State Estimation and Voltage Security Monitoring Using

Synchronized Phasor Measurements

Reynaldo Francisco Nuqui

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

In

Electrical Engineering

Arun. G. Phadke, Chair

Lee Johnson

Yilu Liu

Lamine Mili

Jaime de la Ree

July 2, 2001

Blacksburg, Virginia

State Estimation and Voltage Security Monitoring Using
Synchronized Phasor Measurements

Reynaldo Francisco Nuqui

(ABSTRACT)

The phasor measurement unit (PMU) is considered to be one of the most important measuring devices in the future of power systems. The distinction comes from its unique ability to provide synchronized phasor measurements of voltages and currents from widely dispersed locations in an electric power grid. The commercialization of the global positioning satellite (GPS) with accuracy of timing pulses in the order of 1 microsecond made possible the commercial production of phasor measurement units.

Simulations and field experiences suggest that PMUs can revolutionize the way power systems are monitored and controlled. However, it is perceived that costs and communication links will affect the number of PMUs to be installed in any power system. Furthermore, defining the appropriate PMU system application is a utility problem that must be resolved. This thesis will address two key issues in any PMU initiative: placement and system applications.

A novel method of PMU placement based on incomplete observability using graph theoretic approach is proposed. The objective is to reduce the required number of PMUs by intentionally creating widely dispersed pockets of unobserved buses in the network. Observable buses enveloped such pockets of unobserved regions thus enabling the interpolation of the unknown voltages. The concept of depth of unobservability is introduced. It is a general measure of the physical distance of unobserved buses from

those known. The effects of depth of unobservability on the number of PMU placements and the errors in the estimation of unobserved buses will be shown.

The extent and location of communication facilities affects the required number and optimal placement of PMUs. The pragmatic problem of restricting PMU placement only on buses with communication facilities is solved using the simulated annealing (SA) algorithm. SA energy functions are developed so as to minimize the deviation of communication-constrained placement from the ideal strategy as determined by the graph theoretic algorithm.

A technique for true real time monitoring of voltage security using synchronized phasor measurements and decision trees is presented as a promising system application. The relationship of widening bus voltage angle separation with network stress is exploited and its connection to voltage security and margin to voltage collapse established. Decision trees utilizing angle difference attributes are utilized to classify the network voltage security status. It will be shown that with judicious PMU placement, the PMU angle measurement is equally a reliable indicator of voltage security class as generator var production.

A method of enhancing the weighted least square state estimator (WLS-SE) with PMU measurements using a non-invasive approach is presented. Here, PMU data is not directly inputted to the WLS estimator measurement set. A separate linear state estimator model utilizing the state estimate from WLS, as well as PMU voltage and current measurement is shown to enhance the state estimate.

Finally, the mathematical model for a streaming state estimation will be presented. The model is especially designed for systems that are not completely observable by PMUs. Basically, it is proposed to estimate the voltages of unobservable buses from the voltages of those observable using interpolation. The interpolation coefficients (or the linear state estimators, LSE) will be calculated from a base case operating point. Then, these coefficients will be periodically updated using their sensitivities to the unobserved bus injections. It is proposed to utilize the state from the traditional WLS estimator to calculate the injections needed to update the coefficients. The resulting hybrid estimator

is capable of producing a streaming state of the power system. Test results show that with the hybrid estimator, a significant improvement in the estimation of unobserved bus voltages as well as power flows on unobserved lines is achieved.

ACKNOWLEDGEMENTS

The successful completion of all research tasks would not have been possible if not for the consistent guidance of my mentor, Dr. Arun G. Phadke. His firm grasps and forte on all diverse areas of power systems ensured a steady stream of ideas that spawns gateways for solving the problems at hand. His personal concern on the well being of my family is truly appreciated. I am forever his student.

Dr. Lamine Mili is to be credited for his enthusiastic discussions on state estimation and voltage stability. I also thank Dr. Jaime de la Ree, Dr. Lee Johnson, and Dr. Yilu Liu for their guidance in this endeavor. Dr. Nouredin Hadjsaid provided interesting discussions on voltage stability.

Special thanks goes to Carolyn Guynn and Glenda Caldwell of the Center for Power Engineering. The competitive but healthy academic environment provided by the following students and former students at the Power Lab always spark my interest to learn more. They are Dr. Aysen Arsoy, Dr. Virgilio Centeno, David Elizondo, Arturo Bretas, Liling Huang, Abdel Khatib Rahman, B. Qiu, Q. Qiu, and Dr. Dong.

I also wish to thank Dr. Nelson Simons and Dr. Aaron Snyder of ABB Power T&D Company. Dr. Francisco L. Viray, Domingo Bulatao (now deceased), Rolando Bacani, and Erlinda de Guzman of the National Power Corporation, Philippines have made available the necessary support to initiate my studies.

I am grateful to my sister Debbie Nuqui for her unwavering support.

A special thanks to my wife, Vernie Nuqui, for allowing this old student to return to school and for taking care of the family. This would not have been possible without her.

For John, Sandra and Kyle

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Figures

List of Tables

List of Symbols

ℓ_x – vector of buses linked to bus x

υ - depth-of-unobservability

S – a PMU placement strategy, which is a set of bus numbers of PMU locations

d – the distance vector, $f: S \rightarrow d$ a mapping of strategy to the distance vector

U_S – the set of unobserved buses associated with a PMU placement strategy S.

(R_U) – the collection of unobserved regions $R_1, R2...RN$ associated with S.

Wo - the vector of buses without communication facilities

Wi - the vector of buses with communication facilities

A – the vector of minimum distance between a bus without communication facilities to a bus with communication facilities

$\Phi(S)$ – a collection of transition strategies of length one from S

Γ_x - the set of buses with distance less than or equal γ from a PMU bus x

$C(S)$ – the cost or energy function used in Simulated Annealing to value a strategy S

T – control parameter in Simulated Annealing

M – maximum number of transitions allowed at each control parameter T

α - the temperature decrement factor for Simulated Annealing

Symbols for Voltage Security Monitoring and State Estimation

DT – decision tree. A collection of hierarchical rules arranged in a binary tree like structure.

T – a tree whose nodes and edges define the structure of a decision tree

Tt – a subtree of tree T emanating from a node t down to the terminal nodes

L – the measurement space. This is a 2-dimensional array containing the values of all attributes a in the collection of M number of sampled cases.

s(a) – the split value corresponding to attribute a in a measurement space M

U(T) – the misclassification rate of a decision tree T

I(t) – the impurity of node t in tree T

$\Delta I(s(a),t)$ – the change in impurity at node t caused by split s on attribute a.

P_{NLj} - real load distribution factors

Q_{NLj} – imaginary load distribution factors

λ - the loading factor

SE – state estimator or state estimation

x – alternatively (V, θ), the state of a power system

U – the vector of unobserved buses

O – the vector of observed buses

Y_{BUS} – the bus admittance matrix

Y_{UU} – bus admittance submatrix of unobserved buses

Y_{OO} – bus admittance submatrix of observed buses

Y_{UO} – mutual bus admittance submatrix between the unobserved and observed buses

Y_L – the load admittance matrix of unobserved buses

H – the matrix of interpolation coefficients of dimension *length(*U*) by length(*O*)*

List of Symbols

$\mathbf{H}i$ – the vector of interpolation coefficients of unobserved bus i

f_{ij} – the apparent power flow from bus i to bus j

$\mathbf{R}$ – the error covariance matrix

z – the vector of measurements in traditional SE

$\mathbf{h}$ – a nonlinear vector function expressing the measurements in terms of the state x

$\mathbf{H}^J$ – the Jacobian matrix of $\mathbf{h}$.

