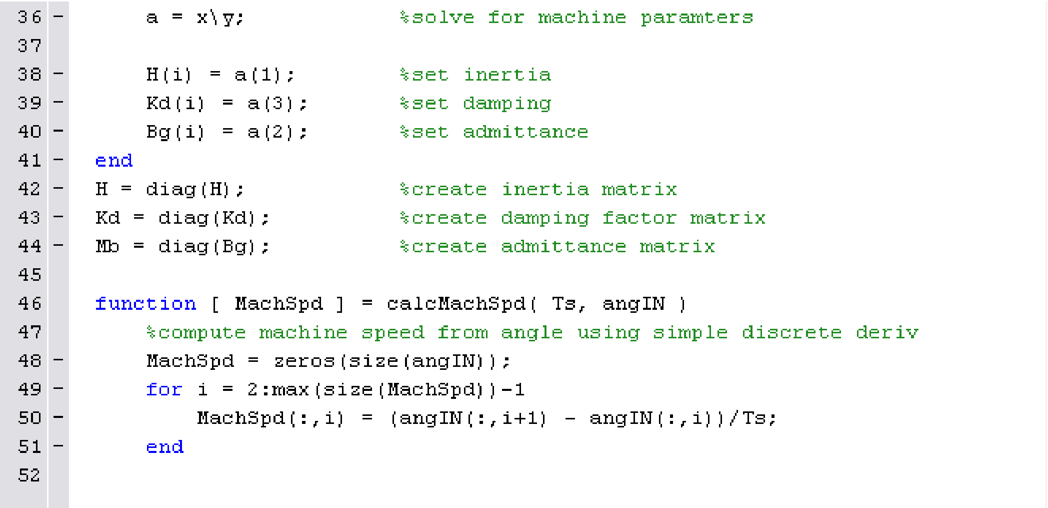


```
    [m1 n1] = size(searchMat);
    [m2 n2] = size(row2find);

    if( n1 ~= n2)
        error('dimension mismatch');
    end
    if( m2 ~= 1)
        error('input row to find is not a row vector');
    end

    rowInd = 0;
    j = 1;

    while ((j <= m1)&&(rowInd == 0))
        if(sum(searchMat(j,:) == row2find) == n1)
            rowInd = j;
        else
            j = j+1;
        end
    end

```

IV Machine Parameter Estimation Sub-Process – CalcHKdXg.m

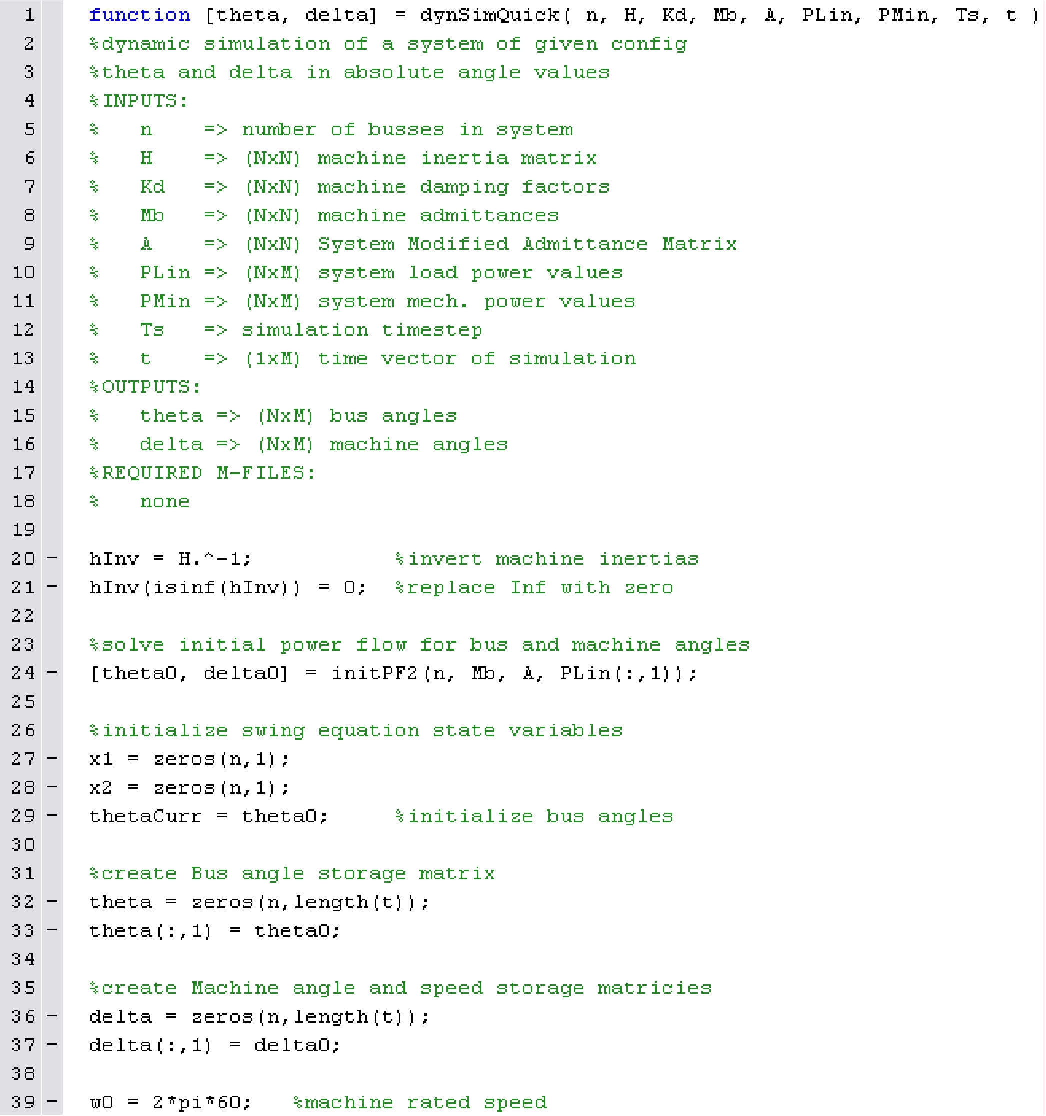


```
function [ H, Kd, Mb ] = CalcHKdXg( PmVec, MangVec, BangVec, Ts )
%estimates machine inertias, damping factors, and impedances for a reduced
%system model
%INPUTS:    (N = number of busses in system, M = number of timesteps)
%   PmVec   => (NxM) Machine Mech. Powers
%   MangVec => (NxM) Machine Angles
%   BangVec => (NxM) Bus Angles
%   Ts      => Sampling period
%OUTPUTS:
%   H       => (NxN) diagonal matrix of machine inertias
%   Kd      => (NxN) diagonal matrix of machine damping factors
%   Mb      => (NxN) diagonal matrix of machine admittances

w0 = 2*pi*60;       %machine rated speed

spdVec = calcMachSpd( Ts, MangVec );    %calculate machine speed from angle

len = max(size(spdVec));    %num timestamps
nBus = min(size(spdVec));   %num busses

H = zeros(nBus, 1);     %init inertias
Kd = zeros(nBus, 1);    %init dampings
Bg = zeros(nBus, 1);    %init admittances

for i = 1:nBus          %for each machine in system
    y = PmVec(i,[1:len-1])';    %set mech. power vector

    xc1 = 2*(spdVec(i,[2:len])' - spdVec(i,[1:len-1])')/(Ts*w0);
        %^^compute first column of x matrix
    xc2 = MangVec(i,[1:len-1])' - BangVec(i,[1:len-1])';
        %^^compute second column of x matrix
    xc3 = spdVec(i,[1:len-1])';
        %^^compute third column of x matrix
    x = [xc1 xc2 xc3];  %build x matrix

```

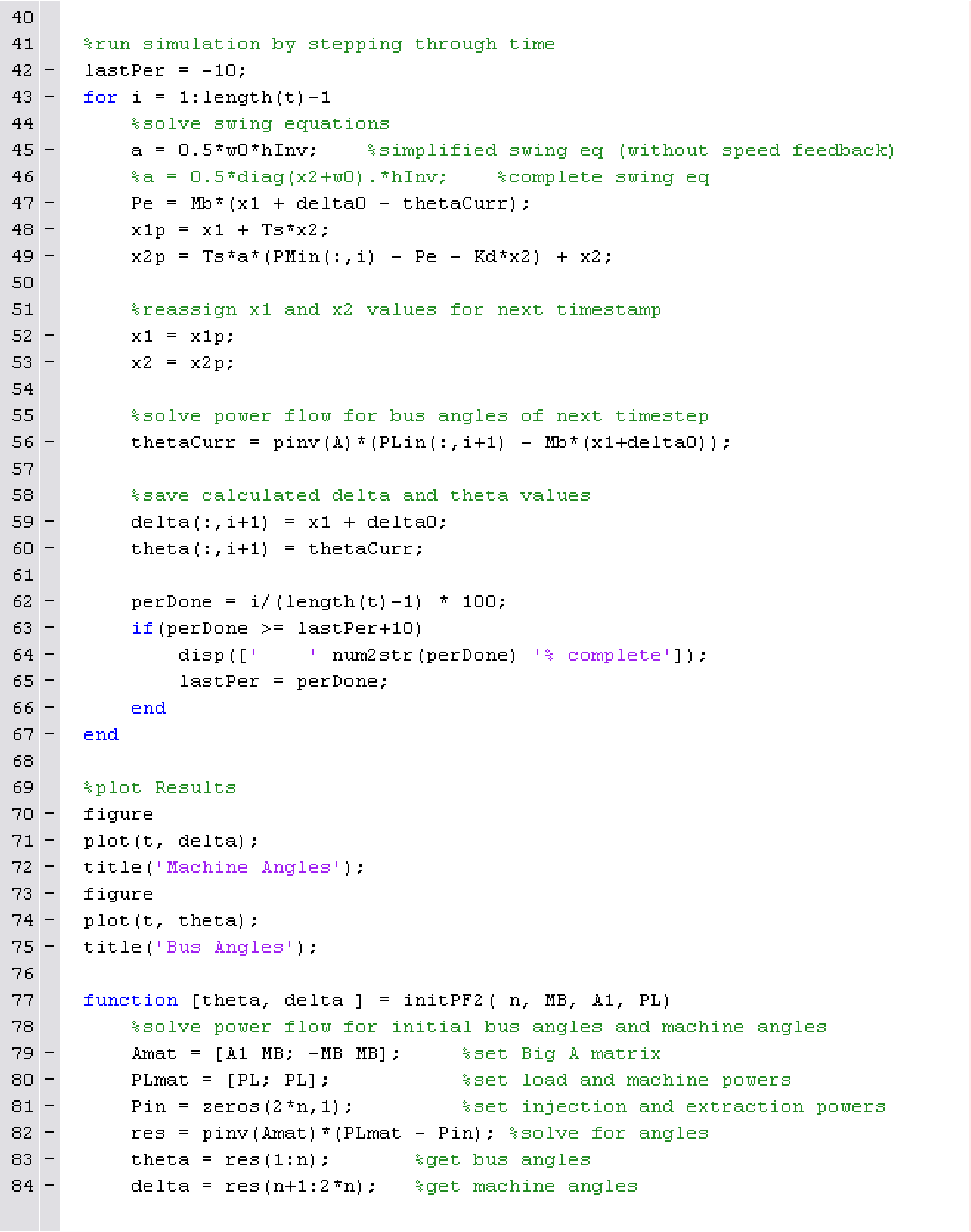


```matlab
    a = x\y;            %solve for machine paramters

    H(i) = a(1);        %set inertia
    Kd(i) = a(3);       %set damping
    Bg(i) = a(2);       %set admittance
end
H = diag(H);            %create inertia matrix
Kd = diag(Kd);          %create damping factor matrix
Mb = diag(Bg);          %create admittance matrix

function [ MachSpd ] = calcMachSpd( Ts, angIN )
    %compute machine speed from angle using simple discrete deriv
    MachSpd = zeros(size(angIN));
    for i = 2:max(size(MachSpd))-1
        MachSpd(:,i) = (angIN(:,i+1) - angIN(:,i))/Ts;
    end

```

V REDUCED SYSTEM DYNAMIC SIMULATOR – dynSimQuick.m



```matlab
function [theta, delta] = dynSimQuick( n, H, Kd, Mb, A, PLin, PMin, Ts, t )
%dynamic simulation of a system of given config
%theta and delta in absolute angle values
%INPUTS:
%   n    => number of busses in system
%   H    => (NxN) machine inertia matrix
%   Kd   => (NxN) machine damping factors
%   Mb   => (NxN) machine admittances
%   A    => (NxN) System Modified Admittance Matrix
%   PLin => (NxM) system load power values
%   PMin => (NxM) system mech. power values
%   Ts   => simulation timestep
%   t    => (1xM) time vector of simulation
%OUTPUTS:
%   theta => (NxM) bus angles
%   delta => (NxM) machine angles
%REQUIRED M-FILES:
%   none

hInv = H.^-1;           %invert machine inertias
hInv(isinf(hInv)) = 0;  %replace Inf with zero

%solve initial power flow for bus and machine angles
[theta0, delta0] = initPF2(n, Mb, A, PLin(:,1));

%initialize swing equation state variables
x1 = zeros(n,1);
x2 = zeros(n,1);
thetaCurr = theta0;     %initialize bus angles

%create Bus angle storage matrix
theta = zeros(n,length(t));
theta(:,1) = theta0;

%create Machine angle and speed storage matricies
delta = zeros(n,length(t));
delta(:,1) = delta0;

w0 = 2*pi*60;   %machine rated speed
```

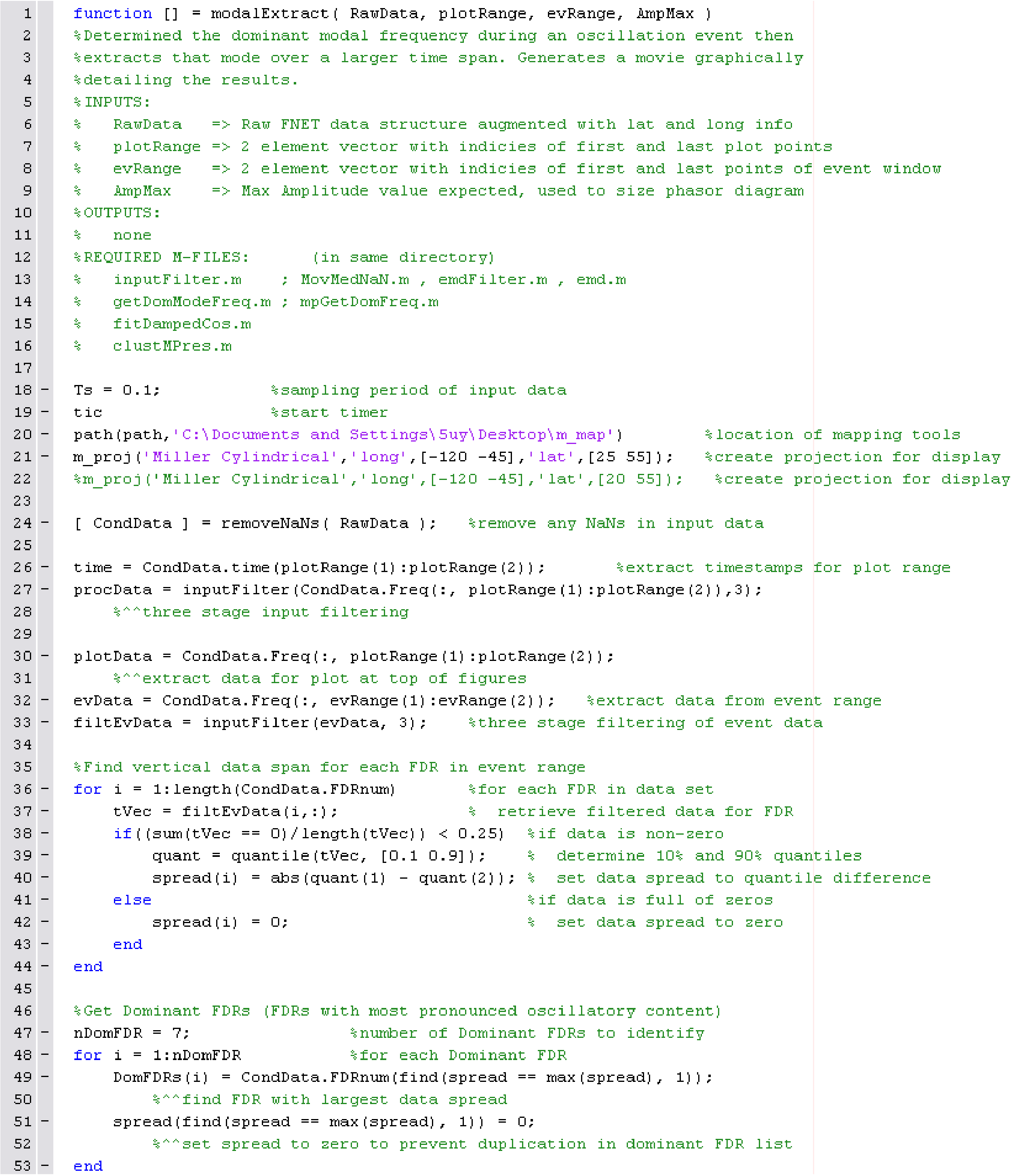


```

%run simulation by stepping through time
lastPer = -10;
for i = 1:length(t)-1
    %solve swing equations
    a = 0.5*w0*hInv;    %simplified swing eq (without speed feedback)
    %a = 0.5*diag(x2+w0).*hInv;    %complete swing eq
    Pe = Mb*(x1 + delta0 - thetaCurr);
    x1p = x1 + Ts*x2;
    x2p = Ts*a*(PMin(:,i) - Pe - Kd*x2) + x2;

    %reassign x1 and x2 values for next timestamp
    x1 = x1p;
    x2 = x2p;

    %solve power flow for bus angles of next timestep
    thetaCurr = pinv(A)*(PLin(:,i+1) - Mb*(x1+delta0));

    %save calculated delta and theta values
    delta(:,i+1) = x1 + delta0;
    theta(:,i+1) = thetaCurr;

    perDone = i/(length(t)-1) * 100;
    if(perDone >= lastPer+10)
        disp(['     ' num2str(perDone) '% complete']);
        lastPer = perDone;
    end
end

%plot Results
figure
plot(t, delta);
title('Machine Angles');
figure
plot(t, theta);
title('Bus Angles');

function [theta, delta ] = initPF2( n, MB, A1, PL)
    %solve power flow for initial bus angles and machine angles
    Amat = [A1 MB; -MB MB];     %set Big A matrix
    PLmat = [PL; PL];           %set load and machine powers
    Pin = zeros(2*n,1);         %set injection and extraction powers
    res = pinv(Amat)*(PLmat - Pin); %solve for angles
    theta = res(1:n);       %get bus angles
    delta = res(n+1:2*n);   %get machine angles
```