$\mathbf{G}$ – the gain matrix

$\mathbf{W}$ – the inverse of the covariance matrix $\mathbf{R}$

Chapter 1. Introduction

The phasor measurement unit (PMU) is a power system device capable of measuring the synchronized voltage and current phasor in a power system. Synchronicity among phasor measurement units (PMUs) is achieved by same-time sampling of voltage and current waveforms using a common synchronizing signal from the global positioning satellite (GPS). The ability to calculate synchronized phasors makes the PMU one of the most important measuring devices in the future of power system monitoring and control [50].

The technology behind PMUs traced back to the field of computer relaying. In this equally revolutionary field in power system protection, microprocessors technology made possible the direct calculation of the sequence components of phase quantities from which fault detection algorithms were based [51]. The phasor are calculated via Discrete Fourier Transform applied on a moving data window whose width can vary from fraction of a cycle to multiple of a cycle [54]. Equation (1.1) shows how the fundamental frequency component X of the Discrete Fourier transform is calculated from the collection of X_k waveform samples.

$$X = \frac{\sqrt{2}}{N} \sum_{k=1}^{N} X_k \varepsilon^{-j2k\pi/N} \tag{1.1}$$

Synchronization of sampling was achieved using a common timing signal available locally at the substation. Timing signal accuracy in the order of milliseconds suffices for this relaying application. It became clear that the same approach of calculating phasors for computer relaying could be extended to the field of power system monitoring. However the phasor calculations demand greater than the 1-millisecond accuracy. It is only with the opening for commercial use of GPS that phasor measurement unit was finally developed. GPS is capable of providing timing signal of the order of 1 microsecond at any locations around the world. It basically solved the logistical problem of allocating dedicated land based links to distribute timing pulses of the indicated accuracy. Reference [32] presents a detailed analysis of the required synchronization accuracy of several phasor measurement applications.

Figure 1-1 shows a hardware block diagram of a phasor measurement unit. The anti-aliasing filter is used to filter out from the input waveform frequencies above the Nyquist rate. The phase locked oscillator converts the GPS 1 pulse per second into a sequence of high-speed timing pulses used in the waveform sampling. The microprocessor executes the DFT phasor calculations. Finally, the phasor is time-stamped and uploaded to a collection device known as a data concentrator. An IEEE standard format now exists for real time phasor data transmission [33].
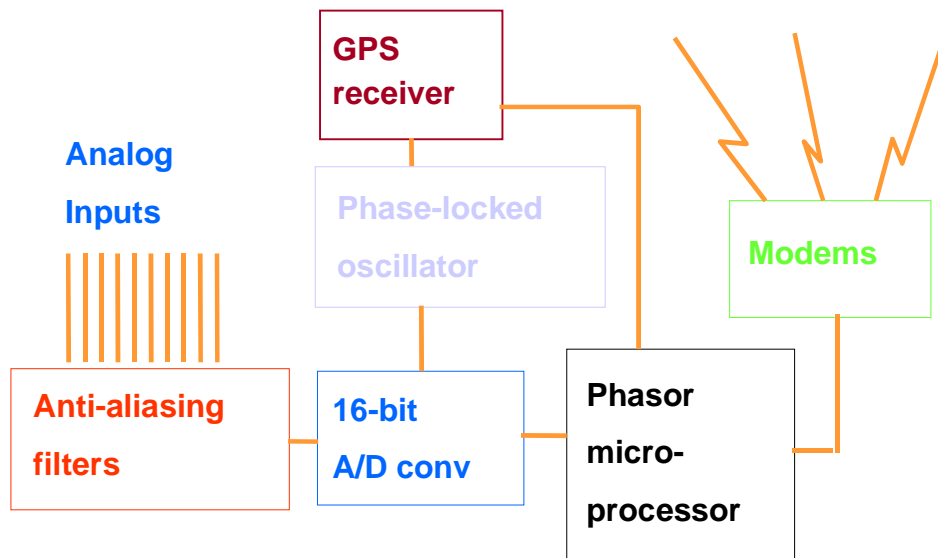
Figure 1-1. Phasor Measurement Unit Hardware Block Diagram

The benefits of synchronized phasor measurements to power system monitoring, operation and control have been well recognized. An EPRI publication [27] provides a thorough discussion of the current and potential PMU applications around the world. PMUs improve the monitoring and control of power systems through accurate, synchronized and direct measurement of the system state. The greatest benefit coming from its unique capability to provide real time synchronized measurements. For example, the positive sequence components of the fundamental frequency bus voltages are used

directly by such advanced control center applications as contingency analysis and on-line load flow. With PMUs the security indicators produced by these advance applications are representative of the true real time status of the power system. Figure 1-2 shows a conceptual picture of a phasor measurement unit system. It must be recognized that the current thrust of utilities is to install fiber optic links among substations. The phasor measurement unit uploads its time stamped phasor data using such medium as dedicated telephone line or through the wide area network (WAN).
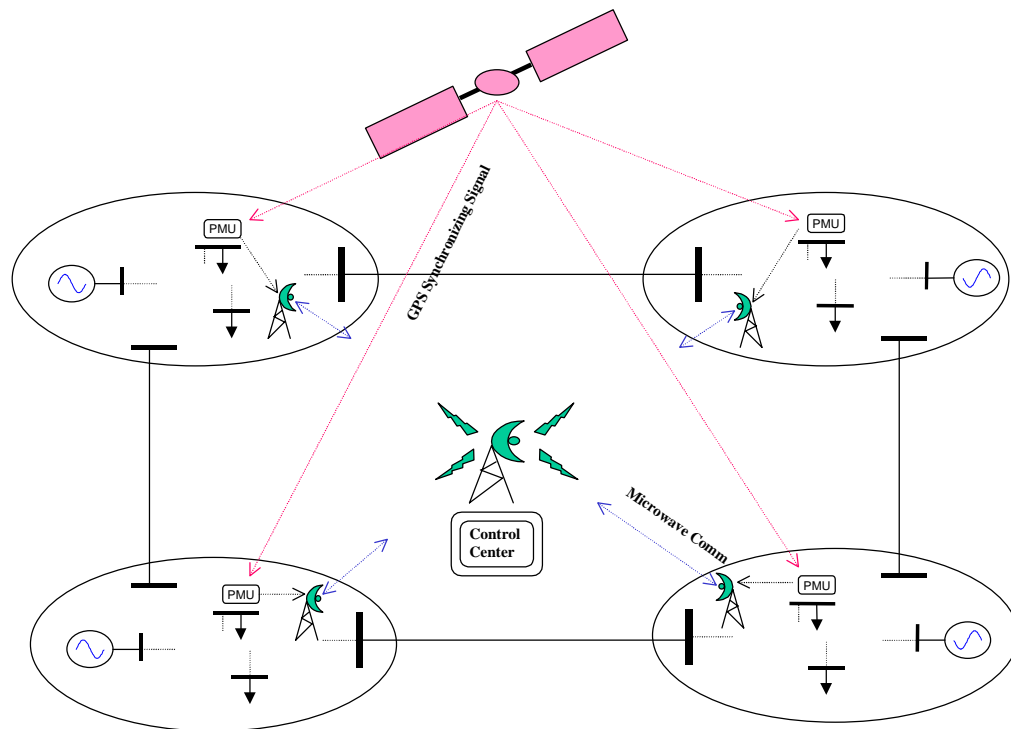
Figure 1-2. Conceptual Diagram of a Synchronized Phasor Measuring System

A system of PMUs must be supported by communication infrastructure of sufficient speed to match the fast streaming PMU measurements. Oftentimes, power systems are not totally equipped with matching communication. As such, any potential move to deploy PMUs must recognize this limitation. It is a possibility that the benefits brought forth by PMUs could justify the installation of their matching communication infrastructure. However, it must be recognized that deployment of PMUs in every bus is a major economic undertaking and alternative placement techniques must consider partial

PMU deployment. Baldwin [5][6] showed that a minimum number of 1/5 to ¼ of the system buses would have to be provided with PMUs to completely observe a network. For large systems, these suggested numbers could still be an overwhelming initial task. Alternate approach of PMU placement is necessary to reduce the numbers further.

The foremost concern among potential users is the application that will justify initial installation of PMUs. As expected from an emerging technology, initial installation of PMUs was made for purposes of gaining experience with the device and its applications. For this purpose, PMUs were deployed mainly on a localized basis. It is our opinion however that the greatest positive impact from the PMU would come from system applications such as state estimation and wide area protection and control. The following survey although by no means exhaustive gives a glimpse of the present and potential applications of synchronized phasor measurements.

A worthwhile albeit simple application is to use a system of PMUs as visual tool to operators. Figure 1-3 for example is a surface plot of the angle measurements among PMUs located in widely dispersed location around a power system. To the control center operator this is very graphic picture of what is happening to the power system in real time. For example, the angle picture represents the general direction of power flows and as well as the areas of sources and sinks. Remote feedback control of excitation systems using PMU measurements has been studied to damp local and inter-area modes of oscillation [63] [49][37]. In this application, frequency and angle measurements from remote locations are utilized directly by a controlled machine's power system stabilizer. In this same application, Snyder et al addressed the problem of input signal delay on the centralized controller using linear matrix inequalities [64]. The reader is referred to a book [56] that delivers a thorough discussion on modes of oscillations. Electricite de France has developed a "Coordinated Defense plan" against loss of synchronism [21]. The scheme makes use of PMU voltage, phase angle and frequency measurements to initiate controlled islanding and load shedding to prevent major cascading events. Similarly, Tokyo Electric Power Company used the difference of PMU phase angle measurements between large generator groups to separate their system and protect it from out-of-step condition [48]. In the area of adaptive system protection, PMUs have been used to determine the system model used by the relay from which the stability of an

evolving swing is predicted [15]. PMUs have also been used to ascertain the accuracy of system dynamic models [11]. Here PMUs measure the dynamic response of the system to staged tripping of transmission lines, which is subsequently compared to computer simulation of the same event. Similarly, PMUs have provided detailed look on known oscillations that were not observed by traditional measurement devices before [17]. Its synchronized high-speed measurement capability has made it favorable for recording system events for after-the-fact reconstruction [57]. A decision tree based voltage security monitoring system using synchronized phasor measurements could be another worthy application [47]. It will be presented in detail in Chapter 4. State estimation is a potential application that has its merits. A PMU-based state estimation ascertains real time monitoring of the state of the power system. It provides a platform for most advanced control center applications. This thesis will deal greatly on PMU-based state estimation in Chapter 5. The use of phase angle measurements has been shown to improve the existing state estimator [62][53].
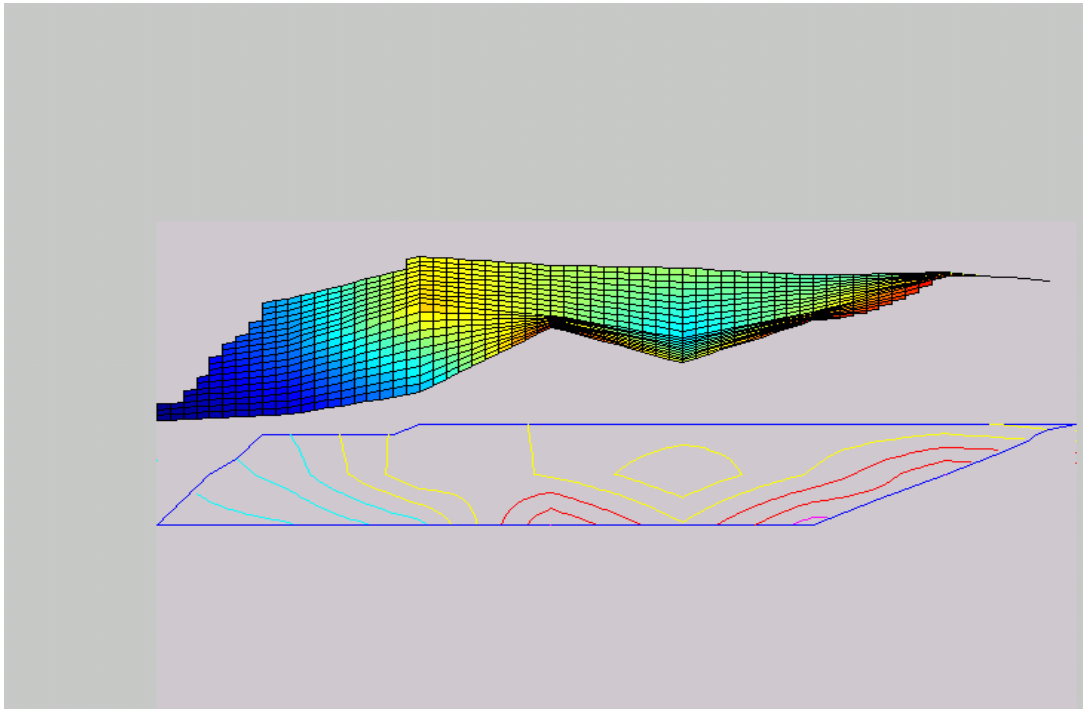
Figure 1-3. Surface Plot of PMU Angle Measurements on a Power System

The major objective of this thesis is the development of models and algorithms for advanced system applications in support of PMU deployment in the industry. The investigated topics in detail are as follows:

1. The development of a PMU placement technique based on incomplete observability. This task is concerned with the sparse deployment of PMUs to satisfy a desired depth of unobservability. The end result is a PMU placement strategy that results in *near* even distribution of unobserved buses in the system. The crux of the overall scheme is the interpolation of the voltages of the unobserved buses from the known buses.

2. The development of a PMU placement with communication constraints. The PMU placement based on incomplete observability was enhanced so that PMUs are deployed only on locations where communication facilities exist. Simulated annealing was successfully utilized to solve this communication constrained PMU placement problem.

3. The real time monitoring of voltage security using synchronized phasor measurement and decision trees. Classification type decision trees complements the high-speed PMU measurements to warn system operators of voltage security risks in real time.

4. The development of models and algorithms for a hybrid type state estimation using phasor measurements and the traditional state estimator. Under the assumption that the system is not fully observable by PMUs, the state from traditional AC state estimator is used to update the interpolators used by the PMU based state estimator. The hybrid estimator is a pragmatic approach of utilizing a reduced number of PMUs for state estimation with a functioning traditional AC state estimator. The hybrid estimator is capable of providing a streaming state of the power system with speed limited only by the quality of communication available to the PMUs.

This thesis is composed of five main chapters including this introduction. The chapters are presented in such a way that each of the four main objectives presented above is contained in one chapter. Chapter 6 summarizes all the research task of this thesis and recommends directions for future research.

Chapter 2. Phasor Measurement Unit Placement For Incomplete Observability

2.1 Introduction

PMU placement in each substation allows for direct measurement of the state of the network. However, a ubiquitous placement of PMUs is rarely conceivable due to cost or non-existence of communication facilities in some substations. Nonetheless, the ability of PMUs to measure line current phasors allows the calculation of the voltage at the other end of the line using Ohm's Law. Baldwin, Mili, et al. [5] showed that optimal placement of PMUs requires only 1/5 to ¼ of the number of network buses to ensure observability.

It is possible to reduce the numbers even further if PMUs are placed for incomplete observability. In this approach, PMUs are placed sparingly in such a way as to allow unobserved buses to exist in the system. The technique is to place PMUs so that in the resulting system the topological distance of unobserved buses from those whose voltages are known is not too great. The crux of this overall scheme is the interpolation of any unobserved bus voltage from the voltages of its neighbors.