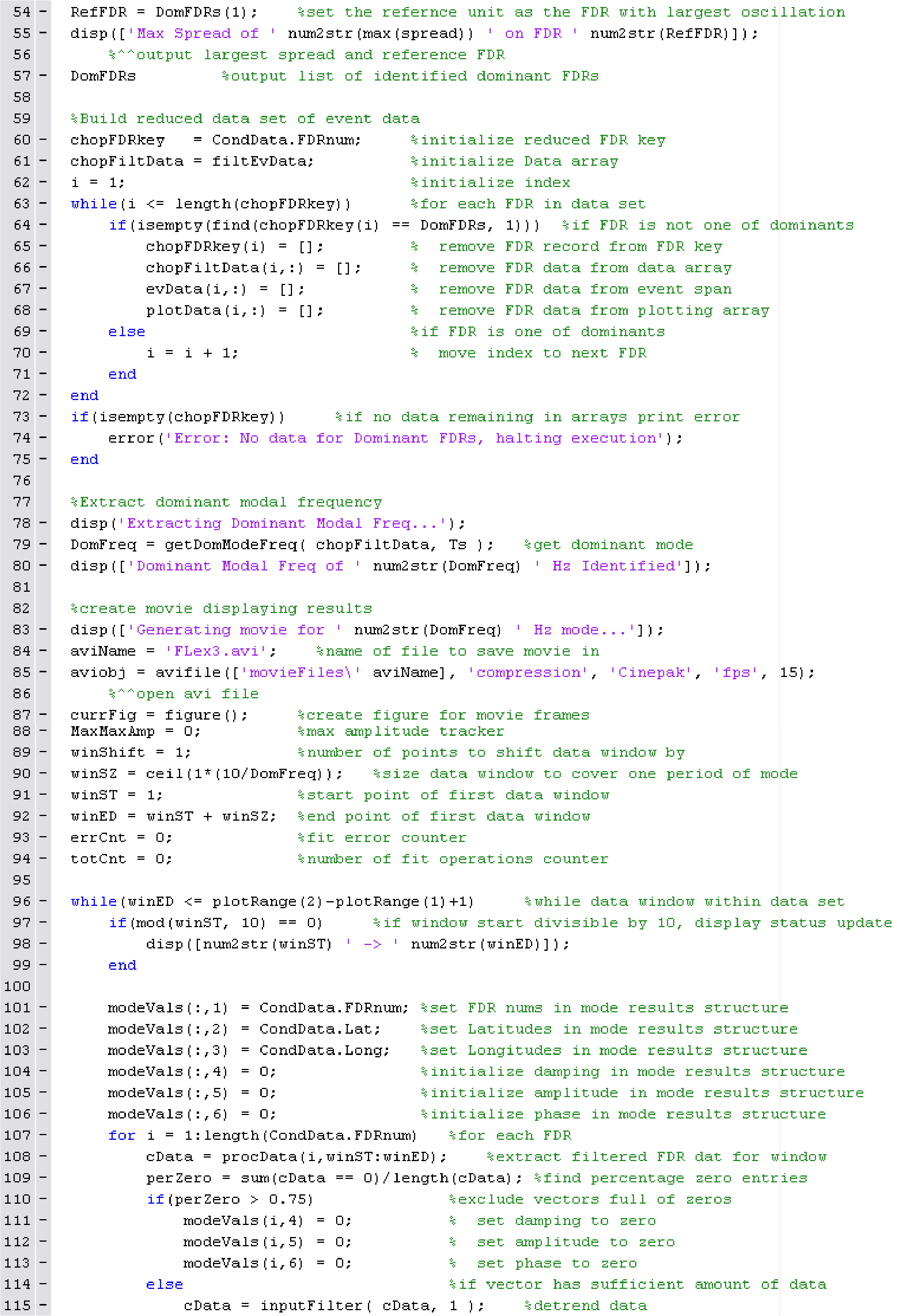
APPENDIX C

MATLAB CODE IMPLEMENTING MODAL EXTRACTION AND VISUALIZATION PACKAGE

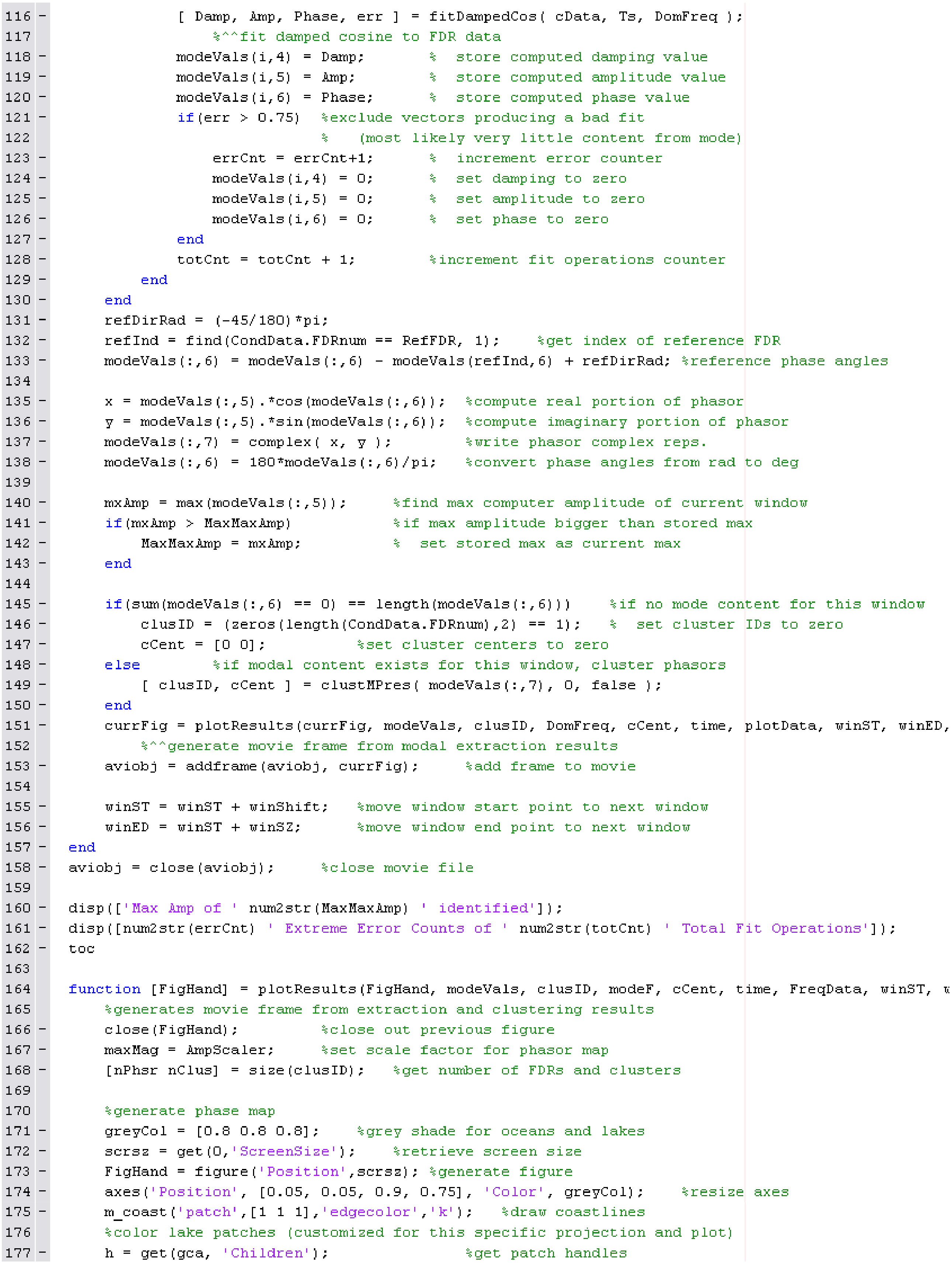
I MAIN PROCESS AND MOVIE GENERATION – modalExtract.m



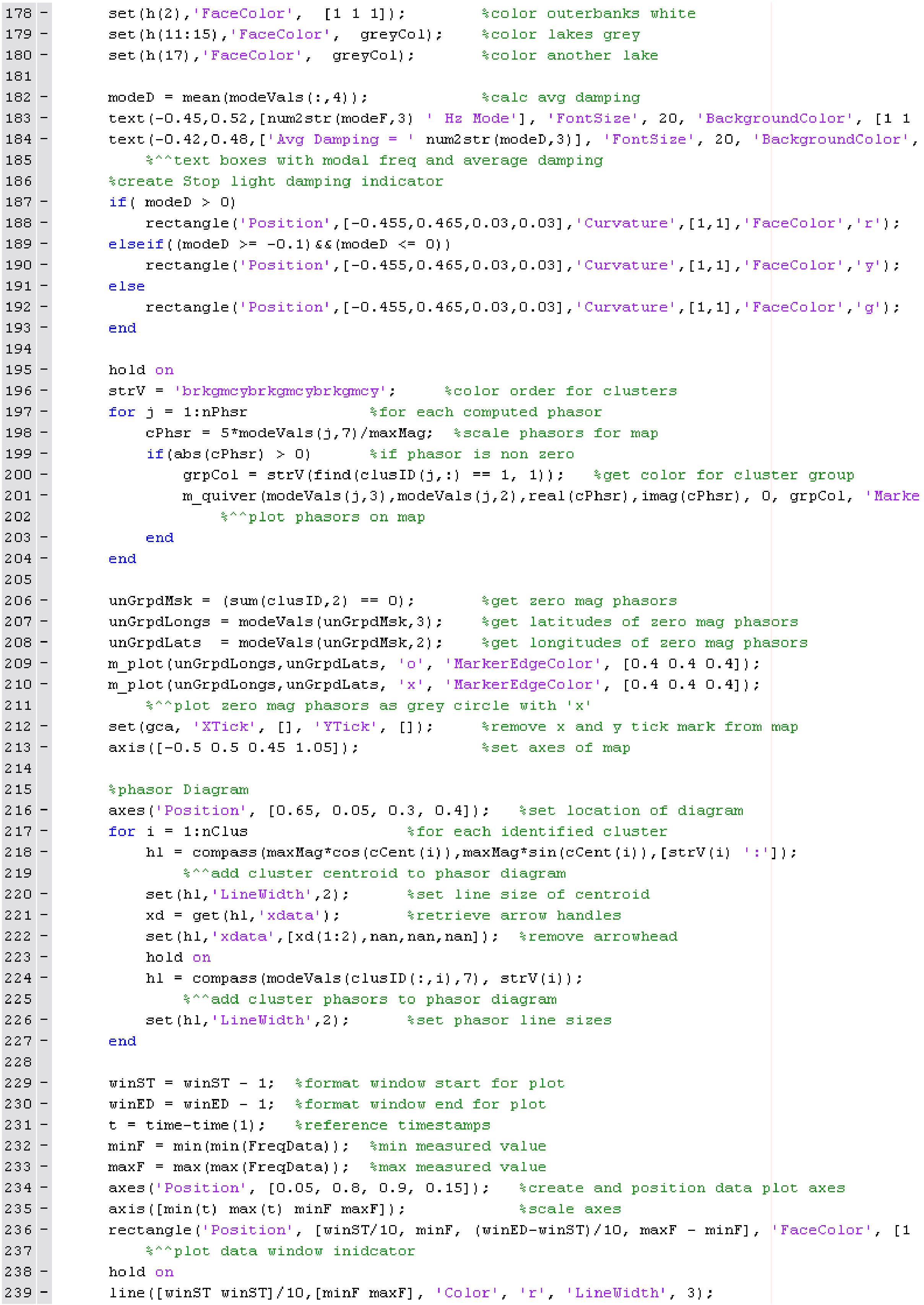
```matlab
function [] = modalExtract( RawData, plotRange, evRange, AmpMax )
%Determined the dominant modal frequency during an oscillation event then
%extracts that mode over a larger time span. Generates a movie graphically
%detailing the results.
%INPUTS:
%   RawData   => Raw FNET data structure augmented with lat and long info
%   plotRange => 2 element vector with indicies of first and last plot points
%   evRange   => 2 element vector with indicies of first and last points of event window
%   AmpMax    => Max Amplitude value expected, used to size phasor diagram
%OUTPUTS:
%   none
%REQUIRED M-FILES:      (in same directory)
%   inputFilter.m    ; MovMedNaN.m , emdFilter.m , emd.m
%   getDomModeFreq.m ; mpGetDomFreq.m
%   fitDampedCos.m
%   clustMPres.m

Ts = 0.1;           %sampling period of input data
tic                 %start timer
path(path,'C:\Documents and Settings\5uy\Desktop\m_map')        %location of mapping tools
m_proj('Miller Cylindrical','long',[-120 -45],'lat',[25 55]);   %create projection for display
%m_proj('Miller Cylindrical','long',[-120 -45],'lat',[20 55]);   %create projection for display

[ CondData ] = removeNaNs( RawData );   %remove any NaNs in input data

time = CondData.time(plotRange(1):plotRange(2));      %extract timestamps for plot range
procData = inputFilter(CondData.Freq(:, plotRange(1):plotRange(2)),3);
    %^^three stage input filtering

plotData = CondData.Freq(:, plotRange(1):plotRange(2));
    %^^extract data for plot at top of figures
evData = CondData.Freq(:, evRange(1):evRange(2));  %extract data from event range
filtEvData = inputFilter(evData, 3);    %three stage filtering of event data

%Find vertical data span for each FDR in event range
for i = 1:length(CondData.FDRnum)       %for each FDR in data set
    tVec = filtEvData(i,:);             %  retrieve filtered data for FDR
    if((sum(tVec == 0)/length(tVec)) < 0.25)  %if data is non-zero
        quant = quantile(tVec, [0.1 0.9]);    %  determine 10% and 90% quantiles
        spread(i) = abs(quant(1) - quant(2)); %  set data spread to quantile difference
    else                                      %if data is full of zeros
        spread(i) = 0;                        %  set data spread to zero
    end
end

%Get Dominant FDRs (FDRs with most pronounced oscillatory content)
nDomFDR = 7;                %number of Dominant FDRs to identify
for i = 1:nDomFDR           %for each Dominant FDR
    DomFDRs(i) = CondData.FDRnum(find(spread == max(spread), 1));
        %^^find FDR with largest data spread
    spread(find(spread == max(spread), 1)) = 0;
        %^^set spread to zero to prevent duplication in dominant FDR list
end
```



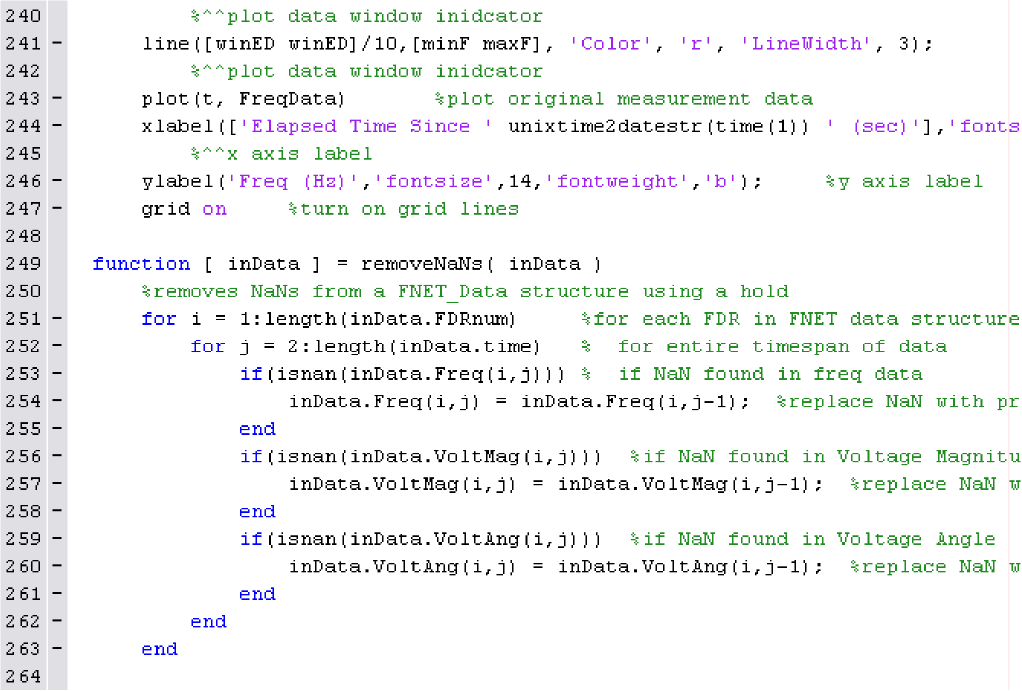
```
RefFDR = DomFDRs(1);    %set the refernce unit as the FDR with largest oscillation
disp(['Max Spread of ' num2str(max(spread)) ' on FDR ' num2str(RefFDR)]);
    %^^output largest spread and reference FDR
DomFDRs         %output list of identified dominant FDRs

%Build reduced data set of event data
chopFDRkey   = CondData.FDRnum;     %initialize reduced FDR key
chopFiltData = filtEvData;          %initialize Data array
i = 1;                              %initialize index
while(i <= length(chopFDRkey))      %for each FDR in data set
    if(isempty(find(chopFDRkey(i) == DomFDRs, 1)))  %if FDR is not one of dominants
        chopFDRkey(i) = [];         %  remove FDR record from FDR key
        chopFiltData(i,:) = [];     %  remove FDR data from data array
        evData(i,:) = [];           %  remove FDR data from event span
        plotData(i,:) = [];         %  remove FDR data from plotting array
    else                            %if FDR is one of dominants
        i = i + 1;                  %  move index to next FDR
    end
end
if(isempty(chopFDRkey))     %if no data remaining in arrays print error
    error('Error: No data for Dominant FDRs, halting execution');
end

%Extract dominant modal frequency
disp('Extracting Dominant Modal Freq...');
DomFreq = getDomModeFreq( chopFiltData, Ts );   %get dominant mode
disp(['Dominant Modal Freq of ' num2str(DomFreq) ' Hz Identified']);

%create movie displaying results
disp(['Generating movie for ' num2str(DomFreq) ' Hz mode...']);
aviName = 'FLex3.avi';    %name of file to save movie in
aviobj = avifile(['movieFiles\' aviName], 'compression', 'Cinepak', 'fps', 15);
    %^^open avi file
currFig = figure();     %create figure for movie frames
MaxMaxAmp = 0;          %max amplitude tracker
winShift = 1;           %number of points to shift data window by
winSZ = ceil(1*(10/DomFreq));   %size data window to cover one period of mode
winST = 1;              %start point of first data window
winED = winST + winSZ;  %end point of first data window
errCnt = 0;             %fit error counter
totCnt = 0;             %number of fit operations counter

while(winED <= plotRange(2)-plotRange(1)+1)     %while data window within data set
    if(mod(winST, 10) == 0)     %if window start divisible by 10, display status update
        disp([num2str(winST) ' -> ' num2str(winED)]);
    end

    modeVals(:,1) = CondData.FDRnum; %set FDR nums in mode results structure
    modeVals(:,2) = CondData.Lat;     %set Latitudes in mode results structure
    modeVals(:,3) = CondData.Long;   %set Longitudes in mode results structure
    modeVals(:,4) = 0;               %initialize damping in mode results structure
    modeVals(:,5) = 0;               %initialize amplitude in mode results structure
    modeVals(:,6) = 0;               %initialize phase in mode results structure
    for i = 1:length(CondData.FDRnum)   %for each FDR
        cData = procData(i,winST:winED);    %extract filtered FDR dat for window
        perZero = sum(cData == 0)/length(cData); %find percentage zero entries
        if(perZero > 0.75)              %exclude vectors full of zeros
            modeVals(i,4) = 0;          %  set damping to zero
            modeVals(i,5) = 0;          %  set amplitude to zero
            modeVals(i,6) = 0;          %  set phase to zero
        else                            %if vector has sufficient amount of data
            cData = inputFilter( cData, 1 );    %detrend data
```



```
            [ Damp, Amp, Phase, err ] = fitDampedCos( cData, Ts, DomFreq );
                %^^fit damped cosine to FDR data
            modeVals(i,4) = Damp;       %  store computed damping value
            modeVals(i,5) = Amp;        %  store computed amplitude value
            modeVals(i,6) = Phase;      %  store computed phase value
            if(err > 0.75)  %exclude vectors producing a bad fit
                            %    (most likely very little content from mode)
                errCnt = errCnt+1;      %  increment error counter
                modeVals(i,4) = 0;      %  set damping to zero
                modeVals(i,5) = 0;      %  set amplitude to zero
                modeVals(i,6) = 0;      %  set phase to zero
            end
            totCnt = totCnt + 1;        %increment fit operations counter
        end
    end
    refDirRad = (-45/180)*pi;
    refInd = find(CondData.FDRnum == RefFDR, 1);    %get index of reference FDR
    modeVals(:,6) = modeVals(:,6) - modeVals(refInd,6) + refDirRad; %reference phase angles

    x = modeVals(:,5).*cos(modeVals(:,6));  %compute real portion of phasor
    y = modeVals(:,5).*sin(modeVals(:,6));  %compute imaginary portion of phasor
    modeVals(:,7) = complex( x, y );        %write phasor complex reps.
    modeVals(:,6) = 180*modeVals(:,6)/pi;   %convert phase angles from rad to deg

    mxAmp = max(modeVals(:,5));     %find max computer amplitude of current window
    if(mxAmp > MaxMaxAmp)           %if max amplitude bigger than stored max
        MaxMaxAmp = mxAmp;          %  set stored max as current max
    end

    if(sum(modeVals(:,6) == 0) == length(modeVals(:,6)))    %if no mode content for this window
        clusID = (zeros(length(CondData.FDRnum),2) == 1);   %  set cluster IDs to zero
        cCent = [0 0];          %set cluster centers to zero
    else        %if modal content exists for this window, cluster phasors
        [ clusID, cCent ] = clustMPres( modeVals(:,7), 0, false );
    end
    currFig = plotResults(currFig, modeVals, clusID, DomFreq, cCent, time, plotData, winST, winED,
        %^^generate movie frame from modal extraction results
    aviobj = addframe(aviobj, currFig);     %add frame to movie

    winST = winST + winShift;   %move window start point to next window
    winED = winST + winSZ;      %move window end point to next window
end
aviobj = close(aviobj);     %close movie file

disp(['Max Amp of ' num2str(MaxMaxAmp) ' identified']);
disp([num2str(errCnt) ' Extreme Error Counts of ' num2str(totCnt) ' Total Fit Operations']);
toc

function [FigHand] = plotResults(FigHand, modeVals, clusID, modeF, cCent, time, FreqData, winST, w
    %generates movie frame from extraction and clustering results
    close(FigHand);         %close out previous figure
    maxMag = AmpScaler;     %set scale factor for phasor map
    [nPhsr nClus] = size(clusID);   %get number of FDRs and clusters

    %generate phase map
    greyCol = [0.8 0.8 0.8];    %grey shade for oceans and lakes
    scrsz = get(0,'ScreenSize');    %retrieve screen size
    FigHand = figure('Position',scrsz); %generate figure
    axes('Position', [0.05, 0.05, 0.9, 0.75], 'Color', greyCol);    %resize axes
    m_coast('patch',[1 1 1],'edgecolor','k');   %draw coastlines
    %color lake patches (customized for this specific projection and plot)
    h = get(gca, 'Children');               %get patch handles
```