This chapter is divided into four main sections. The first section introduces one of the fundamental contributions of this thesis – the depth of unobservability. This concept sparks the motivation and subsequent modeling of a placement algorithm for incomplete observability. Here we used a tree search technique to find the optimal placements of PMUs satisfying a desired depth of unobservability. Placement results are presented for three IEEE test systems and two utility test systems. We present a model for phased installation wherein PMUs are installed in batches through time as the system migrates to full PMU observability. The phased installation approach recognizes the economical constraints of installing significant number of PMUs in any utility system. Another real world constraint in PMU placements is limited communication - a separate problem by itself and is deferred for Chapter 3.

2.2 Concept of Depth of Unobservability

Unobservability within the context of this thesis refers a network condition wherein in lieu of meter or PMU placement a subset of the system bus voltages cannot be directly calculated from the known measurements.

We introduce the concept of depth of unobservability – *one of the fundamental contributions of this thesis.* Figures 2-1 and 2-2 illustrate this concept. In Figure 2-1 the PMUs at buses B and F directly measure the voltages V_B and V_F respectively. The voltage at bus C is calculated using the voltage at bus B and the PMU-1 line current measurement for branch AB. The voltage at bus E is also calculated in a similar fashion. We define buses C and E as *calculated* buses. The voltage of bus X cannot be determined from the available measurements however (since the injection at either bus C or bus E is not observed). Bus X is defined to be *depth of one unobservable* bus because it is bounded by two observed (calculated) buses. Furthermore, a *depth of one unobservability* condition exists for that section of the power system in Figure 2-1. A *depth of one unobservability placement* refers to the process of placing PMUs that *strives* to create depth of one unobservable buses in the system.

Similarly, Figure 2-2 characterizes a *depth-of-two unobservability* condition. Buses R and U are directly observed by the PMUs, while voltages at buses S and T are calculated from the PMU line current measurements. Buses Y and Z are *depth of two* unobserved buses. A *depth of two unobservability* condition exists when two observed buses bound two adjoining unobserved buses. It is important to realize that such condition exists if we traverse the path defined by the bus sequence R-S-Y-Z-T-U.

The concept of depth of unobservability and the aforementioned definitions are extendable for higher depths. This innovative concept will drive the PMU placement algorithm in Section 2.3. Imposing a depth of unobservability ensures that PMUs are well distributed throughout the power system and that the distances of unobserved buses from those observed is kept at a minimum.
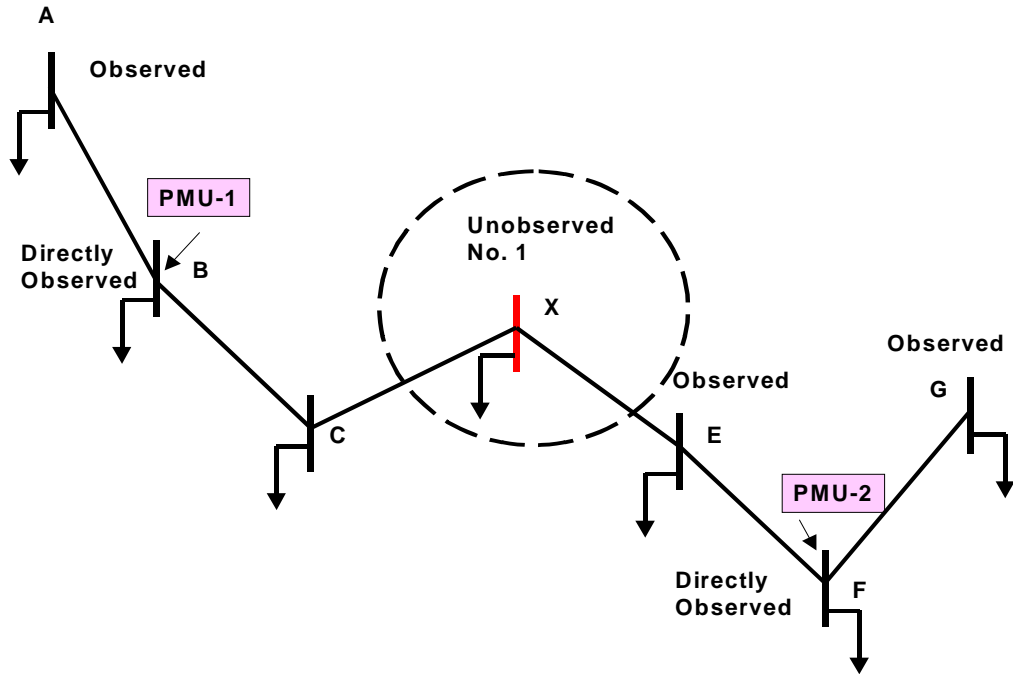
Figure 2-1 Depth of One Unobservability Illustrated
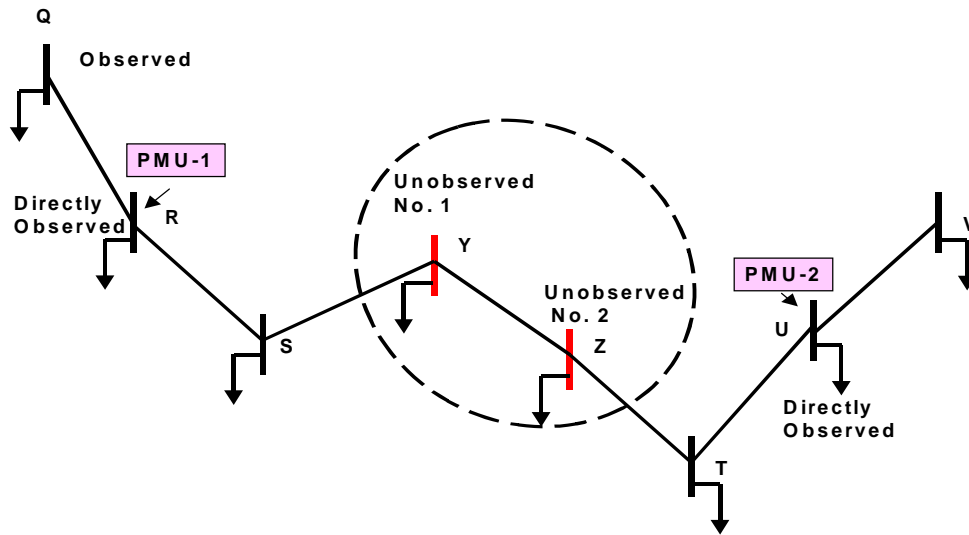
Figure 2-2. Depth of Two Unobservability Illustrated

For any given depth of unobservability condition the voltages of unobserved buses can be estimated from the known voltages. Consequently, the vector of directly measured and calculated voltages augmented by the estimated voltages completes the state of the system. A streaming type of state exists with rate as fast as the speed of the

PMU measurements.   The rest of this section is focused on mathematical formulation on how to estimate the unknown voltages.

Consider once more Figure 2-1. The voltage E_X of the unobserved bus X can be expressed in terms of the calculated voltages E_C and E_E. Applying Kirchoff's Current Law (KCL) on unobserved bus X yields

$$0 = V_X y_{XX} + (V_X - V_C) y_{CX} + (V_X - V_E) y_{EX} \tag{2.1}$$

y_{XX} here refers to the complex admittance of the injection at unobserved bus x; equation (2.2) expresses y_{XX} in terms of bus real and complex power injection and bus voltage

$$y_{XX} = \frac{I_U}{V_U} = \frac{S_U^*}{|V_U|^2} \tag{2.2}$$

y_{CX} and y_{EX} are complex admittances of the lines linking bus X to buses C and E. From equation (2.1) V_X can be expressed in terms of V_C and V_E as shown in equation (2.3)

$$V_X = \frac{y_{CX}}{y_{XX} + y_{CX} + y_{EX}} V_C + \frac{y_{EX}}{y_{XX} + y_{CX} + y_{EX}} V_E \tag{2.3}$$

Alternatively,

$$V_X = a_{XC} V_C + a_{XE} V_E \tag{2.4}$$

where

$$a_{XC} = \frac{y_{CX}}{y_{XX} + y_{CX} + y_{EX}}$$
$$a_{XE} = \frac{y_{EX}}{y_{XX} + y_{CX} + y_{EX}} \tag{2.5}$$

It can be concluded from equation (2.4) that the voltage of unobserved bus X can be expressed in terms of the known voltages of the buses linked to it. The same equation implies that this relationship is linear. The terms a_{XC} and a_{XE} are the interpolation coefficients that weighs the contribution of V_C and V_E respectively to V_X. Equation (2.5) shows that these interpolation coefficients are functions of the equivalent admittance of

the load injection at the unobserved bus X and admittances of the lines linking X to the buses with known voltages.