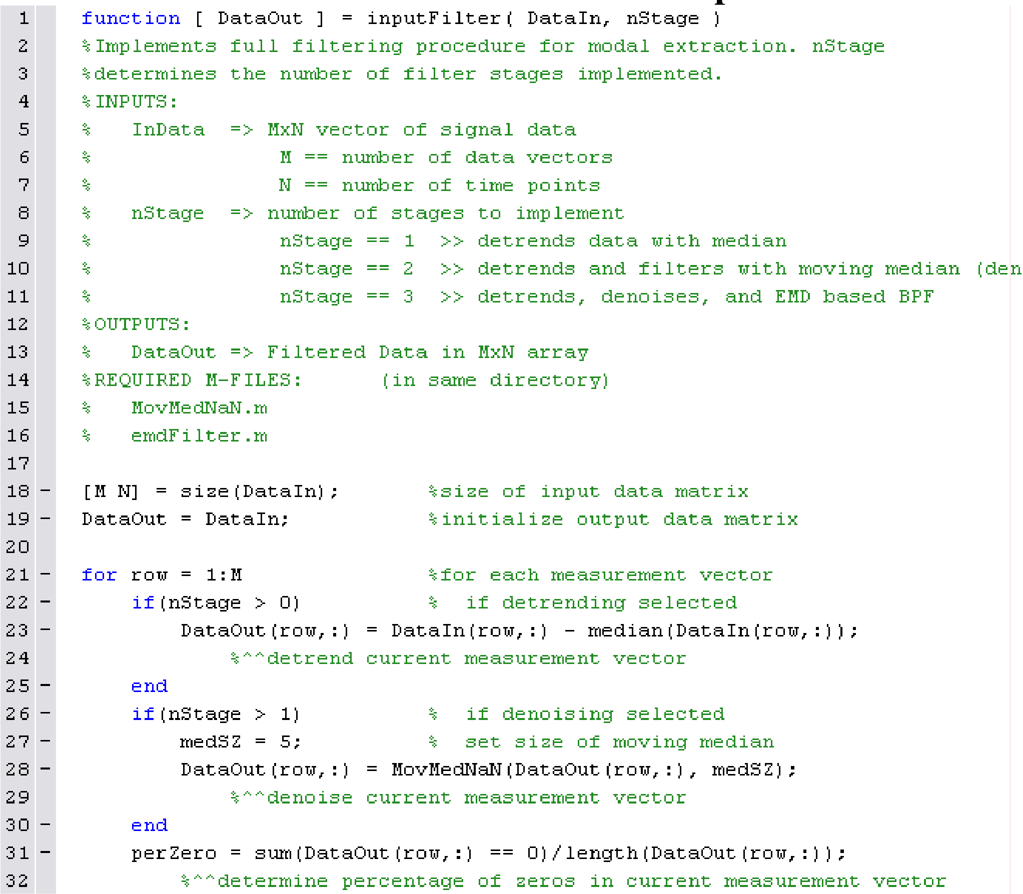


```
set(h(2),'FaceColor',  [1 1 1]);        %color outerbanks white
set(h(11:15),'FaceColor',  greyCol);    %color lakes grey
set(h(17),'FaceColor',  greyCol);       %color another lake

modeD = mean(modeVals(:,4));            %calc avg damping
text(-0.45,0.52,[num2str(modeF,3) ' Hz Mode'], 'FontSize', 20, 'BackgroundColor', [1 1
text(-0.42,0.48,['Avg Damping = ' num2str(modeD,3)], 'FontSize', 20, 'BackgroundColor',
    %^^text boxes with modal freq and average damping
%create Stop light damping indicator
if( modeD > 0)
    rectangle('Position',[-0.455,0.465,0.03,0.03],'Curvature',[1,1],'FaceColor','r');
elseif((modeD >= -0.1)&&(modeD <= 0))
    rectangle('Position',[-0.455,0.465,0.03,0.03],'Curvature',[1,1],'FaceColor','y');
else
    rectangle('Position',[-0.455,0.465,0.03,0.03],'Curvature',[1,1],'FaceColor','g');
end

hold on
strV = 'brkgmcybrkgmcybrkgmcy';     %color order for clusters
for j = 1:nPhsr             %for each computed phasor
    cPhsr = 5*modeVals(j,7)/maxMag;  %scale phasors for map
    if(abs(cPhsr) > 0)      %if phasor is non zero
        grpCol = strV(find(clusID(j,:) == 1, 1));   %get color for cluster group
        m_quiver(modeVals(j,3),modeVals(j,2),real(cPhsr),imag(cPhsr), 0, grpCol, 'Marke
            %^^plot phasors on map
    end
end

unGrpdMsk = (sum(clusID,2) == 0);       %get zero mag phasors
unGrpdLongs = modeVals(unGrpdMsk,3);    %get latitudes of zero mag phasors
unGrpdLats  = modeVals(unGrpdMsk,2);    %get longitudes of zero mag phasors
m_plot(unGrpdLongs,unGrpdLats, 'o', 'MarkerEdgeColor', [0.4 0.4 0.4]);
m_plot(unGrpdLongs,unGrpdLats, 'x', 'MarkerEdgeColor', [0.4 0.4 0.4]);
    %^^plot zero mag phasors as grey circle with 'x'
set(gca, 'XTick', [], 'YTick', []);     %remove x and y tick mark from map
axis([-0.5 0.5 0.45 1.05]);             %set axes of map

%phasor Diagram
axes('Position', [0.65, 0.05, 0.3, 0.4]);   %set location of diagram
for i = 1:nClus                 %for each identified cluster
    hl = compass(maxMag*cos(cCent(i)),maxMag*sin(cCent(i)),[strV(i) ':']);
        %^^add cluster centroid to phasor diagram
    set(hl,'LineWidth',2);      %set line size of centroid
    xd = get(hl,'xdata');       %retrieve arrow handles
    set(hl,'xdata',[xd(1:2),nan,nan,nan]);  %remove arrowhead
    hold on
    hl = compass(modeVals(clusID(:,i),7), strV(i));
        %^^add cluster phasors to phasor diagram
    set(hl,'LineWidth',2);      %set phasor line sizes
end

winST = winST - 1;  %format window start for plot
winED = winED - 1;  %format window end for plot
t = time-time(1);   %reference timestamps
minF = min(min(FreqData));  %min measured value
maxF = max(max(FreqData));  %max measured value
axes('Position', [0.05, 0.8, 0.9, 0.15]);   %create and position data plot axes
axis([min(t) max(t) minF maxF]);            %scale axes
rectangle('Position', [winST/10, minF, (winED-winST)/10, maxF - minF], 'FaceColor', [1
    %^^plot data window inidcator
hold on
line([winST winST]/10,[minF maxF], 'Color', 'r', 'LineWidth', 3);
```

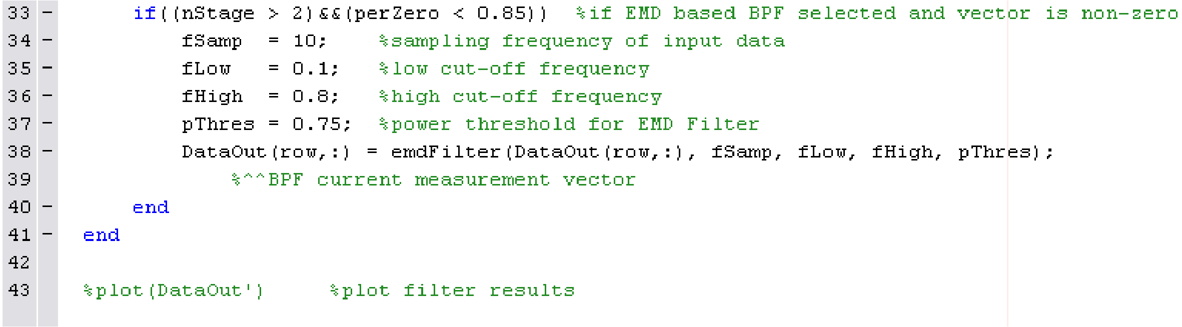


```matlab
         %^^plot data window inidcator
     line([winED winED]/10,[minF maxF], 'Color', 'r', 'LineWidth', 3);
         %^^plot data window inidcator
     plot(t, FreqData)       %plot original measurement data
     xlabel(['Elapsed Time Since ' unixtime2datestr(time(1)) ' (sec)'],'fontsize',14,'fontweight','b');
         %^^x axis label
     ylabel('Freq (Hz)','fontsize',14,'fontweight','b');     %y axis label
     grid on     %turn on grid lines

 function [ inData ] = removeNaNs( inData )
     %removes NaNs from a FNET_Data structure using a hold
     for i = 1:length(inData.FDRnum)     %for each FDR in FNET data structure
         for j = 2:length(inData.time)   %  for entire timespan of data
             if(isnan(inData.Freq(i,j))) %  if NaN found in freq data
                 inData.Freq(i,j) = inData.Freq(i,j-1);  %replace NaN with previous measurement value
             end
             if(isnan(inData.VoltMag(i,j)))  %if NaN found in Voltage Magnitude
                 inData.VoltMag(i,j) = inData.VoltMag(i,j-1);  %replace NaN with previous measurement value
             end
             if(isnan(inData.VoltAng(i,j)))  %if NaN found in Voltage Angle
                 inData.VoltAng(i,j) = inData.VoltAng(i,j-1);  %replace NaN with previous measurement value
             end
         end
     end

```

II THREE STAGE FILTER SUB-PROCESS – inputFilter.m

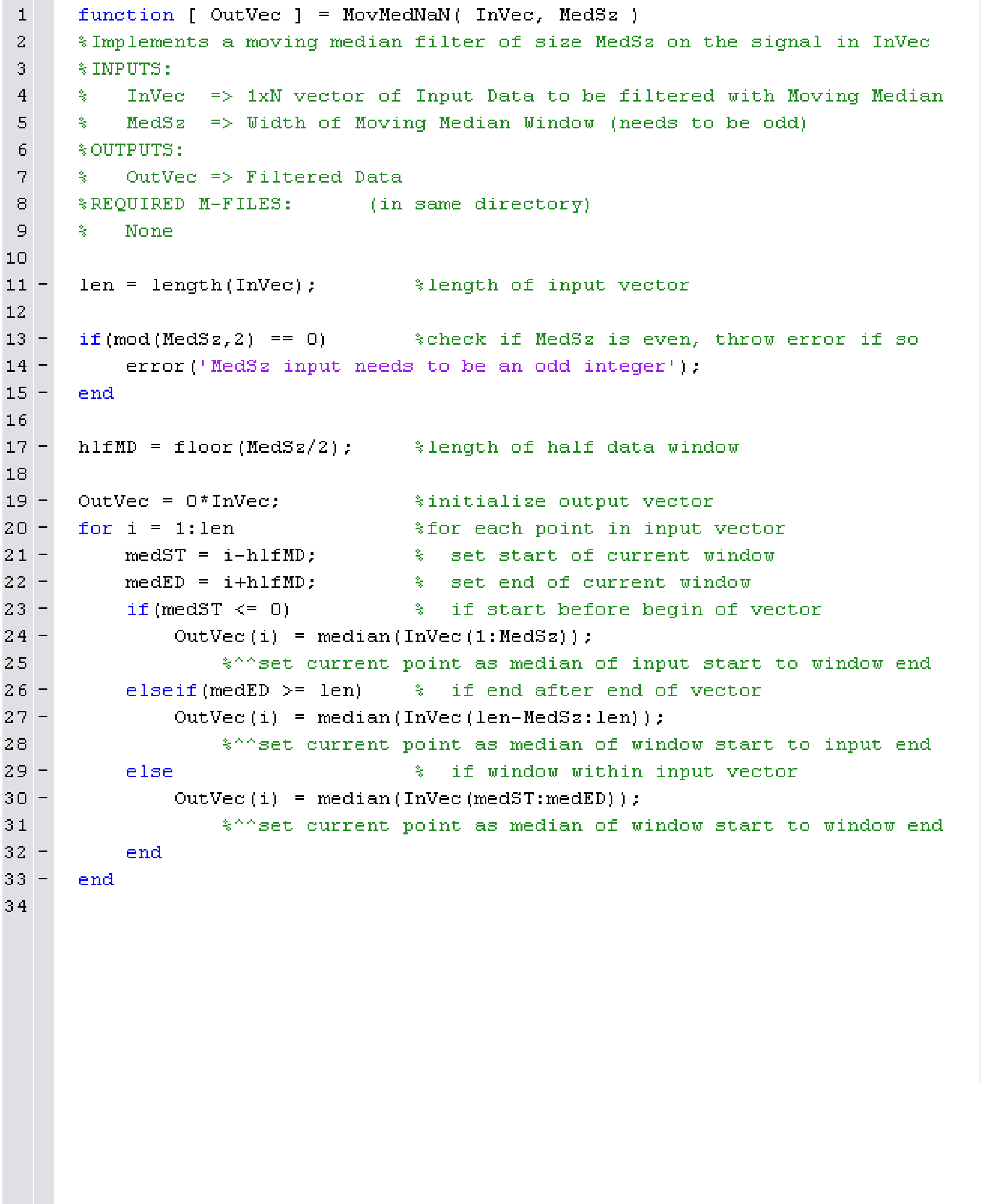


```matlab
function [ DataOut ] = inputFilter( DataIn, nStage )
%Implements full filtering procedure for modal extraction. nStage
%determines the number of filter stages implemented.
%INPUTS:
%   InData  => MxN vector of signal data
%               M == number of data vectors
%               N == number of time points
%   nStage  => number of stages to implement
%               nStage == 1  >> detrends data with median
%               nStage == 2  >> detrends and filters with moving median (denoising)
%               nStage == 3  >> detrends, denoises, and EMD based BPF
%OUTPUTS:
%   DataOut => Filtered Data in MxN array
%REQUIRED M-FILES:      (in same directory)
%   MovMedNaN.m
%   emdFilter.m

[M N] = size(DataIn);       %size of input data matrix
DataOut = DataIn;           %initialize output data matrix

for row = 1:M               %for each measurement vector
    if(nStage > 0)          %  if detrending selected
        DataOut(row,:) = DataIn(row,:) - median(DataIn(row,:));
            %^^detrend current measurement vector
    end
    if(nStage > 1)          %  if denoising selected
        medSZ = 5;          %  set size of moving median
        DataOut(row,:) = MovMedNaN(DataOut(row,:), medSZ);
            %^^denoise current measurement vector
    end
    perZero = sum(DataOut(row,:) == 0)/length(DataOut(row,:));
        %^^determine percentage of zeros in current measurement vector
```



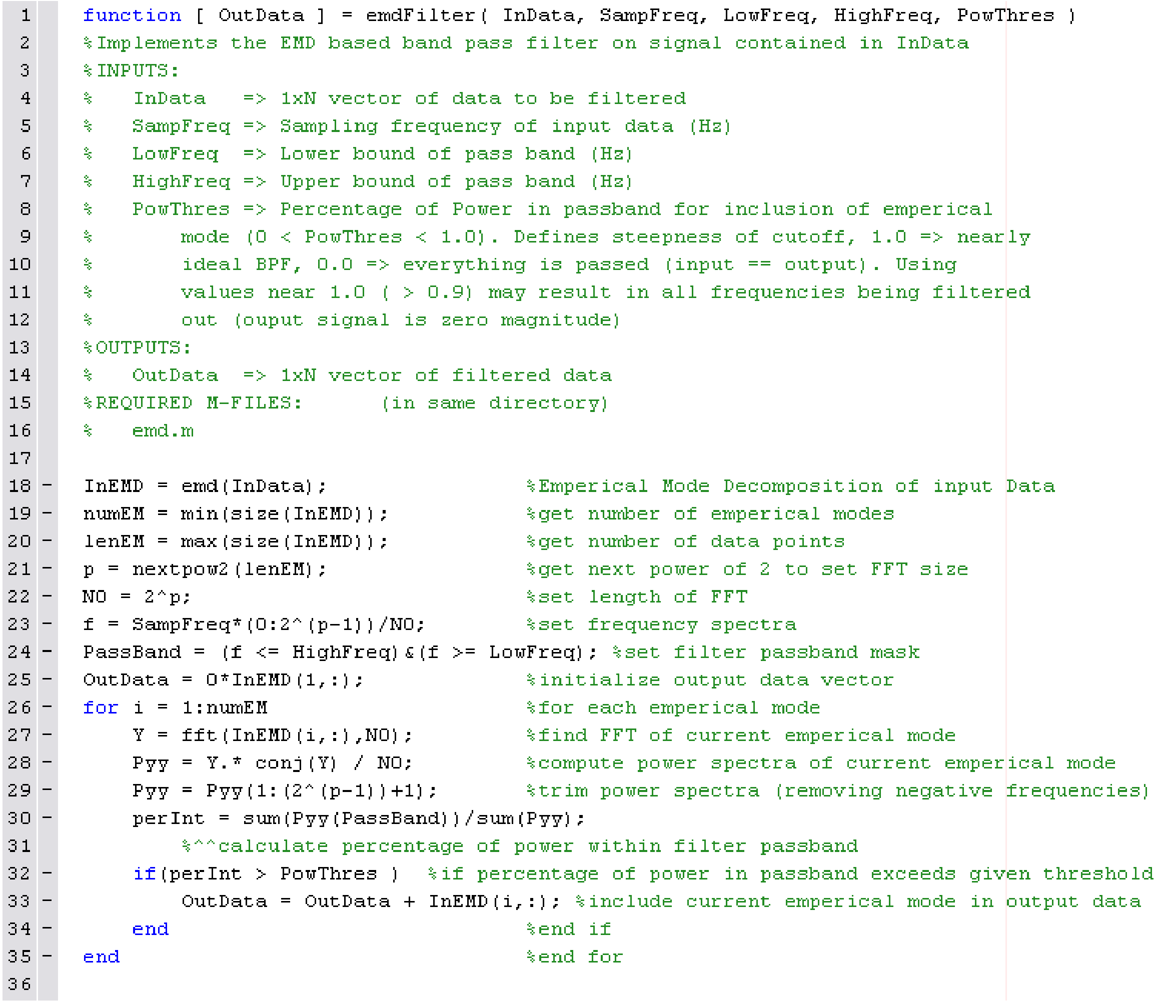
```
    if((nStage > 2)&&(perZero < 0.85))  %if EMD based BPF selected and vector is non-zero
        fSamp  = 10;    %sampling frequency of input data
        fLow   = 0.1;   %low cut-off frequency
        fHigh  = 0.8;   %high cut-off frequency
        pThres = 0.75;  %power threshold for EMD Filter
        DataOut(row,:) = emdFilter(DataOut(row,:), fSamp, fLow, fHigh, pThres);
            %^^BPF current measurement vector
    end
end

%plot(DataOut')     %plot filter results
```