Assuming V_C and V_E are accurately measured, the error in the estimation of V_X in equation (2.4) can only be attributed to y_{XX}, since it dynamically changes with operating conditions (that is, changes in load, generation, etc.). This error is the result of holding y_{XX} to some reference value y_{XX}^{ref} within a predefined operating condition that includes for example, certain time of the day or range of system load.

Similarly, for a depth-of-2 unobservability condition in Figure 2-2 the voltages V_Y and V_Z can be expressed in terms of known voltages V_S and V_T by applying KCL to buses Y and Z.

$$0 = V_Y y_{YY} + (V_Y - V_S) y_{SY} + (V_Y - V_Z) y_{ZY}$$
$$0 = V_Z y_{ZZ} + (V_Z - V_T) y_{TZ} + (V_Z - V_Y) y_{ZY}$$

(2.6)

Where y_{YY} and y_{ZZ} are the complex load admittances of the unobserved buses Y and Z. y_{SY} and y_{TZ} are complex line admittances from Y or Z to the buses with known voltages S and T. y_{ZY} is the complex admittance linking the unobserved buses. Solving (2.6) for V_Y and V_Z yields.

$$V_Y = \frac{y_{SY}}{Y_Y - \dfrac{y_{ZY}^2}{Y_Z}} V_S + \frac{y_{ZY} y_{TZ}}{\left(Y_Y - \dfrac{y_{ZY}^2}{Y_Z}\right) Y_Z} V_T$$

$$V_Z = \frac{y_{ZY} y_{TZ}}{\left(Y_Z - \dfrac{y_{ZY}^2}{Y_Y}\right) Y_Y} V_S + \frac{y_{TZ}}{Y_Z - \dfrac{y_{ZY}^2}{Y_Y}} V_T$$

(2.7)

where

$$Y_Y = y_{SY} + y_{ZY} + y_{YY}$$
$$Y_Z = y_{TZ} + y_{ZY} + y_{ZZ}$$

(2.8)

Similarly, we can express (2.7) in the more concise form as follows

$$V_Y = a_{YS} V_S + a_{YT} V_T$$
$$V_Z = a_{ZS} V_S + a_{ZT} V_T$$

(2.9)

where

$$a_{YS} = \frac{y_{SY}}{Y_Y - \dfrac{y_{ZY}^2}{Y_Z}} \; ; \; a_{YT} = \frac{y_{ZY}\, y_{TZ}}{\left(Y_Y - \dfrac{y_{ZY}^2}{Y_Z}\right) Y_Z}$$

$$a_{ZS} = \frac{y_{ZY}\, y_{TZ}}{\left(Y_Z - \dfrac{y_{ZY}^2}{Y_Y}\right) Y_Y} \; ; \; a_{ZT} = \frac{y_{TZ}}{Y_Z - \dfrac{y_{ZY}^2}{Y_Y}}$$

(2.10)

Equation (2.9) shows that for a depth-of-2 unobservability a linear relationship also exists between the unobserved voltages V_Y and V_Z and the known voltages V_S and V_T. The interpolation coefficients are also functions of complex load admittances and line admittances (2.10). Any error in the interpolation equation (2.9) is solely attributed to the error in the estimate of the complex line admittances y_{YY} and y_{ZZ}. Note however that both y_{YY} and y_{ZZ} contributes to the error on each of the voltages as seen in equation (2.10).

2.3 PMU Placement For Incomplete Observability

Placement for incomplete observability refers to a method of PMU placement that intentionally creates unobserved buses with a desired depth of unobservability. PMUs placed in this way obviously results in lesser number to cover the subject power system. The proof to this assertion follows.

For an N bus radial system, $Pc = ceil(N/3)$ PMUs are required to observe the N bus voltages. This is due to one PMU observing three buses: one by direct measurement and the other two by calculation using the line current measurements. For the same N radial bus system, the upper bound on the number of PMUs required to satisfy a depth of unobservability υ is

$$P_U = ceil\left(\frac{N}{3 + \upsilon/2}\right)$$

(2.11)

Equation (2.12) expresses the approximate upper bound on the PMU number reduction P_C-P_U as a fraction of the P_C, the required number for complete observability.

$$P_C - P_U \approx \frac{N}{3} - \frac{N}{(3+\upsilon/2)} \approx P_C\left(\frac{1}{6/\upsilon+1}\right) \tag{2.12}$$

For meshed systems a PMU generally cover more than three buses. It is to be expected in this case that the required number for either complete or incomplete observability is less than *N/3* or P_U respectively. An analytic expression for the expected reduction in the number of PMUs is graph specific and cannot be determined. The expected reduction can only be done through numerical experimentation. However we expect that equation (2.12) also approximate the expected reduction in the number of PMUs.

<u>Motivation</u>

A graphic illustration of the proposed PMU placement technique applied to a hypothetical 12-bus system is illustrated in Figure 2-3. Here PMUs are placed sequentially in the system with the tree branches acting as paths or direction for the next candidate placement. Presented are 3 snapshots of the PMU placement process each time a new PMU is installed. The objective is a depth of one unobservability placement. Note that the network is "tree" by structure. A logical first PMU placement should be one bus away from a terminal bus. This makes sense since Ohm's Law can calculate the terminal bus anyway. We arbitrarily placed PMU-1 at bus 1 (see Figure 2-3(A)). To create a depth of one unobserved bus the next candidate placement should be 4 buses away from PMU-1 along an arbitrarily chosen path. Here the search for the next PMU placement traversed the path depicted by the bus sequence 1-4-5-6-7. PMU-2 is placed at bus 7 (Figure 2-3(B)) wherein it will likewise observe terminal bus 8. Bus 5 is now a depth of one unobservable bus. At this point we backtrack and search for another path not yet traversed and this brings us all the way back to bus 4. The search now traverses the sequence of buses 4-9-10-11-12 subsequently placing PMU-3 at bus 11 that creates the other depth of one bus – bus 9 (see Figure 2-3(C)). At this point all buses have been searched and the procedure terminates with the indicated PMU placement. Power systems are typically meshed by topology. The practical implementation of the illustrated placement technique requires the generation and placement search on a large number of

spanning trees of the power system graph. The optimal placement is taken from the tree that yields the minimum number of PMUs.
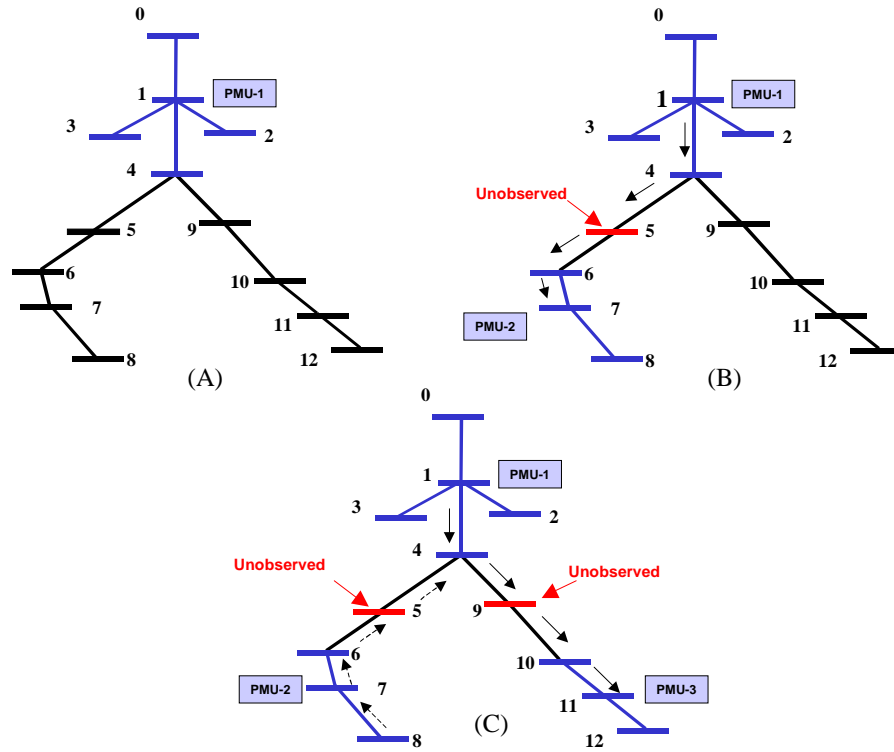
Figure 2-3. Placement for Incomplete Observability Illustrated

Some graph theoretic terminology will be interjected at this point to prepare for the development of the PMU placement algorithm. For our purpose "nodes or buses" are used interchangeably, as are "branches, lines, or edges." We define G(N,E) as the power system graph with N number of buses and E number of lines. A spanning tree T(N,N-1) of the power system graph is a sub-graph that is incident to all nodes of the parent graph. It has N-1 branches, and has no loops or cycles. A branch is expressed by the ordered pair (x,y) with the assumed direction x→y, that is, y is the head of the arrow and x is the tail. Alternatively, a branch can also be identified as an encircled number. The connectivity (or structure) of the parent graph will be defined by the links array L whose column Lj contains the set of buses directly linked to bus *j*. An array subset of L denoted by L^t defines the connectivity of a spanning tree *t* of G(N,E). The degree of a node is the number of nodes linked to it. A leaf node is a node of degree one, alternatively defined as a terminal bus.

<u>Tree Building</u>

The proposed PMU placement works on spanning trees of the power system graph. Thus, a tree-building algorithm is core requirement. Several techniques exist for building spanning trees, but speed is a major consideration especially when we deal with large power system graphs. Even a modest-sized network has large number of trees. If we have n lines and b buses, then the number of unique spanning trees is the combination of n lines taken b-1 at a time, or

$$\binom{n}{b-1} \tag{2.13}$$

For large systems, generating the entire set of unique trees and performing PMU placement on each can take a very long time to finish. The only recourse is to perform a PMU placement on a subset of the total trees. This can be done using a Monte-Carlo type of tree generation. For small systems, tree generation using the network incidence matrix **A** (the Hale algorithm [30]) proved to be best. The graph should be directed. The incidence matrix A is actually the coefficient matrix of Kirchoff's current equations. It is of order nxb, where b is the number of branches in the graph. Its elements $A = [a_{ij}]$ are

$a_{ij} = 1$ if branch e_j is incident at node i and directed away from node i,

$a_{ij} = -1$ if branch e_j is incident at node i and directed toward node i,

$a_{ij} = 0$ if branch e_j is not incident at node i.

First, a sub-graph is selected. It is codified as an ordered list of branches $\{e_1 e_2 e_3 \ldots e_{n-1}\}$. Then, the branches of this graph are successively short-circuited. Short-circuiting a branch e_j will make the associated column *j* of A zero. If during this graph operation, an additional column k becomes zero, the operation is halted. The sub-graph contains a circuit (loop) and therefore is not a tree. Otherwise, the operation will terminate with all columns becoming non-zero. This is now a spanning tree. The procedure is repeated for an entirely unique sub-graph of the network. The process is terminated when all candidate trees whose number defined by (2.14), is exhausted. For large systems however, Hale's algorithm experiences difficulty in building the first tree.

This is attributed to the following reason: that the first N-1 branches in the ordered list of branches does not contain all the unique N buses of the network at all, in which case the algorithm updating the current ordered list of branches with another. Replacing the last branch in the list by the next higher numbered branch does this. If the branch that is creating the problem is in the middle of the list, then computational complexity results.

The solution therefore is to generate an initial tree for the Hale algorithm and allow it to generate the succeeding trees. The technique is quite simple. Assume that we have a bin containing the branches of the initial tree. Initially, the bin is empty. From the set of *free branches* of the graph, we transfer one branch at a time to the bin. Obviously, we test if both nodes of this branch already exist in the bin in which case the branch is discarded (since it creates a loop among the branches). Otherwise, it becomes part of the bin. The process terminates when a total of N-1 branches are transferred to the bin.

Algorithm for PMU Placement for Incomplete Observability: The TREE Search

With the foregoing discussion and Figure 2-3 as the motivation, the PMU placement for incomplete observability is now developed. Basically, this is a tree search technique wherein we move from bus to bus in the spanning tree to locate the next logical placement for a PMU. We terminate the search when all buses have been visited. The algorithm as developed is based on graph theoretic techniques [45] and set notations and operations [55].

Let S be the PMU placement cover for a spanning tree containing the complete list of PMU buses. Since S is built incrementally, let the current partial list of PMUs be S^K with elements S_i, $i=1..K$, K being the size of the placement set S^K and the also the *instance* when the placement set is incremented by a new PMU. It is essential to keep track of the set of buses that have been part of the set queried for possible placement. Define a vector A^J whose elements at any *jth* instance are the set of buses that have been visited in the tree search, which we define as *tagged buses*. J is the counter as we move from bus to bus. Define its complement, Ω^i, $i=N-j$ as the set of free buses, that is, the set of buses not yet visited. Obviously, the search for PMU sites is completed when Ω^i is

null, $\underline{\Omega}^i = \varnothing$. Parenting of nodes must be established to institute directionality while visiting each node of the tree. Parenting becomes particularly important when we need to backtrack and search for a free bus. Define a parent vector P whose element P_j is the *parent node* of *child node* j, that is, bus-j is visited right after bus Pj.

Now define a distance vector d^K whose elements d_j's at any instance K are defined by (2.14).

$$d_j^{\ K} = \begin{cases} 0, \ j \in \mathbf{S}^K \\ 1, \ j \in \mathbf{C}^K \\ \gamma_j, \ j \in \mathbf{U}^K \end{cases} \tag{2.14}$$

Where at instance K, S^K is the set of PMU buses, C^K is the set of calculated buses, and U^K is the set of unobserved buses with distances $\gamma_{\{j \in U\}}$ defined as the maximum number of buses separating an unobserved bus j from the nearest PMU bus.

We can now pose the following <u>PMU placement rule</u>: *given a desired depth of unobservability* υ*, the next candidate PMU placement node p must be of distance* $d_p=\upsilon+3$. This rule can be proven from Figure 2-1. Assuming that PMU-1 is placed at bus C at instance $K=1$, then the distance of PMU-2 at the same instance $K=1$ is $d_F^1=(\upsilon=1)+3=4$. The same proof can be applied to Figure 2-2 wherein $d_U^1=(\upsilon=2)+3=5$. Obviously, after a new PMU is added to the list the PMU placement set and instance K are updated incrementally to $S^{K+1}, K=K+1$. The distance vector (2.14) must also be updated.