III MOVING MEDIAN FILTER SUB-PROCESS – MovMedNaN.m



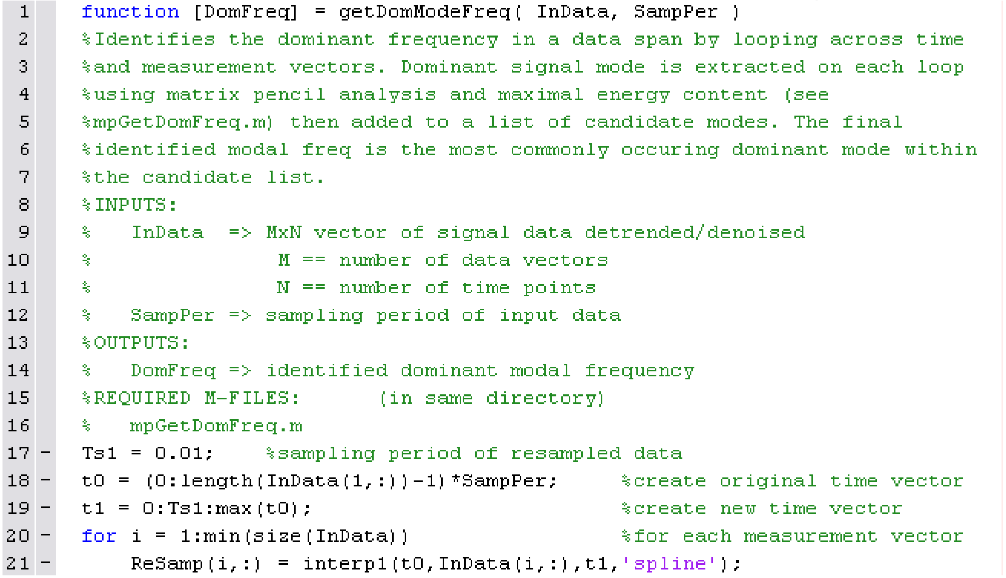
```
function [ OutVec ] = MovMedNaN( InVec, MedSz )
%Implements a moving median filter of size MedSz on the signal in InVec
%INPUTS:
%   InVec  => 1xN vector of Input Data to be filtered with Moving Median
%   MedSz  => Width of Moving Median Window (needs to be odd)
%OUTPUTS:
%   OutVec => Filtered Data
%REQUIRED M-FILES:      (in same directory)
%   None

len = length(InVec);        %length of input vector

if(mod(MedSz,2) == 0)       %check if MedSz is even, throw error if so
    error('MedSz input needs to be an odd integer');
end

hlfMD = floor(MedSz/2);     %length of half data window

OutVec = 0*InVec;           %initialize output vector
for i = 1:len               %for each point in input vector
    medST = i-hlfMD;        %  set start of current window
    medED = i+hlfMD;        %  set end of current window
    if(medST <= 0)          %  if start before begin of vector
        OutVec(i) = median(InVec(1:MedSz));
            %^^set current point as median of input start to window end
    elseif(medED >= len)    %  if end after end of vector
        OutVec(i) = median(InVec(len-MedSz:len));
            %^^set current point as median of window start to input end
    else                    %  if window within input vector
        OutVec(i) = median(InVec(medST:medED));
            %^^set current point as median of window start to window end
    end
end
```

IV EMD BASED BPF SUB-PROCESS – emdFilter.m

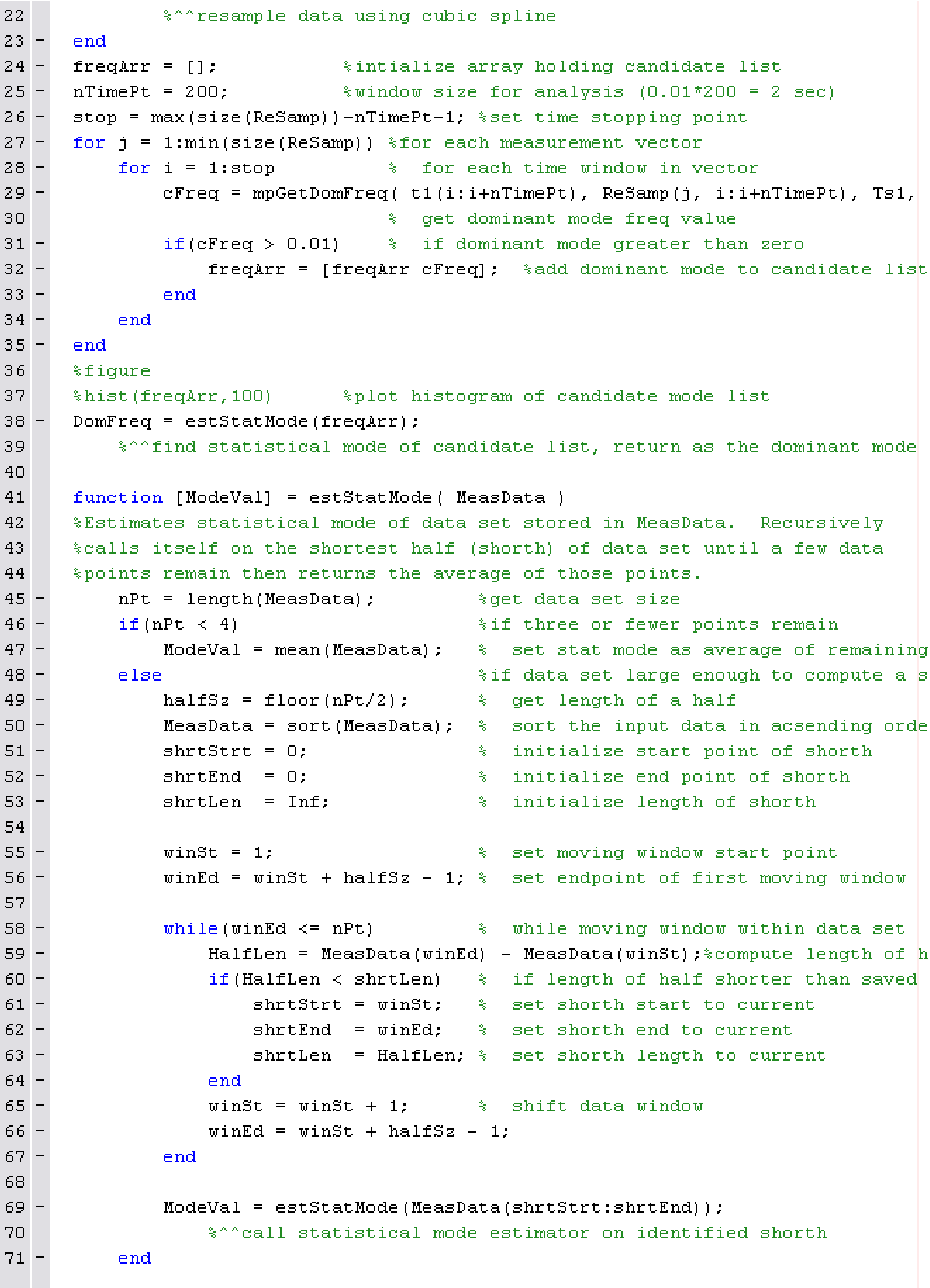


```matlab
function [ OutData ] = emdFilter( InData, SampFreq, LowFreq, HighFreq, PowThres )
%Implements the EMD based band pass filter on signal contained in InData
%INPUTS:
%   InData   => 1xN vector of data to be filtered
%   SampFreq => Sampling frequency of input data (Hz)
%   LowFreq  => Lower bound of pass band (Hz)
%   HighFreq => Upper bound of pass band (Hz)
%   PowThres => Percentage of Power in passband for inclusion of emperical
%       mode (0 < PowThres < 1.0). Defines steepness of cutoff, 1.0 => nearly
%       ideal BPF, 0.0 => everything is passed (input == output). Using
%       values near 1.0 ( > 0.9) may result in all frequencies being filtered
%       out (ouput signal is zero magnitude)
%OUTPUTS:
%   OutData  => 1xN vector of filtered data
%REQUIRED M-FILES:      (in same directory)
%   emd.m

InEMD = emd(InData);                 %Emperical Mode Decomposition of input Data
numEM = min(size(InEMD));            %get number of emperical modes
lenEM = max(size(InEMD));            %get number of data points
p = nextpow2(lenEM);                 %get next power of 2 to set FFT size
NO = 2^p;                            %set length of FFT
f = SampFreq*(0:2^(p-1))/NO;         %set frequency spectra
PassBand = (f <= HighFreq)&(f >= LowFreq); %set filter passband mask
OutData = 0*InEMD(1,:);              %initialize output data vector
for i = 1:numEM                      %for each emperical mode
    Y = fft(InEMD(i,:),NO);          %find FFT of current emperical mode
    Pyy = Y.* conj(Y) / NO;          %compute power spectra of current emperical mode
    Pyy = Pyy(1:(2^(p-1))+1);        %trim power spectra (removing negative frequencies)
    perInt = sum(Pyy(PassBand))/sum(Pyy);
        %^^calculate percentage of power within filter passband
    if(perInt > PowThres )  %if percentage of power in passband exceeds given threshold
        OutData = OutData + InEMD(i,:); %include current emperical mode in output data
    end                              %end if
end                                  %end for

```

V DOMINANT MODE RETRIEVAL SUB-PROCESS – getDomModeFreq.m

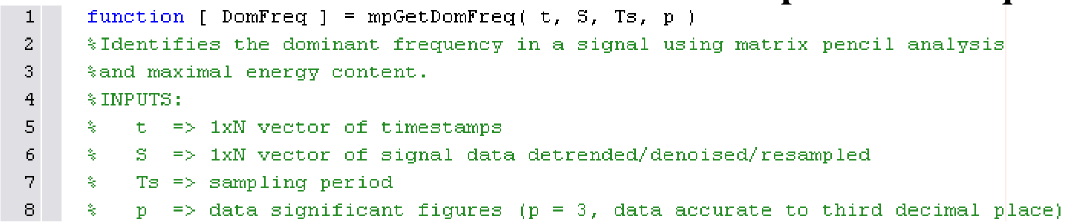


```matlab
function [DomFreq] = getDomModeFreq( InData, SampPer )
%Identifies the dominant frequency in a data span by looping across time
%and measurement vectors. Dominant signal mode is extracted on each loop
%using matrix pencil analysis and maximal energy content (see
%mpGetDomFreq.m) then added to a list of candidate modes. The final
%identified modal freq is the most commonly occuring dominant mode within
%the candidate list.
%INPUTS:
%   InData  => MxN vector of signal data detrended/denoised
%               M == number of data vectors
%               N == number of time points
%   SampPer => sampling period of input data
%OUTPUTS:
%   DomFreq => identified dominant modal frequency
%REQUIRED M-FILES:      (in same directory)
%   mpGetDomFreq.m
Ts1 = 0.01;    %sampling period of resampled data
t0 = (0:length(InData(1,:))-1)*SampPer;     %create original time vector
t1 = 0:Ts1:max(t0);                         %create new time vector
for i = 1:min(size(InData))                 %for each measurement vector
    ReSamp(i,:) = interp1(t0,InData(i,:),t1,'spline');
```

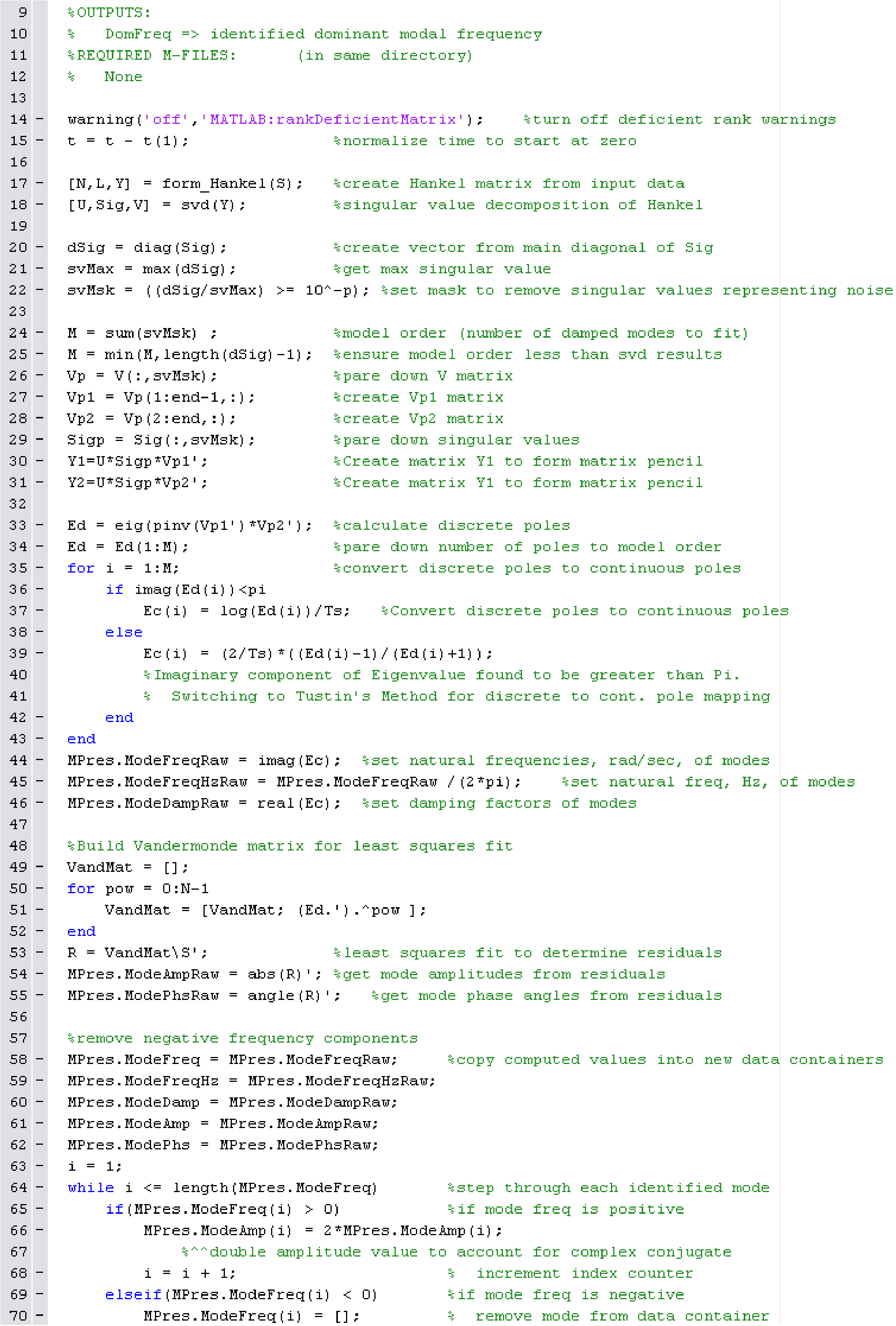


```
        %^^resample data using cubic spline
end
freqArr = [];           %intialize array holding candidate list
nTimePt = 200;          %window size for analysis (0.01*200 = 2 sec)
stop = max(size(ReSamp))-nTimePt-1; %set time stopping point
for j = 1:min(size(ReSamp)) %for each measurement vector
    for i = 1:stop          %  for each time window in vector
        cFreq = mpGetDomFreq( t1(i:i+nTimePt), ReSamp(j, i:i+nTimePt), Ts1, 3 );
                            %  get dominant mode freq value
        if(cFreq > 0.01)    %  if dominant mode greater than zero
            freqArr = [freqArr cFreq];  %add dominant mode to candidate list
        end
    end
end
%figure
%hist(freqArr,100)      %plot histogram of candidate mode list
DomFreq = estStatMode(freqArr);
    %^^find statistical mode of candidate list, return as the dominant mode freq

function [ModeVal] = estStatMode( MeasData )
%Estimates statistical mode of data set stored in MeasData.  Recursively
%calls itself on the shortest half (shorth) of data set until a few data
%points remain then returns the average of those points.
    nPt = length(MeasData);         %get data set size
    if(nPt < 4)                     %if three or fewer points remain
        ModeVal = mean(MeasData);   %  set stat mode as average of remaining points
    else                            %if data set large enough to compute a shorth
        halfSz = floor(nPt/2);      %  get length of a half
        MeasData = sort(MeasData);  %  sort the input data in acsending order
        shrtStrt = 0;               %  initialize start point of shorth
        shrtEnd  = 0;               %  initialize end point of shorth
        shrtLen  = Inf;             %  initialize length of shorth

        winSt = 1;                  %  set moving window start point
        winEd = winSt + halfSz - 1; %  set endpoint of first moving window

        while(winEd <= nPt)         %  while moving window within data set
            HalfLen = MeasData(winEd) - MeasData(winSt);%compute length of half
            if(HalfLen < shrtLen)   %  if length of half shorter than saved
                shrtStrt = winSt;   %  set shorth start to current
                shrtEnd  = winEd;   %  set shorth end to current
                shrtLen  = HalfLen; %  set shorth length to current
            end
            winSt = winSt + 1;      %  shift data window
            winEd = winSt + halfSz - 1;
        end

        ModeVal = estStatMode(MeasData(shrtStrt:shrtEnd));
            %^^call statistical mode estimator on identified shorth
    end
```

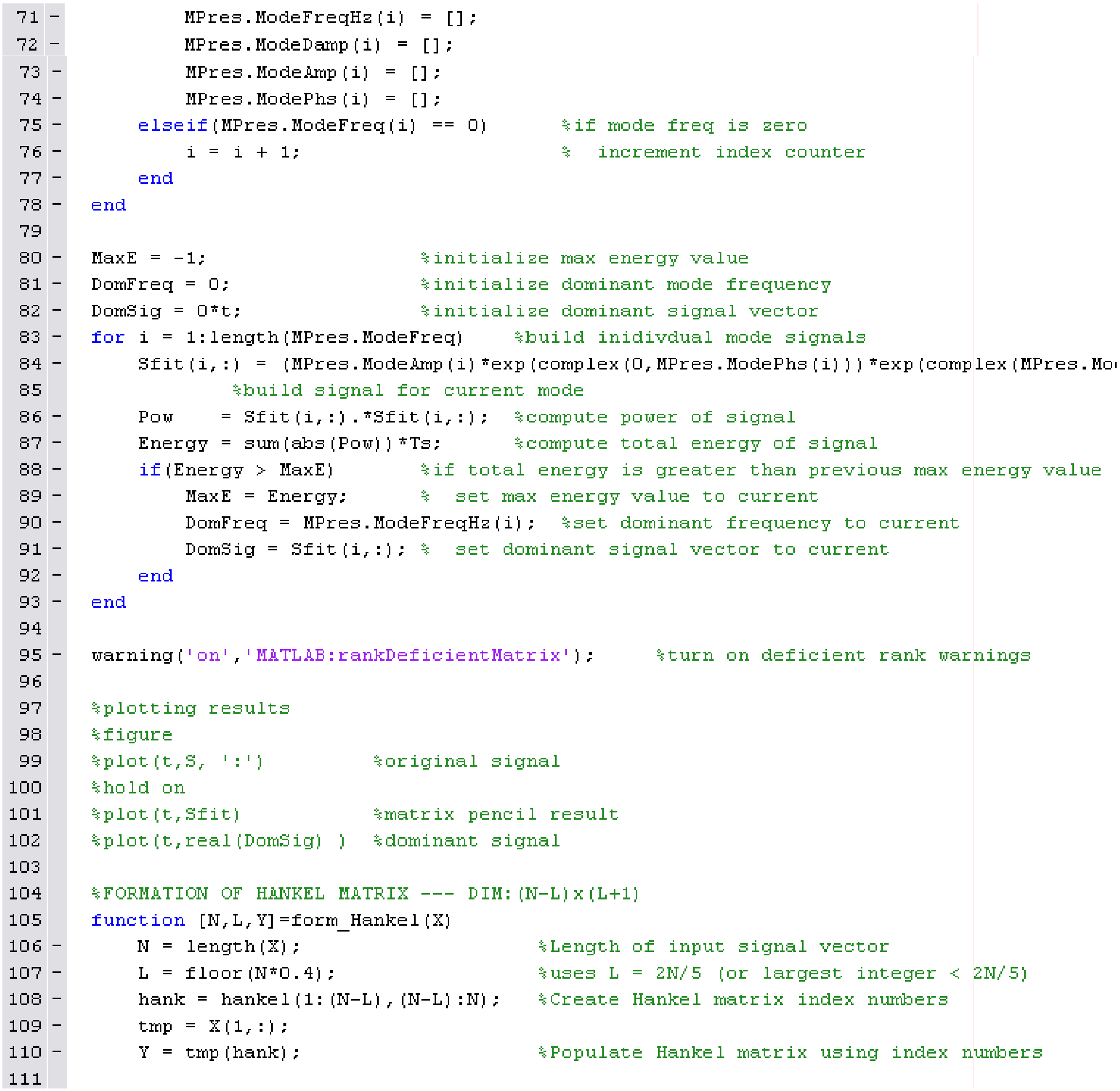
VI MATRIX PENCIL MODE ID SUB-PROCESS – mpGetDomFreq.m



```
function [ DomFreq ] = mpGetDomFreq( t, S, Ts, p )
%Identifies the dominant frequency in a signal using matrix pencil analysis
%and maximal energy content.
%INPUTS:
%   t  => 1xN vector of timestamps
%   S  => 1xN vector of signal data detrended/denoised/resampled
%   Ts => sampling period
%   p  => data significant figures (p = 3, data accurate to third decimal place)
```

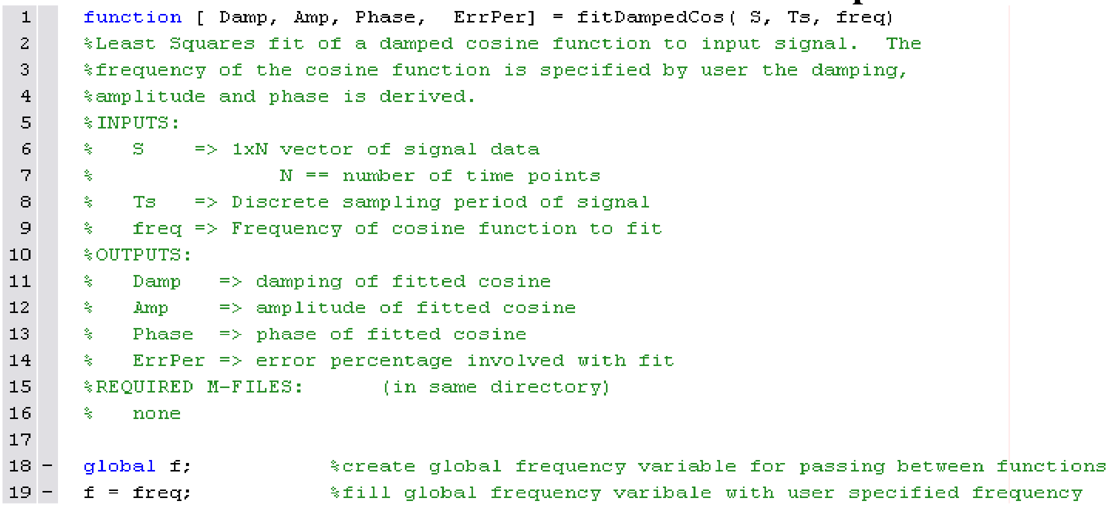


```
%OUTPUTS:
%   DomFreq => identified dominant modal frequency
%REQUIRED M-FILES:      (in same directory)
%   None

warning('off','MATLAB:rankDeficientMatrix');    %turn off deficient rank warnings
t = t - t(1);               %normalize time to start at zero

[N,L,Y] = form_Hankel(S);   %create Hankel matrix from input data
[U,Sig,V] = svd(Y);         %singular value decomposition of Hankel

dSig = diag(Sig);           %create vector from main diagonal of Sig
svMax = max(dSig);          %get max singular value
svMsk = ((dSig/svMax) >= 10^-p); %set mask to remove singular values representing noise

M = sum(svMsk) ;            %model order (number of damped modes to fit)
M = min(M,length(dSig)-1);  %ensure model order less than svd results
Vp = V(:,svMsk);            %pare down V matrix
Vp1 = Vp(1:end-1,:);        %create Vp1 matrix
Vp2 = Vp(2:end,:);          %create Vp2 matrix
Sigp = Sig(:,svMsk);        %pare down singular values
Y1=U*Sigp*Vp1';             %Create matrix Y1 to form matrix pencil
Y2=U*Sigp*Vp2';             %Create matrix Y1 to form matrix pencil

Ed = eig(pinv(Vp1')*Vp2');  %calculate discrete poles
Ed = Ed(1:M);               %pare down number of poles to model order
for i = 1:M;                %convert discrete poles to continuous poles
    if imag(Ed(i))<pi
        Ec(i) = log(Ed(i))/Ts;   %Convert discrete poles to continuous poles
    else
        Ec(i) = (2/Ts)*((Ed(i)-1)/(Ed(i)+1));
        %Imaginary component of Eigenvalue found to be greater than Pi.
        %  Switching to Tustin's Method for discrete to cont. pole mapping
    end
end
MPres.ModeFreqRaw = imag(Ec);  %set natural frequencies, rad/sec, of modes
MPres.ModeFreqHzRaw = MPres.ModeFreqRaw /(2*pi);    %set natural freq, Hz, of modes
MPres.ModeDampRaw = real(Ec);  %set damping factors of modes

%Build Vandermonde matrix for least squares fit
VandMat = [];
for pow = 0:N-1
    VandMat = [VandMat; (Ed.').^pow ];
end
R = VandMat\S';             %least squares fit to determine residuals
MPres.ModeAmpRaw = abs(R)'; %get mode amplitudes from residuals
MPres.ModePhsRaw = angle(R)';   %get mode phase angles from residuals

%remove negative frequency components
MPres.ModeFreq = MPres.ModeFreqRaw;     %copy computed values into new data containers
MPres.ModeFreqHz = MPres.ModeFreqHzRaw;
MPres.ModeDamp = MPres.ModeDampRaw;
MPres.ModeAmp = MPres.ModeAmpRaw;
MPres.ModePhs = MPres.ModePhsRaw;
i = 1;
while i <= length(MPres.ModeFreq)       %step through each identified mode
    if(MPres.ModeFreq(i) > 0)           %if mode freq is positive
        MPres.ModeAmp(i) = 2*MPres.ModeAmp(i);
            %^^double amplitude value to account for complex conjugate
        i = i + 1;                      %  increment index counter
    elseif(MPres.ModeFreq(i) < 0)       %if mode freq is negative
        MPres.ModeFreq(i) = [];         %  remove mode from data container
```

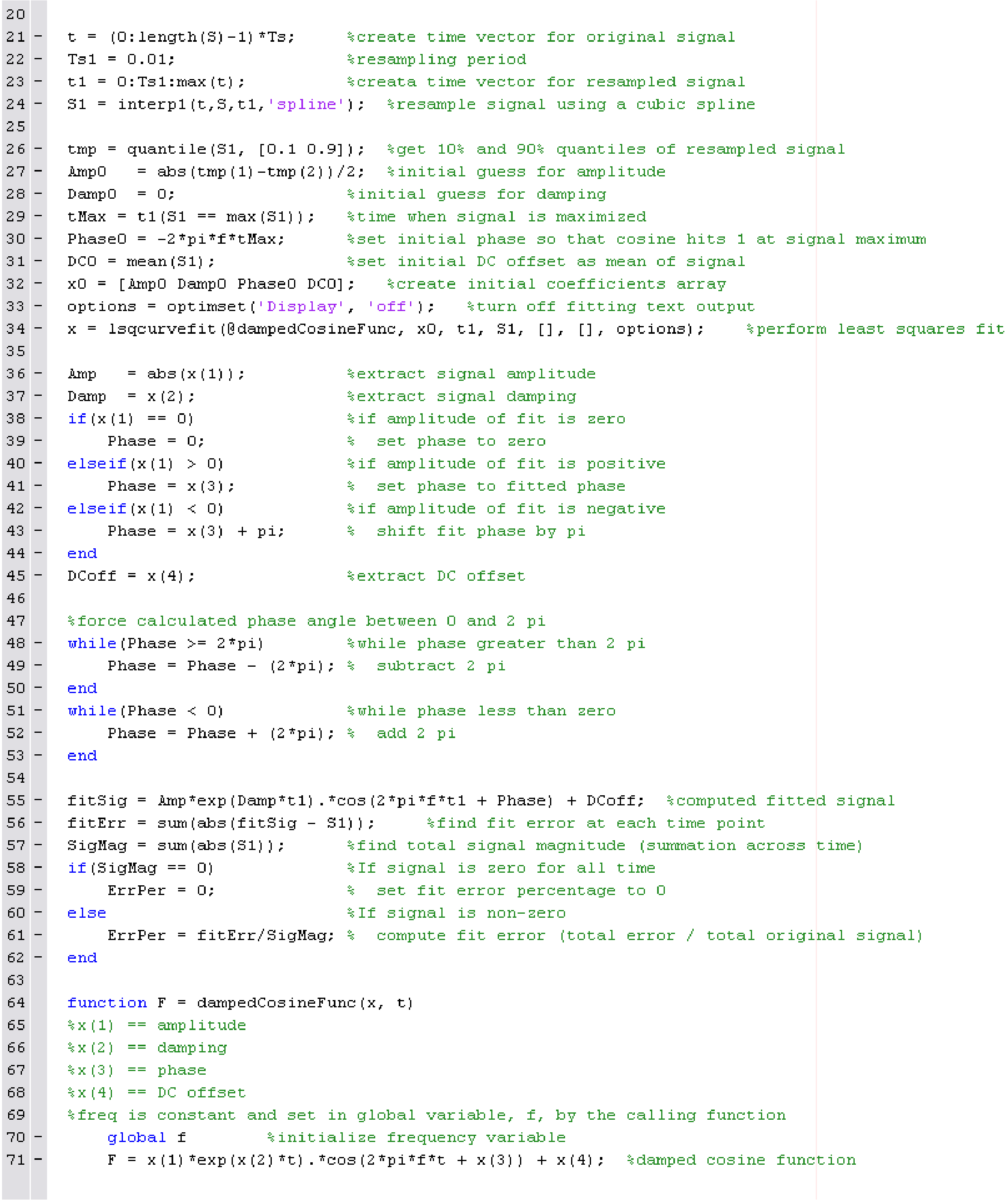


```
        MPres.ModeFreqHz(i) = [];
        MPres.ModeDamp(i) = [];
        MPres.ModeAmp(i) = [];
        MPres.ModePhs(i) = [];
    elseif(MPres.ModeFreq(i) == 0)      %if mode freq is zero
        i = i + 1;                      %  increment index counter
    end
end

MaxE = -1;                  %initialize max energy value
DomFreq = 0;                %initialize dominant mode frequency
DomSig = 0*t;               %initialize dominant signal vector
for i = 1:length(MPres.ModeFreq)    %build inidivdual mode signals
    Sfit(i,:) = (MPres.ModeAmp(i)*exp(complex(0,MPres.ModePhs(i)))*exp(complex(MPres.Mo
            %build signal for current mode
    Pow    = Sfit(i,:).*Sfit(i,:);  %compute power of signal
    Energy = sum(abs(Pow))*Ts;      %compute total energy of signal
    if(Energy > MaxE)       %if total energy is greater than previous max energy value
        MaxE = Energy;      %  set max energy value to current
        DomFreq = MPres.ModeFreqHz(i);  %set dominant frequency to current
        DomSig = Sfit(i,:); %  set dominant signal vector to current
    end
end

warning('on','MATLAB:rankDeficientMatrix');     %turn on deficient rank warnings

%plotting results
%figure
%plot(t,S, ':')         %original signal
%hold on
%plot(t,Sfit)           %matrix pencil result
%plot(t,real(DomSig) )  %dominant signal

%FORMATION OF HANKEL MATRIX --- DIM:(N-L)x(L+1)
function [N,L,Y]=form_Hankel(X)
    N = length(X);                    %Length of input signal vector
    L = floor(N*0.4);                 %uses L = 2N/5 (or largest integer < 2N/5)
    hank = hankel(1:(N-L),(N-L):N);   %Create Hankel matrix index numbers
    tmp = X(1,:);
    Y = tmp(hank);                    %Populate Hankel matrix using index numbers

```

VII DAMPED COSINE FITTING SUB-PROCESS – fitDampedCos.m

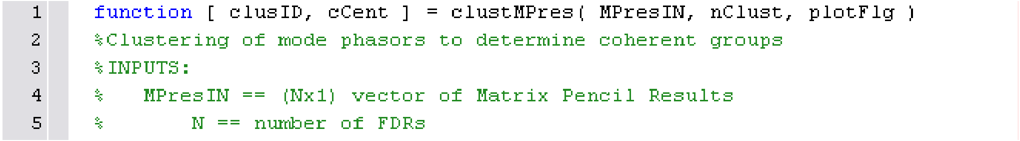


```
function [ Damp, Amp, Phase,  ErrPer] = fitDampedCos( S, Ts, freq)
%Least Squares fit of a damped cosine function to input signal.  The
%frequency of the cosine function is specified by user the damping,
%amplitude and phase is derived.
%INPUTS:
%   S    => 1xN vector of signal data
%               N == number of time points
%   Ts   => Discrete sampling period of signal
%   freq => Frequency of cosine function to fit
%OUTPUTS:
%   Damp   => damping of fitted cosine
%   Amp    => amplitude of fitted cosine
%   Phase  => phase of fitted cosine
%   ErrPer => error percentage involved with fit
%REQUIRED M-FILES:      (in same directory)
%   none

global f;           %create global frequency variable for passing between functions
f = freq;           %fill global frequency varibale with user specified frequency
```

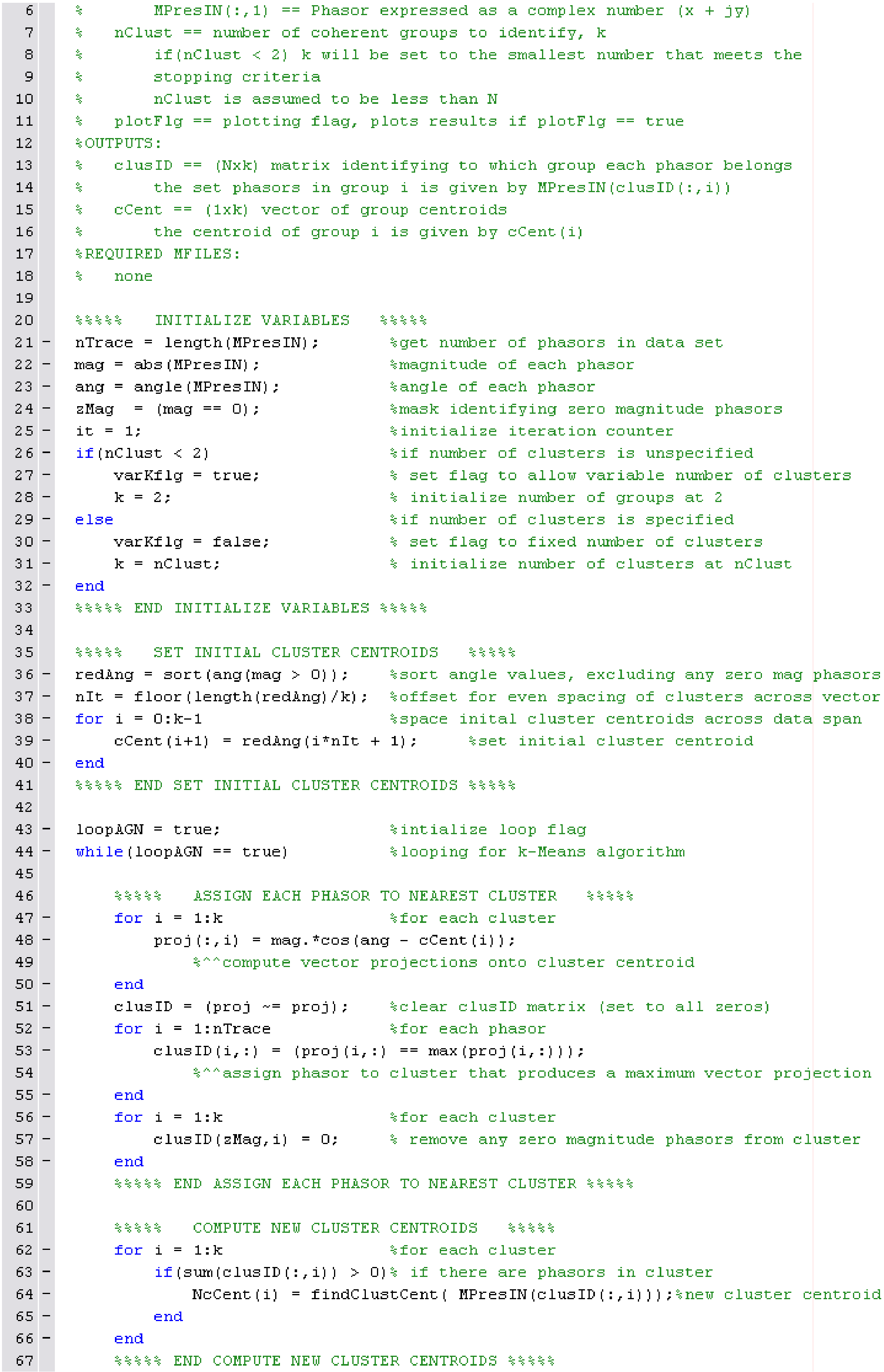


```
t = (0:length(S)-1)*Ts;     %create time vector for original signal
Ts1 = 0.01;                 %resampling period
t1 = 0:Ts1:max(t);          %creata time vector for resampled signal
S1 = interp1(t,S,t1,'spline');  %resample signal using a cubic spline

tmp = quantile(S1, [0.1 0.9]);  %get 10% and 90% quantiles of resampled signal
Amp0   = abs(tmp(1)-tmp(2))/2;  %initial guess for amplitude
Damp0  = 0;                 %initial guess for damping
tMax = t1(S1 == max(S1));   %time when signal is maximized
Phase0 = -2*pi*f*tMax;      %set initial phase so that cosine hits 1 at signal maximum
DC0 = mean(S1);             %set initial DC offset as mean of signal
x0 = [Amp0 Damp0 Phase0 DC0];   %create initial coefficients array
options = optimset('Display', 'off');   %turn off fitting text output
x = lsqcurvefit(@dampedCosineFunc, x0, t1, S1, [], [], options);    %perform least squares fit

Amp   = abs(x(1));          %extract signal amplitude
Damp  = x(2);               %extract signal damping
if(x(1) == 0)               %if amplitude of fit is zero
    Phase = 0;              %  set phase to zero
elseif(x(1) > 0)            %if amplitude of fit is positive
    Phase = x(3);           %  set phase to fitted phase
elseif(x(1) < 0)            %if amplitude of fit is negative
    Phase = x(3) + pi;      %  shift fit phase by pi
end
DCoff = x(4);               %extract DC offset

%force calculated phase angle between 0 and 2 pi
while(Phase >= 2*pi)        %while phase greater than 2 pi
    Phase = Phase - (2*pi); %  subtract 2 pi
end
while(Phase < 0)            %while phase less than zero
    Phase = Phase + (2*pi); %  add 2 pi
end

fitSig = Amp*exp(Damp*t1).*cos(2*pi*f*t1 + Phase) + DCoff;  %computed fitted signal
fitErr = sum(abs(fitSig - S1));     %find fit error at each time point
SigMag = sum(abs(S1));      %find total signal magnitude (summation across time)
if(SigMag == 0)             %If signal is zero for all time
    ErrPer = 0;             %  set fit error percentage to 0
else                        %If signal is non-zero
    ErrPer = fitErr/SigMag; %  compute fit error (total error / total original signal)
end

function F = dampedCosineFunc(x, t)
%x(1) == amplitude
%x(2) == damping
%x(3) == phase
%x(4) == DC offset
%freq is constant and set in global variable, f, by the calling function
    global f        %initialize frequency variable
    F = x(1)*exp(x(2)*t).*cos(2*pi*f*t + x(3)) + x(4);  %damped cosine function
```