Some of the elements of d^K can be determined by an algorithm that maps the PMU placement at instance K to the distances of the buses $j, j=1...N$. Assigning the elements $dj=0$ is straightforward since $dj=0 \ \forall \ j \in S^K$. A convenient way of assigning the elements $dj=1$ is to find the set of buses C^K incident to the set of PMU buses S^K. However, an exact way of determining the set C^K is through the observability algorithm (see [10]) shown in Figure 2-4. Applied to the power system graph $G(N,E)$, the observability algorithm is capable of identifying calculated buses not incident to PMU buses using the list of buses without active injections.

---

1. If a node v has a PMU, then all buses incident to v are observed. Formally, if $v \in S^K$, then $L_v \in C^K$.

2. If a node v is observed, and all nodes linked to v are observed, save one, then all nodes linked to v are observed. Formally, if $v \in C^K$ and $|L_v \cap U^K| \leq 1$, then $L_v \subseteq C^K$.

Figure 2-4. Observability Algorithm

Given the set S^K and C^K from the observability algorithm, we can proceed to determine the distances of a select set of buses Γ^K along a partial tree currently being searched. If we assume the tree in Figure 2-1 as a part of a much bigger spanning tree, and the current search is being conducted along the partial tree defined by the sequence of buses $\Gamma^K = \{C\text{-}X\text{-}E\text{-}F\}$, then distances $d^K_{[C\,X\,E\,F]} = [1\ 2\ 3\ 4]$. Note that these distances are updated incrementally as each bus is visited, that is, $d^K_E = d^K_X + 1$, $d^K_F = d^K_E + 1$, and so on. The next PMU is placed at bus F since its distance $d^K_F = \upsilon + 3 = 4$, hence $S^{K+1} = \{B, F\}$. Running the observability algorithm will update the distances of this partial tree to $d^{K+1}_{[C\,X\,E\,F]} = [1\ 2\ 1\ 0]$.

The tree search backtracks when it encounters a terminal bus τ. Two types of terminal buses exist: the first one is a real terminal bus from the parent graph, the other one is a terminal bus of the spanning tree only. In the former type, if τ is unobserved albeit its distance is less than $\upsilon + 3$, that is, $1 < d\tau < \upsilon + 3$, then a PMU will be placed, but one bus away from τ. This strategy allows more coverage for the new placement while still making τ observable (it is a calculated bus). If τ is of the latter type, more involved tests need to be conducted to see if a new placement is necessary. Let us consider a depth of one placement. If $d^K\tau = 2$ but there exist at least one $j \in L\tau$ but $j \neq P\tau$ with distance $d^K_j = 1$, then a depth of one unobservability structure exists and no new PMU placement is necessary at τ. This rule is exclusive in that if $d^K\tau > 2$ a new PMU will be placed at τ. Similar reasoning is applied for higher depths of unobservability.

Figure 2-5 presents a flow chart of the PMU placement technique. An outer loop that iterates on a subset of spanning trees is added. The loop is between process box 1 and decision box 5. The objective is to find the spanning tree that yields the minimum number of PMU placements. Note that even modest size power systems yield very large number of spanning trees. The PMU placement algorithm can be made to run on any number of trees, time permitting. The discussed modified Hale algorithm is used in this paper.

The search for the optimal PMU placement strategy starts by inputting the structure of the graph. Typically this is as simple as the line p-q list from loadflow. The user inputs the desired depth of unobservability υ and the link array of the parent graph is established.

Process box $\boxed{1}$ involves the generation of a spanning tree based on [30] and establishing the structure of the spanning tree. Process box $\boxed{2}$ initiates the first PMU placement and initializes the set of free buses and tagged buses. Process box $\boxed{3}$ maps the existing PMU placement set with the distance vector. Here the observability algorithm is invoked. Process box $\boxed{4}$ is an involved process that chooses the next bus to visit in the tree search. The basic strategy is to move to any arbitrary free bus b^{I+1} linked to the existing bus b^I , that is, choose b^{I+1}=j, where j$\in$ **tL**$_{bI}$ and j$\in \mathbf{\Omega^I}$. If a terminal bus j=τ is visited, the process backtracks and searches for a free bus in a backward process along the direction child_node $\rightarrow$ parent_node $\rightarrow$ parent_node$\rightarrow$…etc. If this backtracking moves all the way back to the root node, then $\mathbf{\Omega^I}$=$\varnothing$ and the search for this spanning tree is finished (see decision box 7).

Decision box 1 tests if the PMU placement rule is satisfied. If yes, a new PMU is placed at b^I and processes $\boxed{6}$ and $\boxed{7}$ updates the PMU placement set S^{K+1}, $K=K+1$, invokes the observability algorithm and recalculates the distance vector with this updated placement set. If no, a check is made if this is a terminal bus (decision box 2). If this is a terminal bus, another test (decision box 3) is made to determine if a PMU placement is warranted.

Although the flowchart illustrates a one-to-one correspondence between a PMU placement set and a spanning tree, in reality, initiating the search from another bus

location can generate additional placement sets. That minimum sized placement set is associated with this spanning tree.   The optimal placement strategy is taken as the smallest sized placement from the stored collection of strategies.
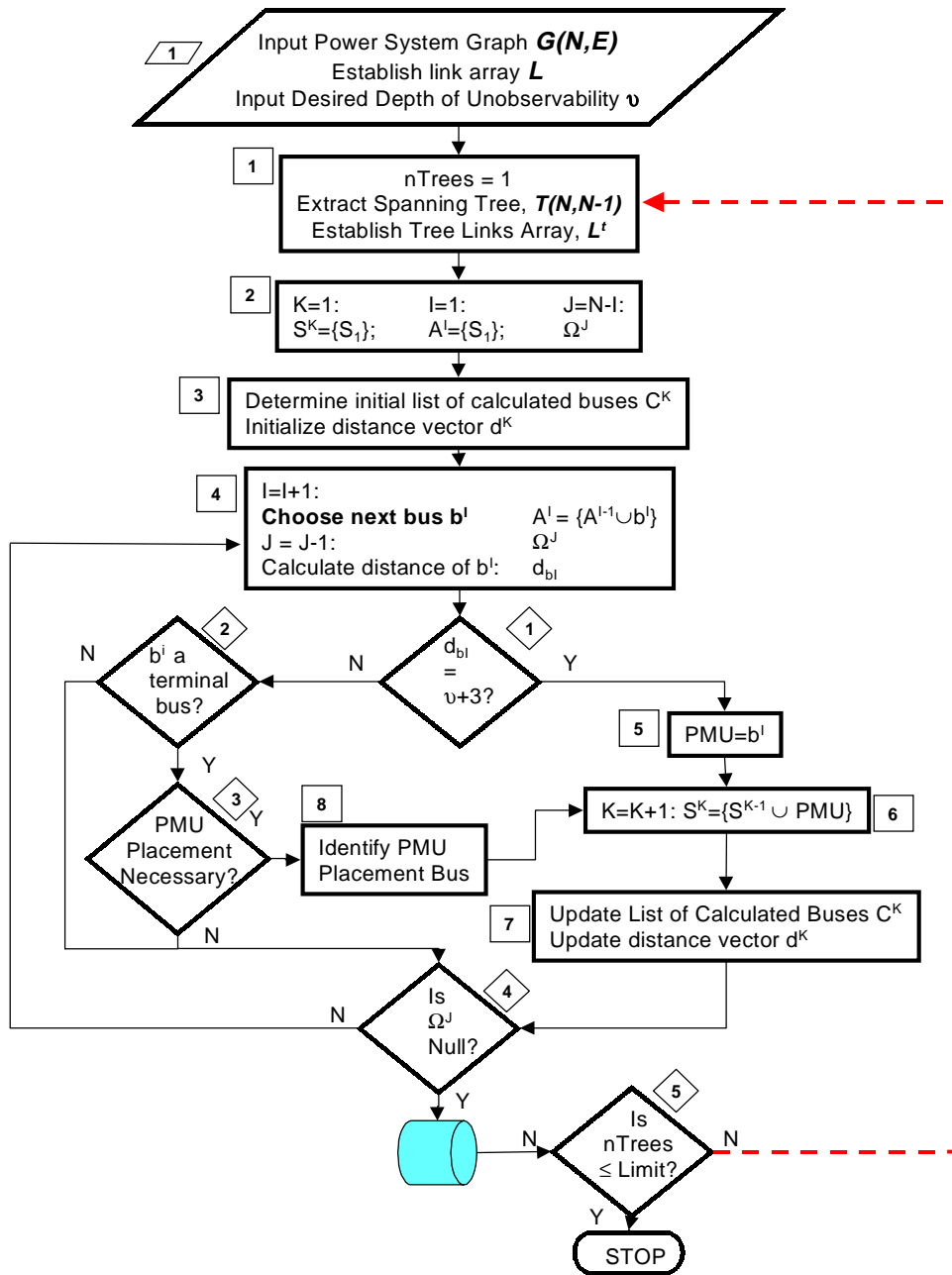
Figure 2-5. Flow Chart of PMU Placement for Incomplete Observability