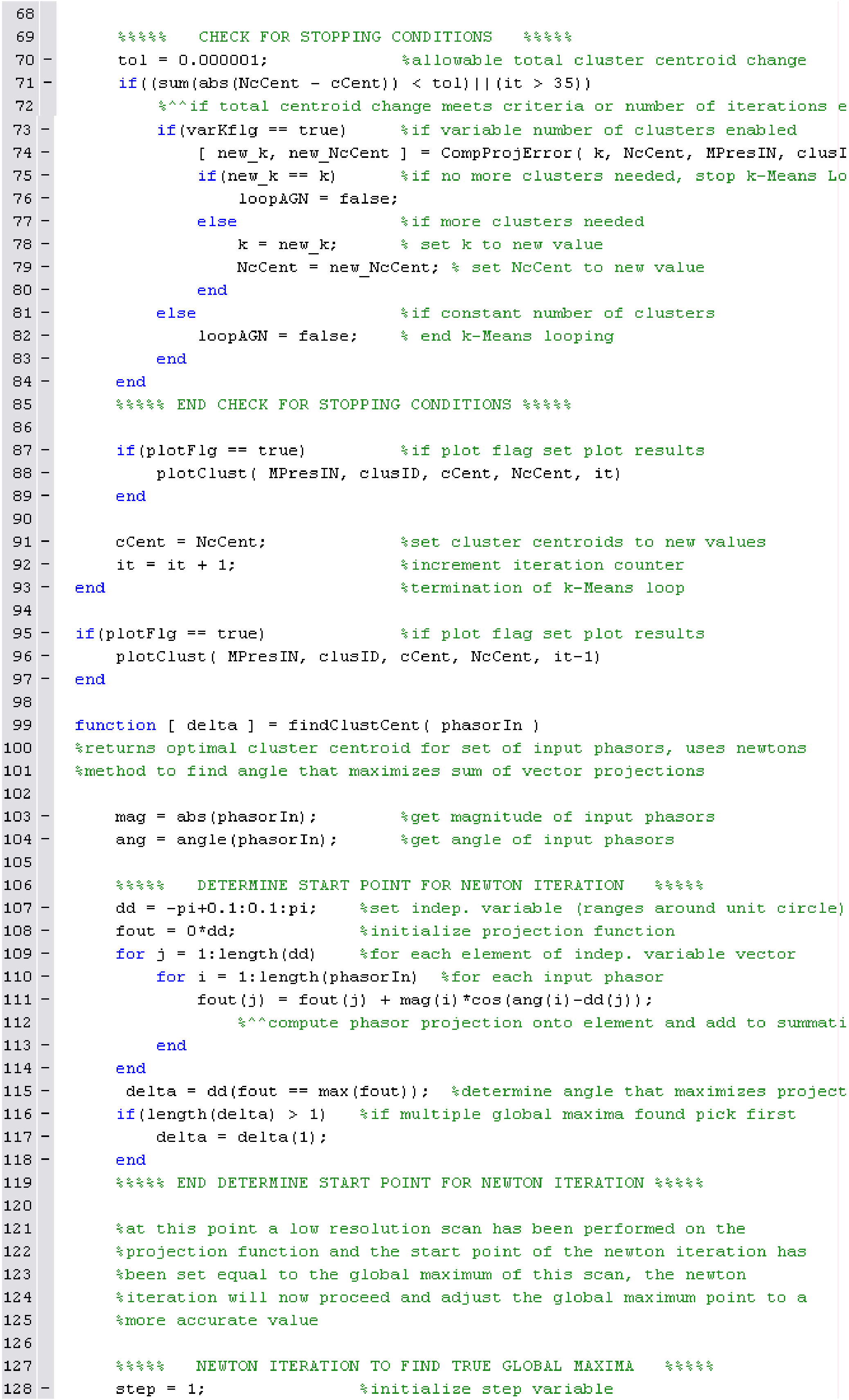
VIII PHASOR CLUSTERING SUB-PROCESS – clustMPres.m



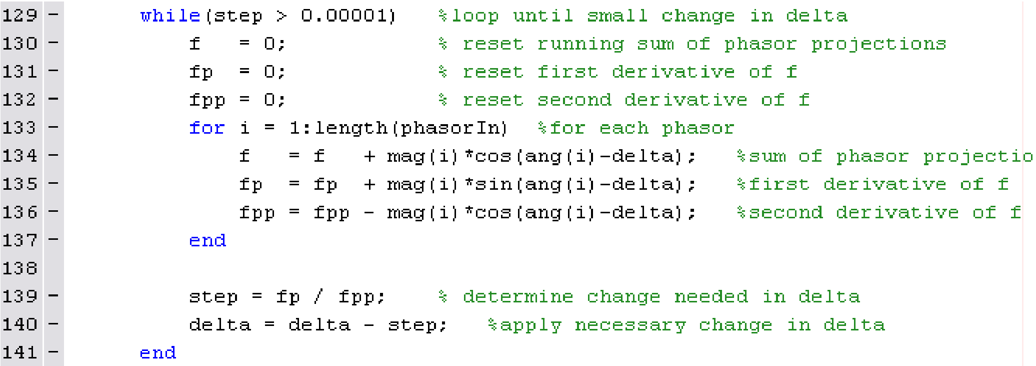
```
function [ clusID, cCent ] = clustMPres( MPresIN, nClust, plotFlg )
%Clustering of mode phasors to determine coherent groups
%INPUTS:
%   MPresIN == (Nx1) vector of Matrix Pencil Results
%       N == number of FDRs
```



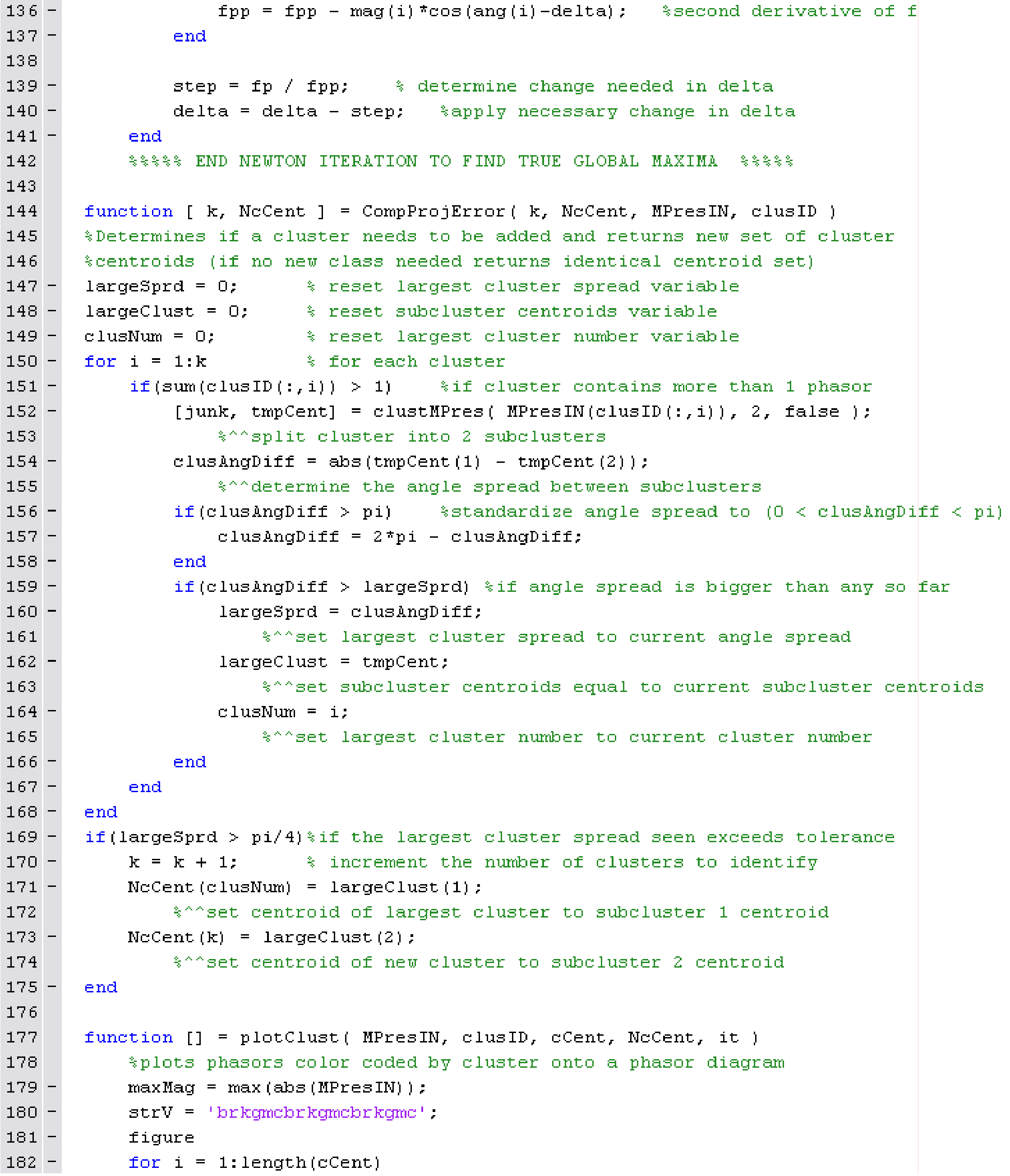
```
%       MPresIN(:,1) == Phasor expressed as a complex number (x + jy)
%   nClust == number of coherent groups to identify, k
%       if(nClust < 2) k will be set to the smallest number that meets the
%       stopping criteria
%       nClust is assumed to be less than N
%   plotFlg == plotting flag, plots results if plotFlg == true
%OUTPUTS:
%   clusID == (Nxk) matrix identifying to which group each phasor belongs
%       the set phasors in group i is given by MPresIN(clusID(:,i))
%   cCent == (1xk) vector of group centroids
%       the centroid of group i is given by cCent(i)
%REQUIRED MFILES:
%   none

%%%%%   INITIALIZE VARIABLES   %%%%%
nTrace = length(MPresIN);       %get number of phasors in data set
mag = abs(MPresIN);             %magnitude of each phasor
ang = angle(MPresIN);           %angle of each phasor
zMag  = (mag == 0);             %mask identifying zero magnitude phasors
it = 1;                         %initialize iteration counter
if(nClust < 2)                  %if number of clusters is unspecified
    varKflg = true;             % set flag to allow variable number of clusters
    k = 2;                      % initialize number of groups at 2
else                            %if number of clusters is specified
    varKflg = false;            % set flag to fixed number of clusters
    k = nClust;                 % initialize number of clusters at nClust
end
%%%%% END INITIALIZE VARIABLES %%%%%

%%%%%   SET INITIAL CLUSTER CENTROIDS   %%%%%
redAng = sort(ang(mag > 0));    %sort angle values, excluding any zero mag phasors
nIt = floor(length(redAng)/k);  %offset for even spacing of clusters across vector
for i = 0:k-1                   %space inital cluster centroids across data span
    cCent(i+1) = redAng(i*nIt + 1);     %set initial cluster centroid
end
%%%%% END SET INITIAL CLUSTER CENTROIDS %%%%%

loopAGN = true;                 %intialize loop flag
while(loopAGN == true)          %looping for k-Means algorithm

    %%%%%   ASSIGN EACH PHASOR TO NEAREST CLUSTER   %%%%%
    for i = 1:k                 %for each cluster
        proj(:,i) = mag.*cos(ang - cCent(i));
            %^^compute vector projections onto cluster centroid
    end
    clusID = (proj ~= proj);    %clear clusID matrix (set to all zeros)
    for i = 1:nTrace            %for each phasor
        clusID(i,:) = (proj(i,:) == max(proj(i,:)));
            %^^assign phasor to cluster that produces a maximum vector projection
    end
    for i = 1:k                 %for each cluster
        clusID(zMag,i) = 0;     % remove any zero magnitude phasors from cluster
    end
    %%%%% END ASSIGN EACH PHASOR TO NEAREST CLUSTER %%%%%

    %%%%%   COMPUTE NEW CLUSTER CENTROIDS   %%%%%
    for i = 1:k                 %for each cluster
        if(sum(clusID(:,i)) > 0)% if there are phasors in cluster
            NcCent(i) = findClustCent( MPresIN(clusID(:,i)));%new cluster centroid
        end
    end
    %%%%% END COMPUTE NEW CLUSTER CENTROIDS %%%%%
```



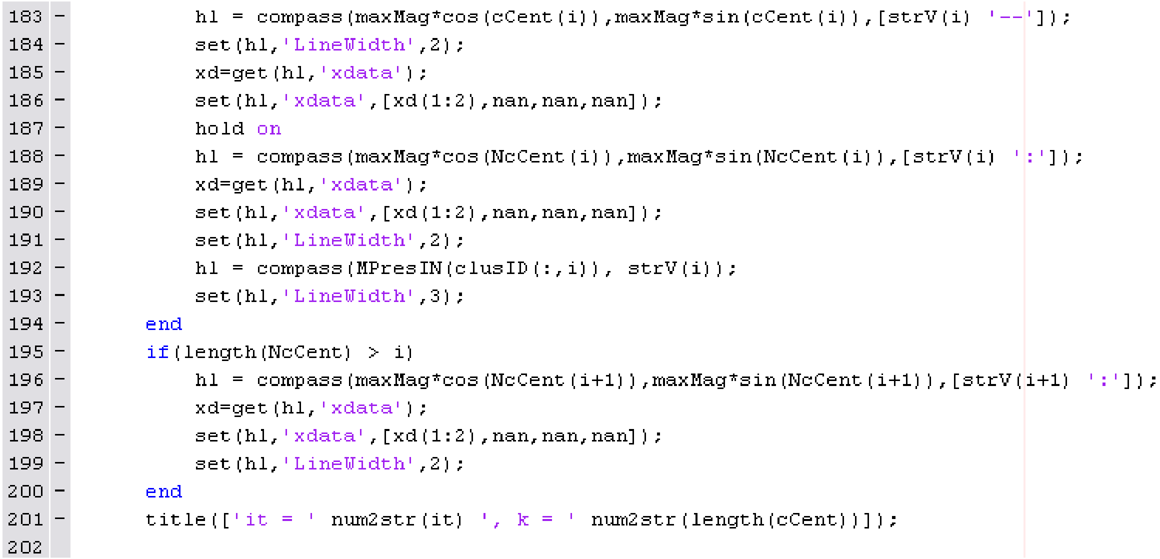
```
    %%%%%   CHECK FOR STOPPING CONDITIONS   %%%%%
    tol = 0.000001;             %allowable total cluster centroid change
    if((sum(abs(NcCent - cCent)) < tol)||(it > 35))
        %^^if total centroid change meets criteria or number of iterations exceeded
        if(varKflg == true)     %if variable number of clusters enabled
            [ new_k, new_NcCent ] = CompProjError( k, NcCent, MPresIN, clusID );
            if(new_k == k)      %if no more clusters needed, stop k-Means Looping
                loopAGN = false;
            else                %if more clusters needed
                k = new_k;      % set k to new value
                NcCent = new_NcCent; % set NcCent to new value
            end
        else                    %if constant number of clusters
            loopAGN = false;    % end k-Means looping
        end
    end
    %%%%% END CHECK FOR STOPPING CONDITIONS %%%%%

    if(plotFlg == true)         %if plot flag set plot results
        plotClust( MPresIN, clusID, cCent, NcCent, it)
    end

    cCent = NcCent;             %set cluster centroids to new values
    it = it + 1;                %increment iteration counter
end                             %termination of k-Means loop

if(plotFlg == true)             %if plot flag set plot results
    plotClust( MPresIN, clusID, cCent, NcCent, it-1)
end

function [ delta ] = findClustCent( phasorIn )
%returns optimal cluster centroid for set of input phasors, uses newtons
%method to find angle that maximizes sum of vector projections

    mag = abs(phasorIn);        %get magnitude of input phasors
    ang = angle(phasorIn);      %get angle of input phasors

    %%%%%   DETERMINE START POINT FOR NEWTON ITERATION   %%%%%
    dd = -pi+0.1:0.1:pi;    %set indep. variable (ranges around unit circle)
    fout = 0*dd;            %initialize projection function
    for j = 1:length(dd)    %for each element of indep. variable vector
        for i = 1:length(phasorIn)  %for each input phasor
            fout(j) = fout(j) + mag(i)*cos(ang(i)-dd(j));
                %^^compute phasor projection onto element and add to summation
        end
    end
     delta = dd(fout == max(fout));  %determine angle that maximizes projection
    if(length(delta) > 1)   %if multiple global maxima found pick first
        delta = delta(1);
    end
    %%%%% END DETERMINE START POINT FOR NEWTON ITERATION %%%%%

    %at this point a low resolution scan has been performed on the
    %projection function and the start point of the newton iteration has
    %been set equal to the global maximum of this scan, the newton
    %iteration will now proceed and adjust the global maximum point to a
    %more accurate value

    %%%%%   NEWTON ITERATION TO FIND TRUE GLOBAL MAXIMA   %%%%%
    step = 1;               %initialize step variable
```



```
    while(step > 0.00001)   %loop until small change in delta
        f   = 0;            % reset running sum of phasor projections
        fp  = 0;            % reset first derivative of f
        fpp = 0;            % reset second derivative of f
        for i = 1:length(phasorIn)  %for each phasor
            f   = f   + mag(i)*cos(ang(i)-delta);   %sum of phasor projections
            fp  = fp  + mag(i)*sin(ang(i)-delta);   %first derivative of f
            fpp = fpp - mag(i)*cos(ang(i)-delta);   %second derivative of f
        end

        step = fp / fpp;    % determine change needed in delta
        delta = delta - step;   %apply necessary change in delta
    end
```



```
            fpp = fpp - mag(i)*cos(ang(i)-delta);   %second derivative of f
        end

        step = fp / fpp;    % determine change needed in delta
        delta = delta - step;   %apply necessary change in delta
    end
    %%%%% END NEWTON ITERATION TO FIND TRUE GLOBAL MAXIMA  %%%%%

function [ k, NcCent ] = CompProjError( k, NcCent, MPresIN, clusID )
%Determines if a cluster needs to be added and returns new set of cluster
%centroids (if no new class needed returns identical centroid set)
largeSprd = 0;      % reset largest cluster spread variable
largeClust = 0;     % reset subcluster centroids variable
clusNum = 0;        % reset largest cluster number variable
for i = 1:k         % for each cluster
    if(sum(clusID(:,i)) > 1)    %if cluster contains more than 1 phasor
        [junk, tmpCent] = clustMPres( MPresIN(clusID(:,i)), 2, false );
            %^^split cluster into 2 subclusters
        clusAngDiff = abs(tmpCent(1) - tmpCent(2));
            %^^determine the angle spread between subclusters
        if(clusAngDiff > pi)    %standardize angle spread to (0 < clusAngDiff < pi)
            clusAngDiff = 2*pi - clusAngDiff;
        end
        if(clusAngDiff > largeSprd) %if angle spread is bigger than any so far
            largeSprd = clusAngDiff;
                %^^set largest cluster spread to current angle spread
            largeClust = tmpCent;
                %^^set subcluster centroids equal to current subcluster centroids
            clusNum = i;
                %^^set largest cluster number to current cluster number
        end
    end
end
if(largeSprd > pi/4)%if the largest cluster spread seen exceeds tolerance
    k = k + 1;      % increment the number of clusters to identify
    NcCent(clusNum) = largeClust(1);
        %^^set centroid of largest cluster to subcluster 1 centroid
    NcCent(k) = largeClust(2);
        %^^set centroid of new cluster to subcluster 2 centroid
end

function [] = plotClust( MPresIN, clusID, cCent, NcCent, it )
    %plots phasors color coded by cluster onto a phasor diagram
    maxMag = max(abs(MPresIN));
    strV = 'brkgmcbrkgmcbrkgmc';
    figure
    for i = 1:length(cCent)
```



```
        hl = compass(maxMag*cos(cCent(i)),maxMag*sin(cCent(i)),[strV(i) '--']);
        set(hl,'LineWidth',2);
        xd=get(hl,'xdata');
        set(hl,'xdata',[xd(1:2),nan,nan,nan]);
        hold on
        hl = compass(maxMag*cos(NcCent(i)),maxMag*sin(NcCent(i)),[strV(i) ':']);
        xd=get(hl,'xdata');
        set(hl,'xdata',[xd(1:2),nan,nan,nan]);
        set(hl,'LineWidth',2);
        hl = compass(MPresIN(clusID(:,i)), strV(i));
        set(hl,'LineWidth',3);
    end
    if(length(NcCent) > i)
        hl = compass(maxMag*cos(NcCent(i+1)),maxMag*sin(NcCent(i+1)),[strV(i+1) ':']);
        xd=get(hl,'xdata');
        set(hl,'xdata',[xd(1:2),nan,nan,nan]);
        set(hl,'LineWidth',2);
    end
    title(['it = ' num2str(it) ', k = ' num2str(length(cCent))]);

```

APPENDIX D

ADDITIONAL RESULTS FROM MODAL EXTRACTION AND VISUALIZATION PACKAGE

I EXAMPLE ONE – EI EVENT ON JUNE 29, 2009

The first additional example for the visualization package demonstrates the system response to another event occurring in the US Eastern Interconnect. This event is most likely the loss of a generator in Northeastern Florida which occurred on June 29, 2009 at 12:25:45 UTC. For this set of test data a 0.267 Hz interarea mode was identified and extracted throughout the data window. The pre-event system condition is given by Figure D.1 here all of the measurement phasors have low magnitudes indicating that the mode has not yet been excited.

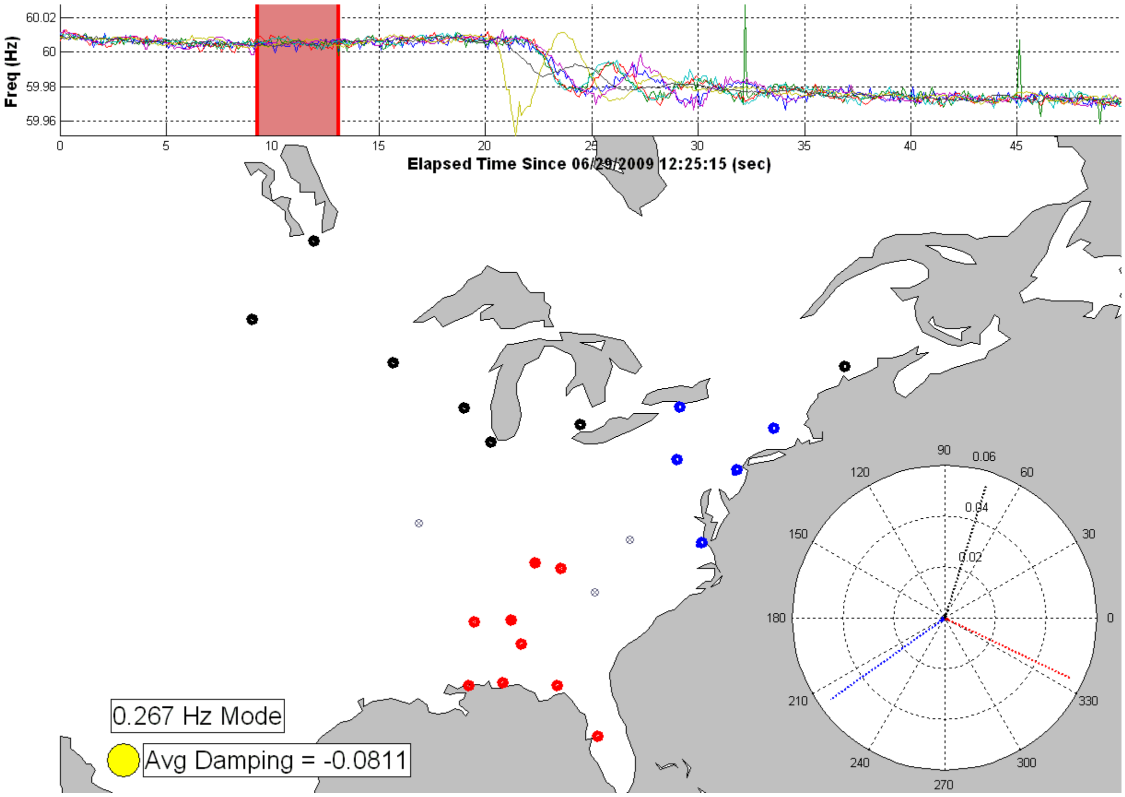


Figure D.1: System before Inciting Event

In Figure D.2 the event has just occurred and the FDR located in Tampa Bay, FL is observing a very large swing in frequency. Because of this the associated phasor is showing sizeable amplitude. Additionally the Tallahassee, FL is also demonstrating some modal content. These two units are grouped together (black group) indicating that Florida is beginning to oscillate against the rest of the system. A few of the northern units are begging to see the oscillation but for the most part the rest of the measurement points are not seeing the mode yet. As this is at the beginning of the oscillation period the oscillations are growing and the computed damping is positive. The average damping factor across the system is 0.172 reflecting this. Because of the positive damping the stop light indicator has now turned red.

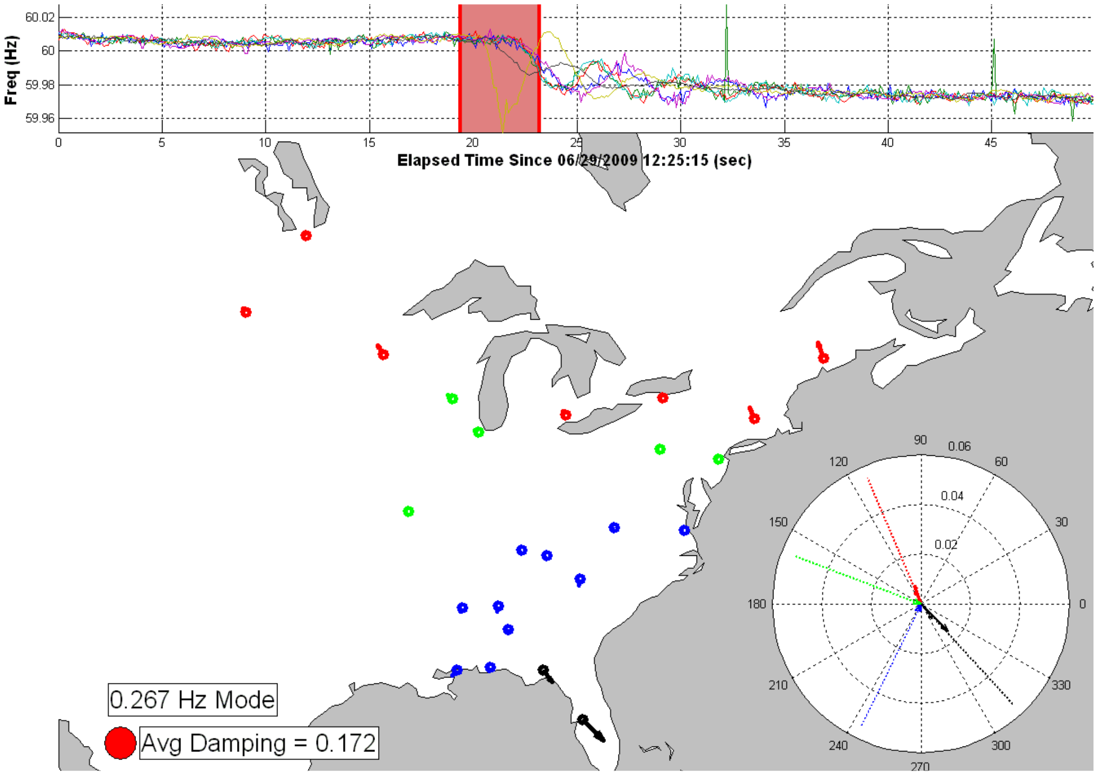


Figure D.2: Event Occurrence and Beginning of Oscillation

Figure D.3 captures a time period at which most of the FDRs see a first full oscillation period. At this point the Florida group (black) is still observing the highest amplitude oscillations. Many of the other units in the south (blue group) are also beginning to see the oscillation and swinging with Florida. The units to the north of the system (red group) are now oscillating out of phase with the southern part of the system. The average damping across the system is now -0.105 as the oscillation has past its first swing and is beginning to decay.

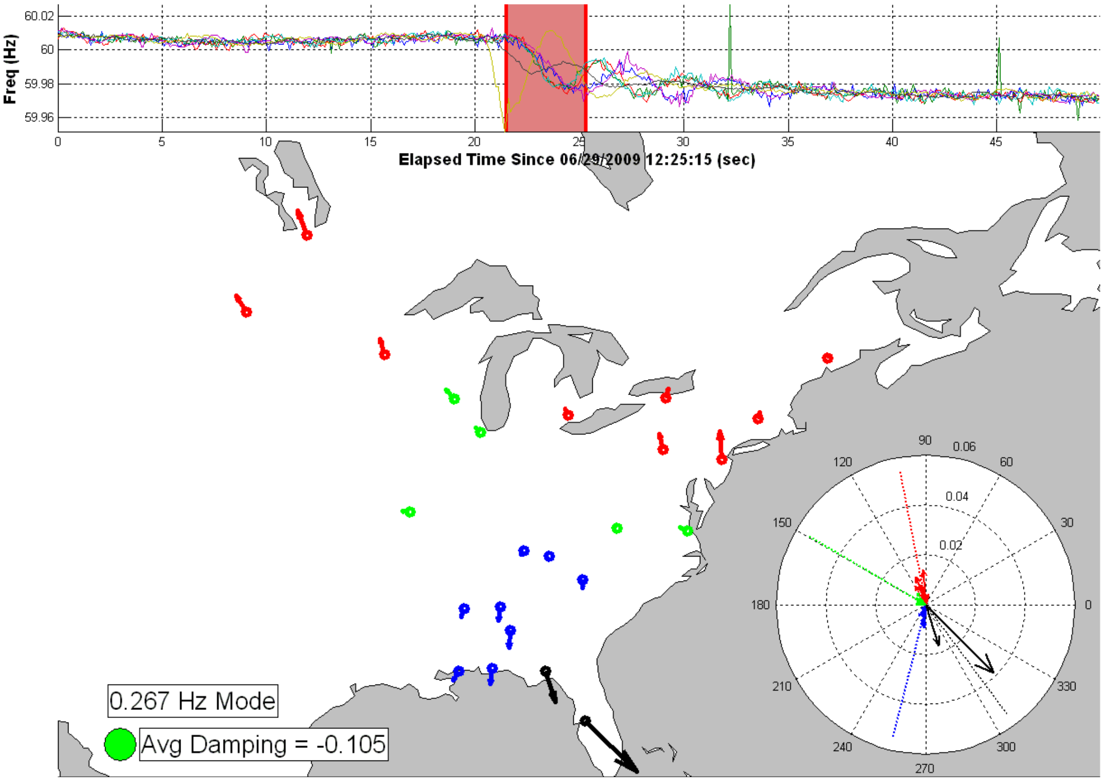


Figure D.3: First Full Cycle of Interarea Oscillation

The same trend continues through Figure D.4 with Florida and the Southeast (green and blue groups) oscillating out of phase with the northern part of the system. The grouping of the northern units has now begun to split into the black and red groups as the East (red group) pulls apart from the West (black group). The interarea mode is now lightly damped with an average system-wide damping factor of -0.00131, inducing a yellow light to indicate a lightly damped oscillation.

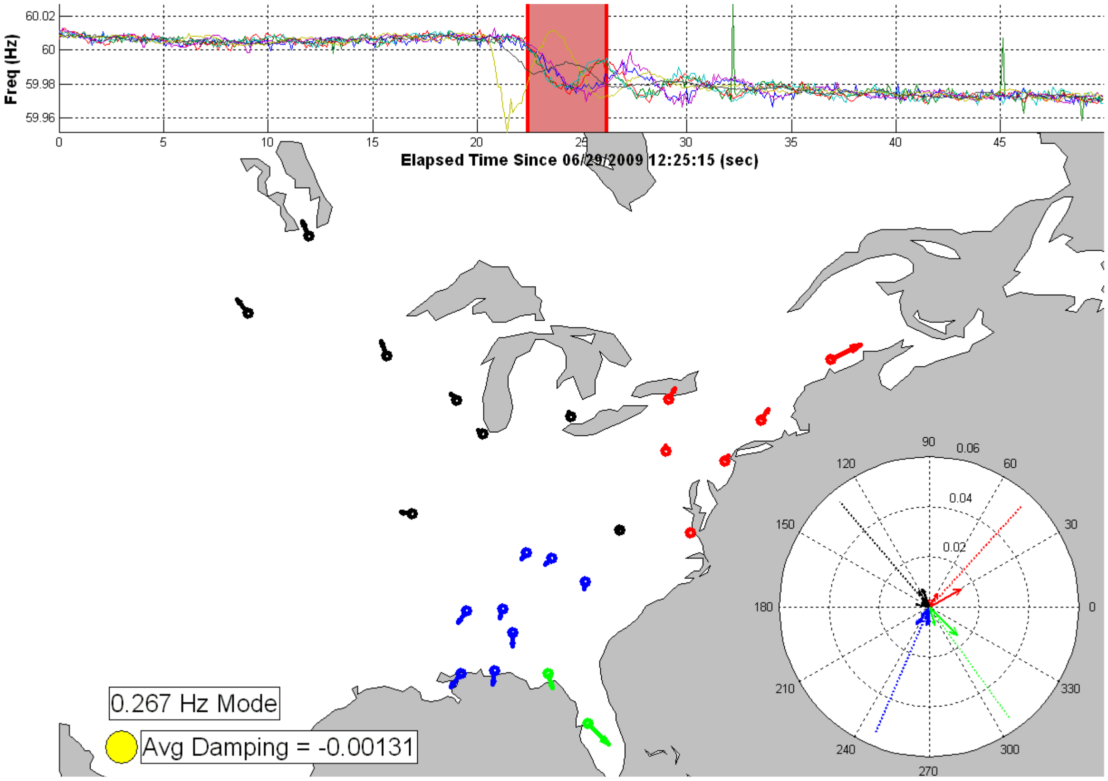


Figure D.4: Continued Separation of Coherent Groups

The separation into three coherent groups is most obvious in Figure D.5. The three identified groups include the South colored in black, the Northeast in red and the Northwest in green. For the most part the Southern group is oscillating against the other two. This oscillation is starting to see increased damping across the system as the average damping value has reached -0.446. This level of damping indicates that the oscillation is decaying quickly and should be dying out shortly; because of this the damping light has turned green.

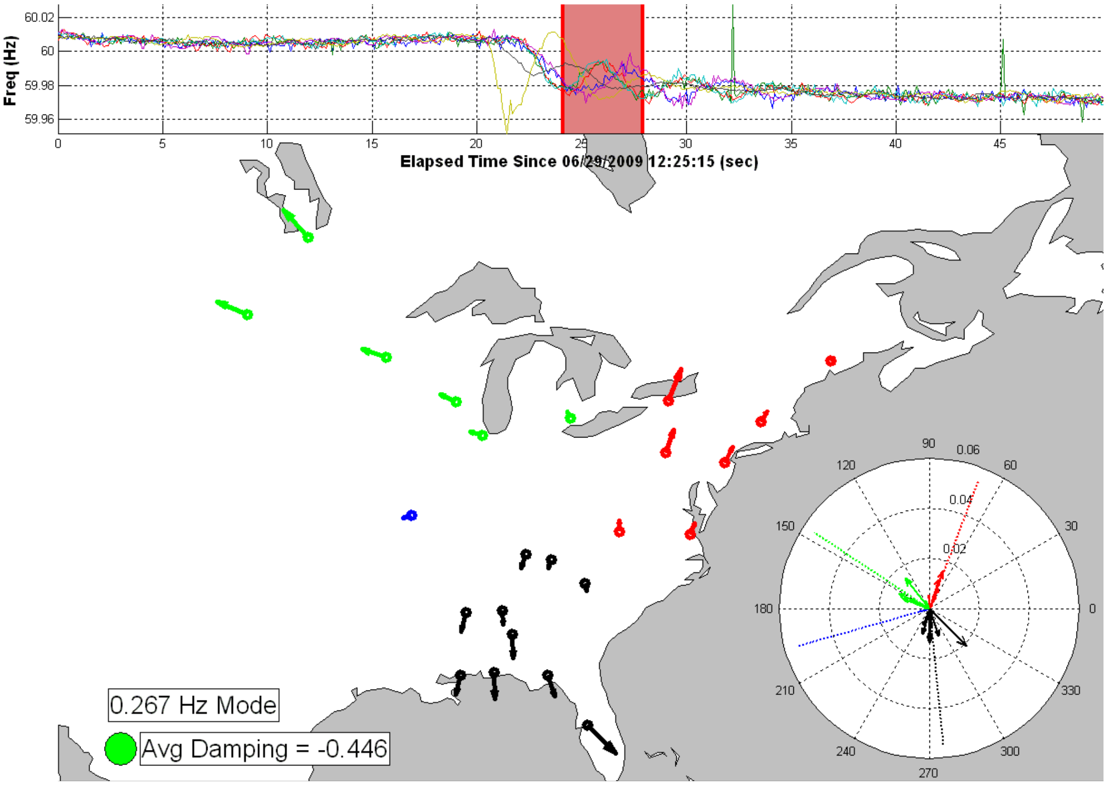


Figure D.5: Separation into Three Groups; Beginning of Heavy Damping

The damping continues through Figure D.6 as the oscillation dies out. The three coherent groups from Figure D.5 persist but the magnitudes of all the phasors have decreased reinforcing the damping observations. The system continues to be heavily damped with an average damping factor of -0.413.

The final steady-state system value is now being approached as the oscillation decays to zero. The post-oscillation system condition is given in Figure D.7 with all of the phasor amplitudes at or near zero, indicating no appreciable oscillatory content at the 0.267 Hz mode.

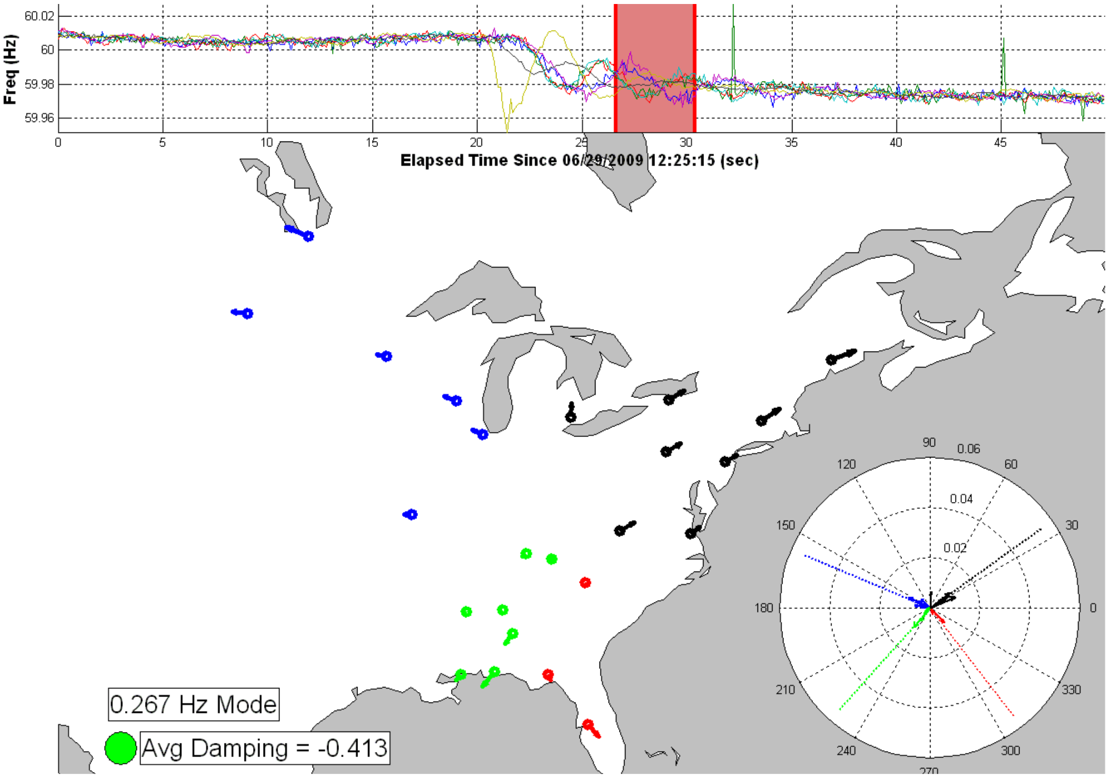


Figure D.6: Final Decay of Oscillation

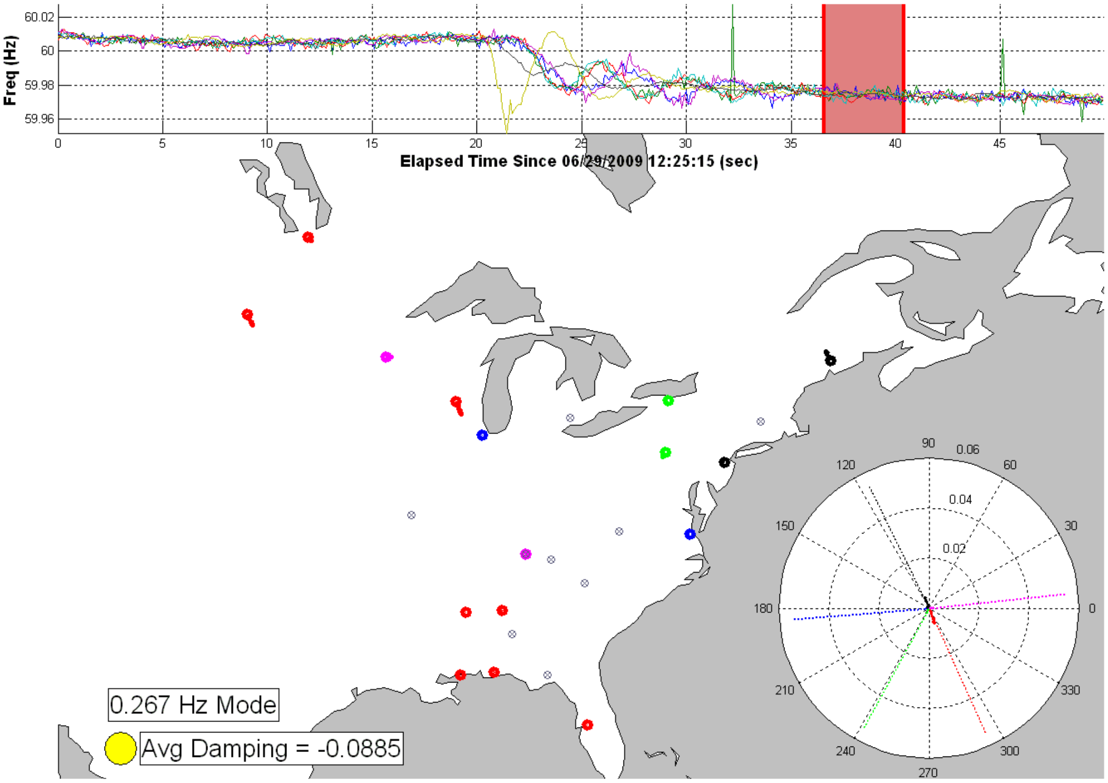


Figure D.7: Post-Oscillation Steady-State

II Example Two – WECC Event on Feb 3, 2005

The third example demonstrating the modal extraction and visualization is drawn from the WECC. The inciting event in this case occurred at the Colstrip generating plant at about 15:30:00 UTC on February 3, 2005. An interarea mode of 0.376 Hz was identified during the response to this event. Extracting this mode across the time frame of the event identified two coherent groups within the system. For this case there are only four active FDRs due to the sparseness of FNET coverage in the WECC and to the time region of the event.

Figure D.8 gives the condition of the system before the generation loss at the Colstrip plant. Here all of the modal phasors are nearly zero as there is no appreciable oscillation in the measurement data.

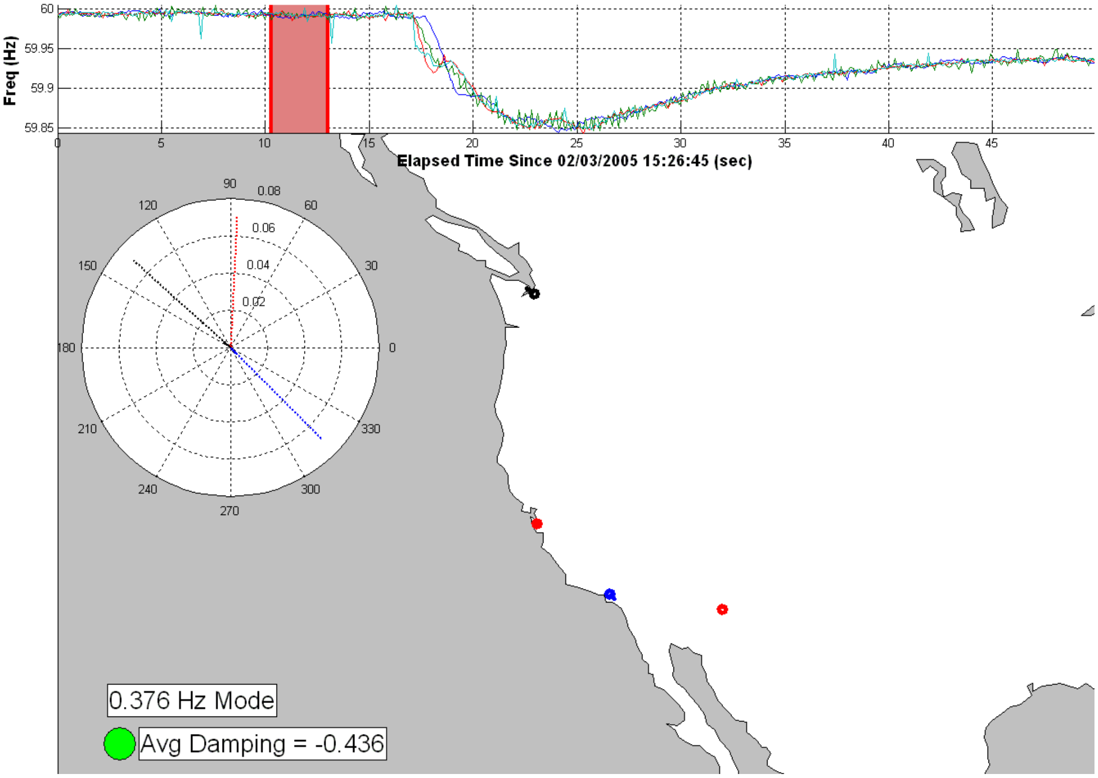


Figure D.8: System before Inciting Event

In Figure D.9 the event has occurred and the interarea mode has been excited. As the oscillation is just beginning to grow the associate phasor magnitudes are still relatively small. The average system damping factor has increased to +0.181 indicating that this oscillation is growing. At this point the two coherent groups are forming with Seattle, WA oscillating against the units to the south (Palo Alto, CA, Pasadena, CA and Tempe, AZ). These two groups have been classified and colored as blue and red respectively.

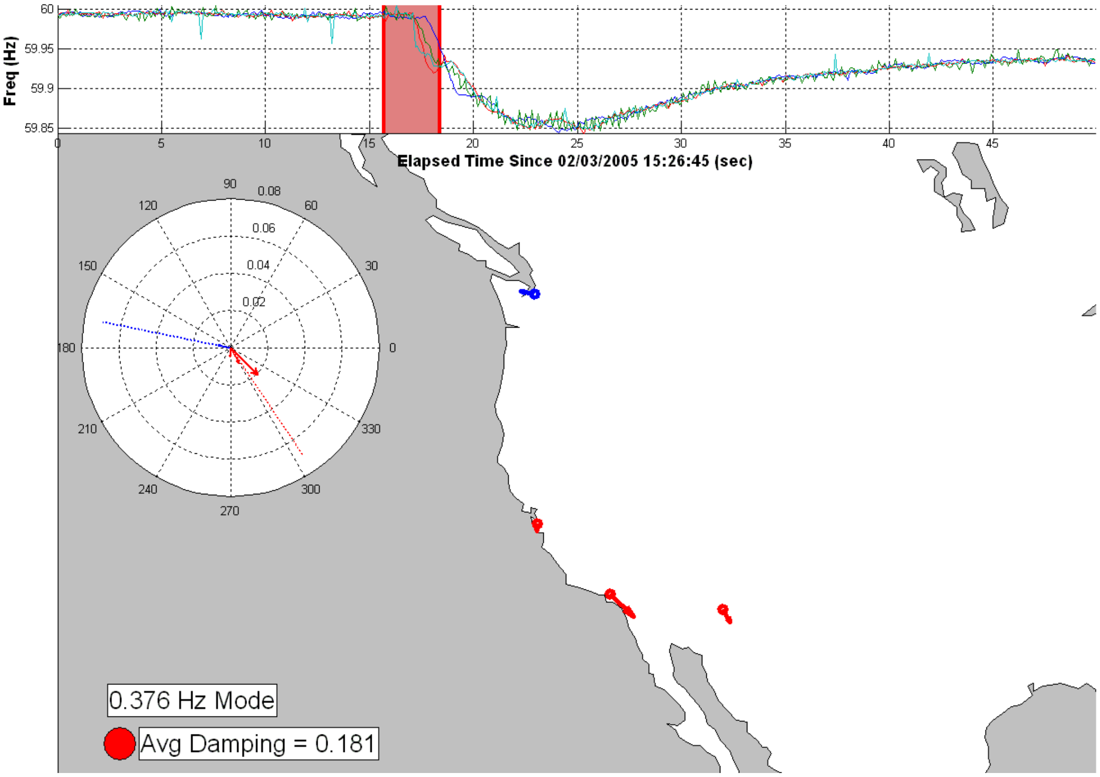


Figure D.9: Event Occurrence and Beginning of Oscillation

As time progresses from Figure D.9 to Figure D.10 the oscillation continues to develop. Now the two groups have separated with Seattle continuing to oscillate out of phase with the southern portion of the WECC system. Figure D.10 also shows that the oscillation has begun to decay with the average damping factor moving to -0.204 Hz.

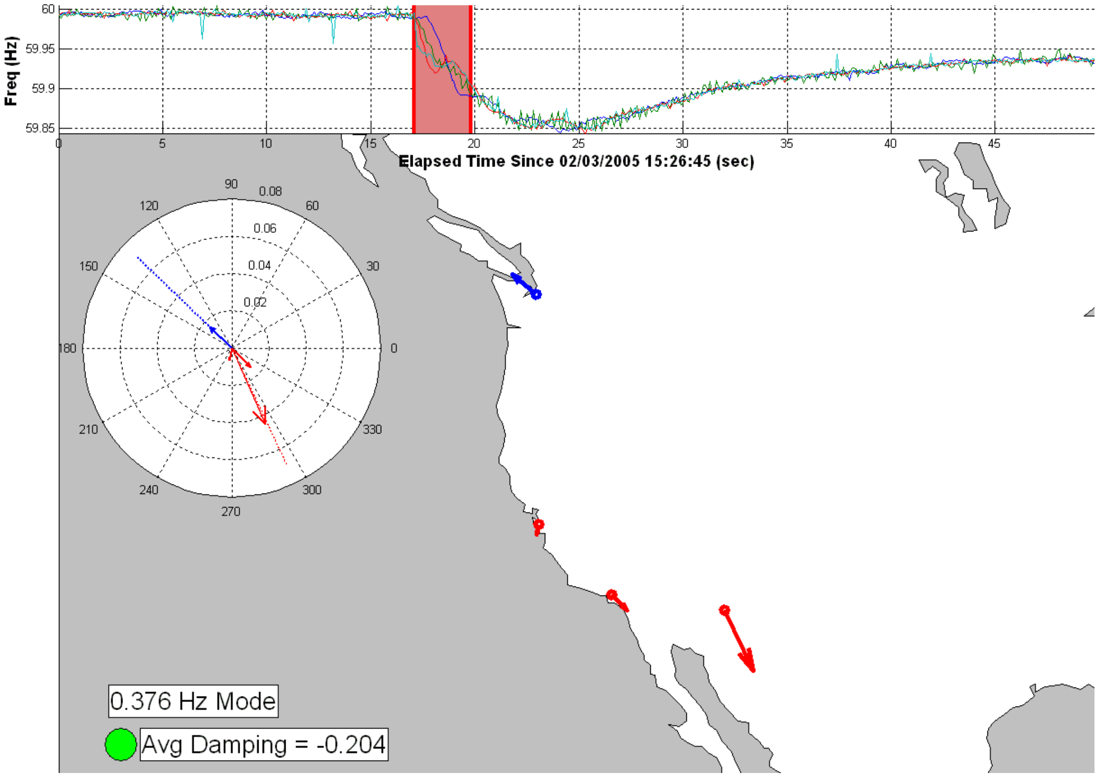


Figure D.10: First Full Cycle of Interarea Oscillation

A few frames later the system approaches the state represented by Figure D.11, here the phasor magnitude reach their highest points, once again the northern part of the system as represented by the Seattle unit is oscillating approximately 180 degrees out of phase with the southern part of the system, most notably Tempe, AZ. It is observed in this frame that the units on the extremes of the system are experiencing the largest swings and unit closest to the center (Palo Alto, CA) is demonstrating the smallest magnitude. At this point the oscillation is continuing to decay as the average damping across the system has approached -0.800. Due to this high amount of damping the stoplight indicator is green, implying a stable, decaying oscillation.

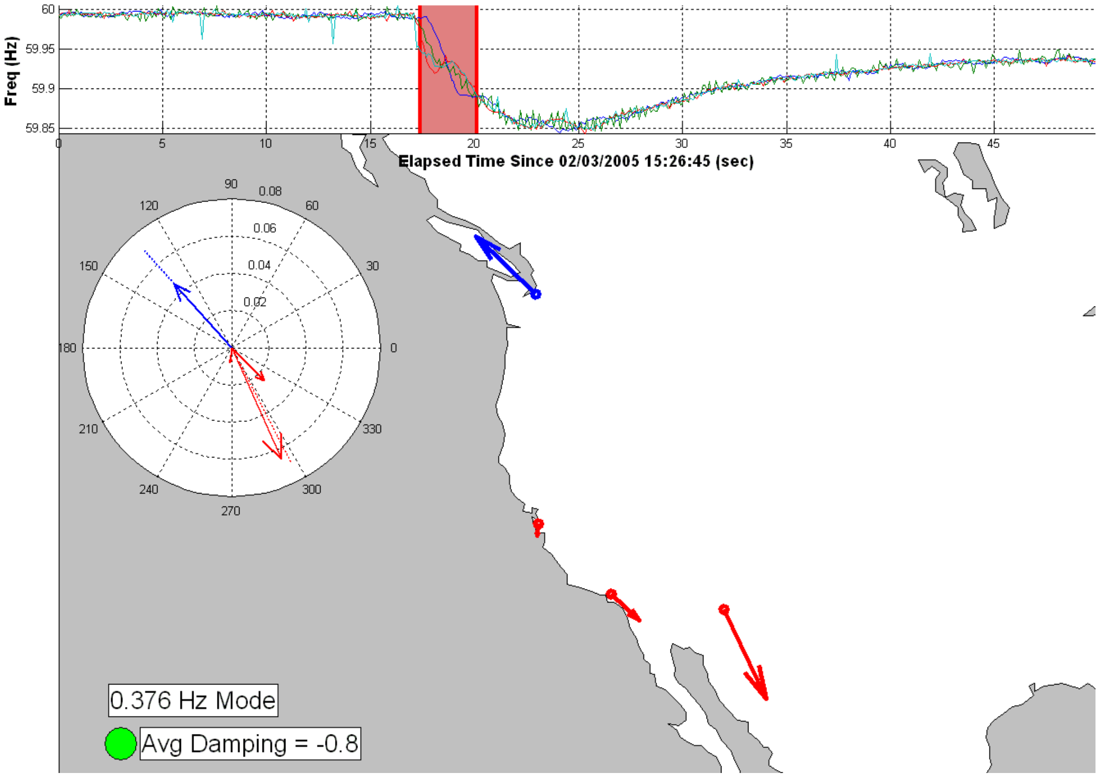


Figure D.11: Continued Separation of Coherent Groups

This high damping continues through Figure D.12 as the oscillation settles out. Now the phasors are smaller than they were before due to damping effects seen in the system. The average damping factor is still very negative with a value of -0.744, as this condition continues the oscillation dies out and the system approaches the post-event steady state operating point. The steady state condition is given by Figure D.13 where all of the phasors have once again achieved a very small value. Throughout these last two figures the two groups remain approximately the same, with the north and south swinging out of phase with each other.

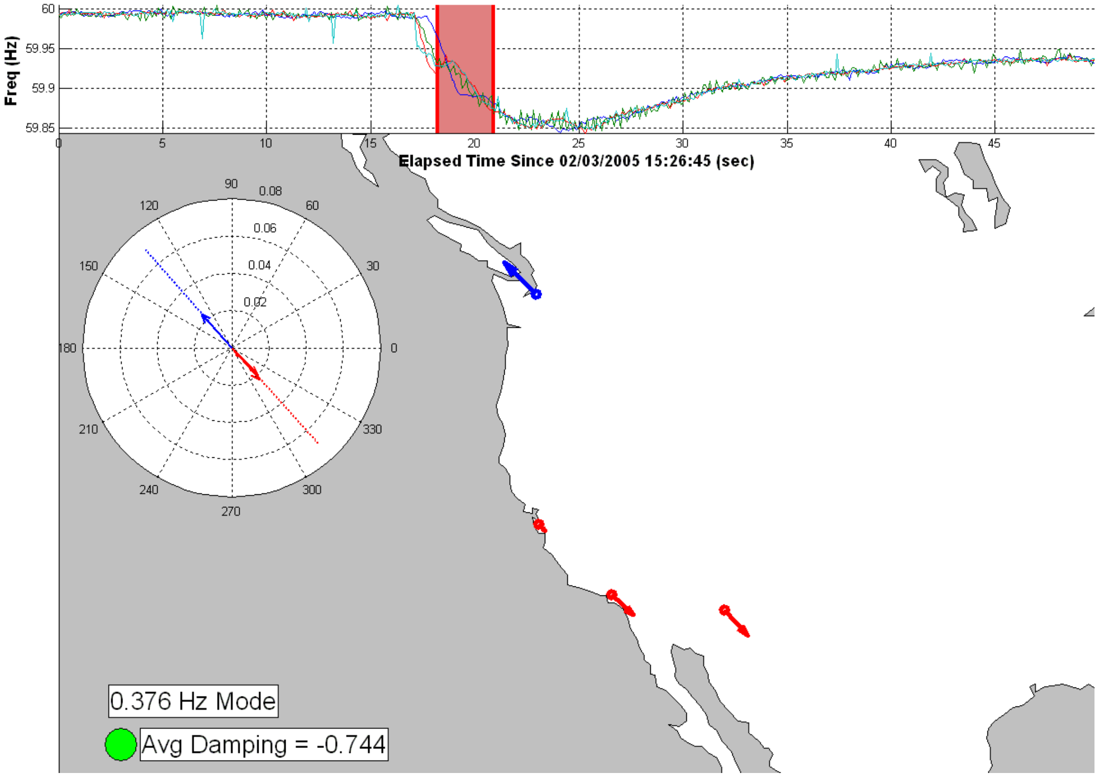


Figure D.12: Interarea Oscillation Dying Out

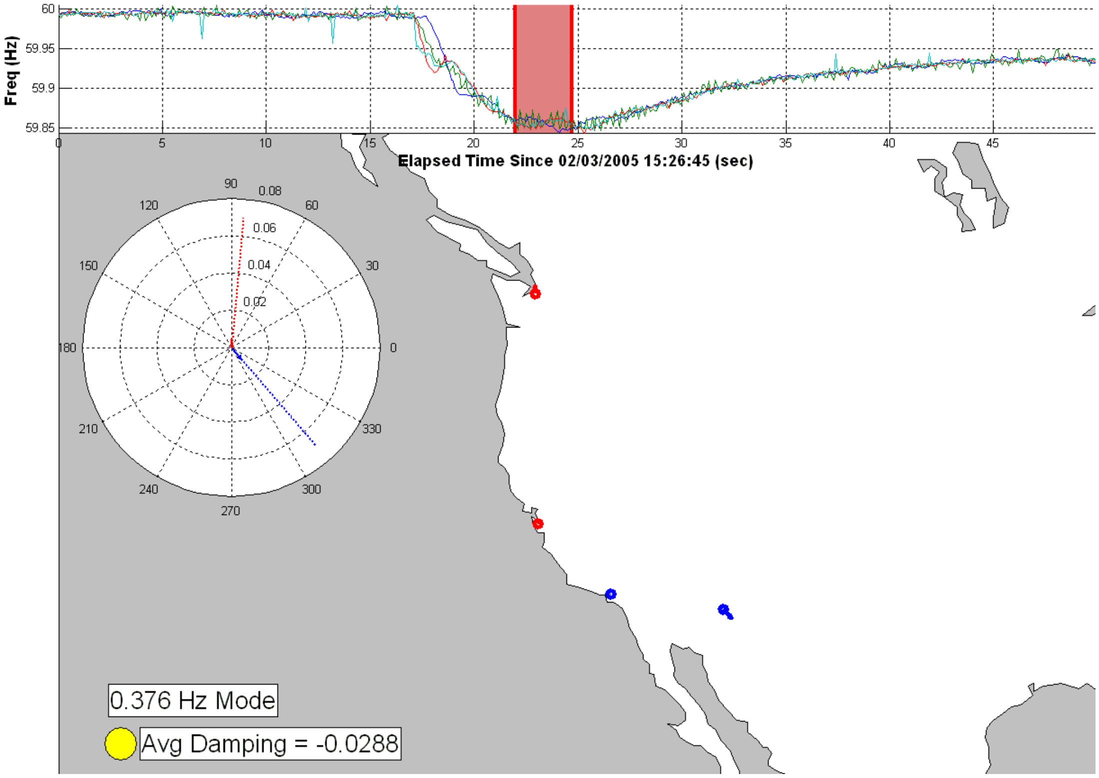


Figure D.13: System Settles to Steady-State