An example for depth of one unobservability placement is now illustrated for the IEEE 14-bus test system. Figure 2-6 shows a spanning tree of the subject test system; co-trees or branches that does not form part of the spanning tree are illustrated as dotted lines. Lines are conveniently numbered (and encircled) so that they represent the chosen route taken during the tree search. An asterisk '*' next to the line number signifies backtracking after the terminal bus is visited. Assume that bus-12 is the root node. At instance $K=1$, the initial placement is Bus-6, that is, $S^1=\{6\}$. Invoking the observability algorithm results in the list of calculated buses $C^1=\{5\ 11\ 12\ 13\}$ and the distance vector $d^1(5\ 6\ 11\ 12\ 13) = \{1\ 0\ 1\ 1\ 1\}$. Initially (i=1) the set of tagged buses is $A^1=\{6\}$ and free buses $\Omega^1=\{1\ 2\ 3\ 4\ 5\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\}$. Now, choose Bus-5 as the next bus to visit. Its distance $d^1\{5\}=d^1\{6\}+1=1$. Since $d^1\{5\} < (4=\mu+3)$, then we proceed to bus-1 with distance $d^1\{1\}= d^1\{5\}+1=2$. Again, this is not a sufficient condition for a PMU placement. In fact, it's only when we reach bus-4 when a PMU is placed since $d^1\{4\}=d^1\{3\}+1=4$. At this instance K=2, we have 2 PMUs installed $S^2=\{6\ 4\}$ and a list of calculated buses $C^2=\{2\ 3\ 5\ 7\ 8\ 9\ 11\ 12\ 13\}$. Although bus-8 is physically located two buses away from PMU bus-4, it is observable via second rule of the observability algorithm. The search proceeds to bus-9, backtracks and goes to bus-7 and then bus-8. At this point, with the current PMU placement set, 3 unobserved depth of one buses have been identified; buses 1, 10 and 14. However, the algorithm goes on to search for another routes since we still have free buses in our list. We backtrack all the way to bus 6, from which the next forward move is to buses 11 and $\tau=10$ (a terminal bus). The distance of terminal bus 10 is $d^{K=2}\tau=2$, but no PMU is placed here since $d^{K=2}{}_9=1$. Again backtracking leads us back to bus-6 from which we move forward to bus 13 and bus $\tau=14$. This bus is in the same situation as bus 10, that is, it is linked to bus 9 whose distance $d^{K=2}{}_9=1$. Thus no PMU is placed here. Finally, a last backtracking move leads to the root node (Bus 12). At this point, the set of free buses is null, $\Omega = \varnothing$, so the search terminates with 2 PMU placements.
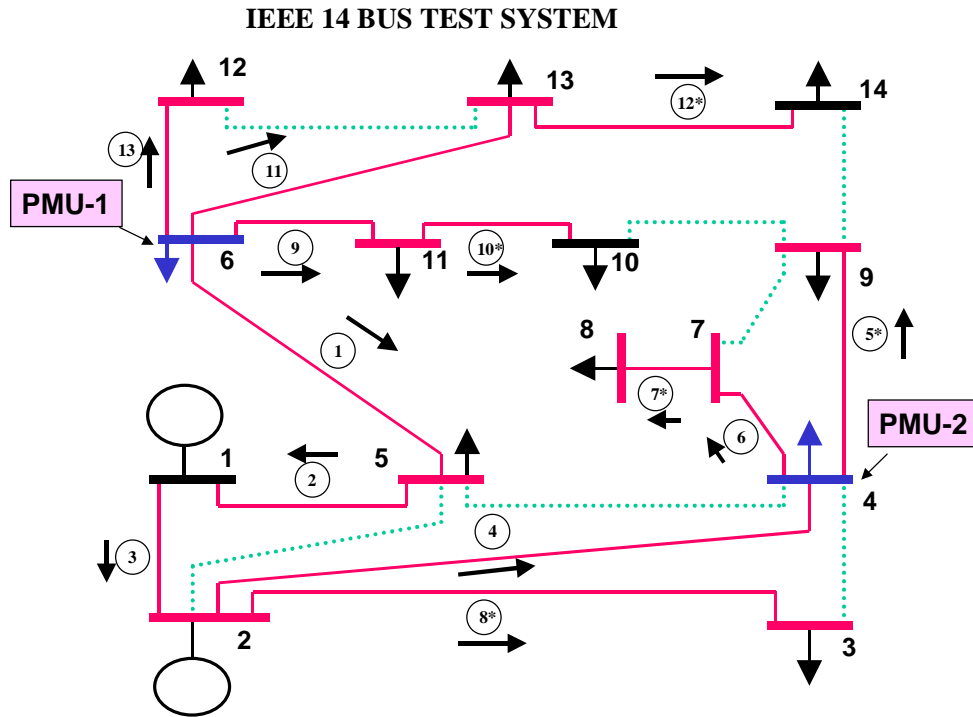
IEEE 14 BUS TEST SYSTEM

Figure 2-6. PMU Placement Illustrated for IEEE 14 Bus Test System

2.4   Numerical Results

A total of 60,000 spanning trees were generated for this exercise using the method described in Section 2.3. The graph theoretic placement technique shown in Figure 2-5 was codified in C.

Figure 2-7 shows a PMU placement strategy for depth of one unobservability of the IEEE 57 bus test system. Nine PMUs are installed. A total of 9 unobserved depth of one buses (encircled) exist. It may seem that the bus pairs 39-57 and 36-40 are depths of two buses, but buses 39 and 40 are without injections. As such, their voltages can be calculated once the voltages at buses 36 and 57 are determined.

The depth of two unobservability placement of Figure 2-8 results in 8 PMUs. There are fifteen unobserved buses; 5 are depths of one buses, the rest form groups of 4 depths of two buses. Note that a desired depth of unobservability structure cannot always be accomplished for typical meshed power system graphs. However, a majority of the unobserved buses will assume the desired depth of unobservability. Figure 2-9 shows a depth of three unobservability placement for the same test system. Table 2-1 shows comparative PMU placements on several systems. Results from complete observability placement [6] are included. Placement for incomplete observability results in significant reduction in the number of PMU requirements.

Table 2-1. Required Number of PMU Placements for Incomplete Observability

Test System	Size (#buses, #lines)	Complete Observability	Incomplete Observability		
			Depth-of-1	Depth-of-2	Depth-of-3
IEEE 14 Bus	(14,20)	3	2	2	1
IEEE 30 Bus	(30,41)	7	4	3	2
IEEE 57 Bus	(57,80)	11	9	8	7
Utility System A	(270,326)	90	62	56	45
Utility System B	(444,574)	121	97	83	68

IEEE 57 BUS SYSTEM

Figure 2-7. Depth of One Unobservability Placement on the IEEE 57 Bus Test System
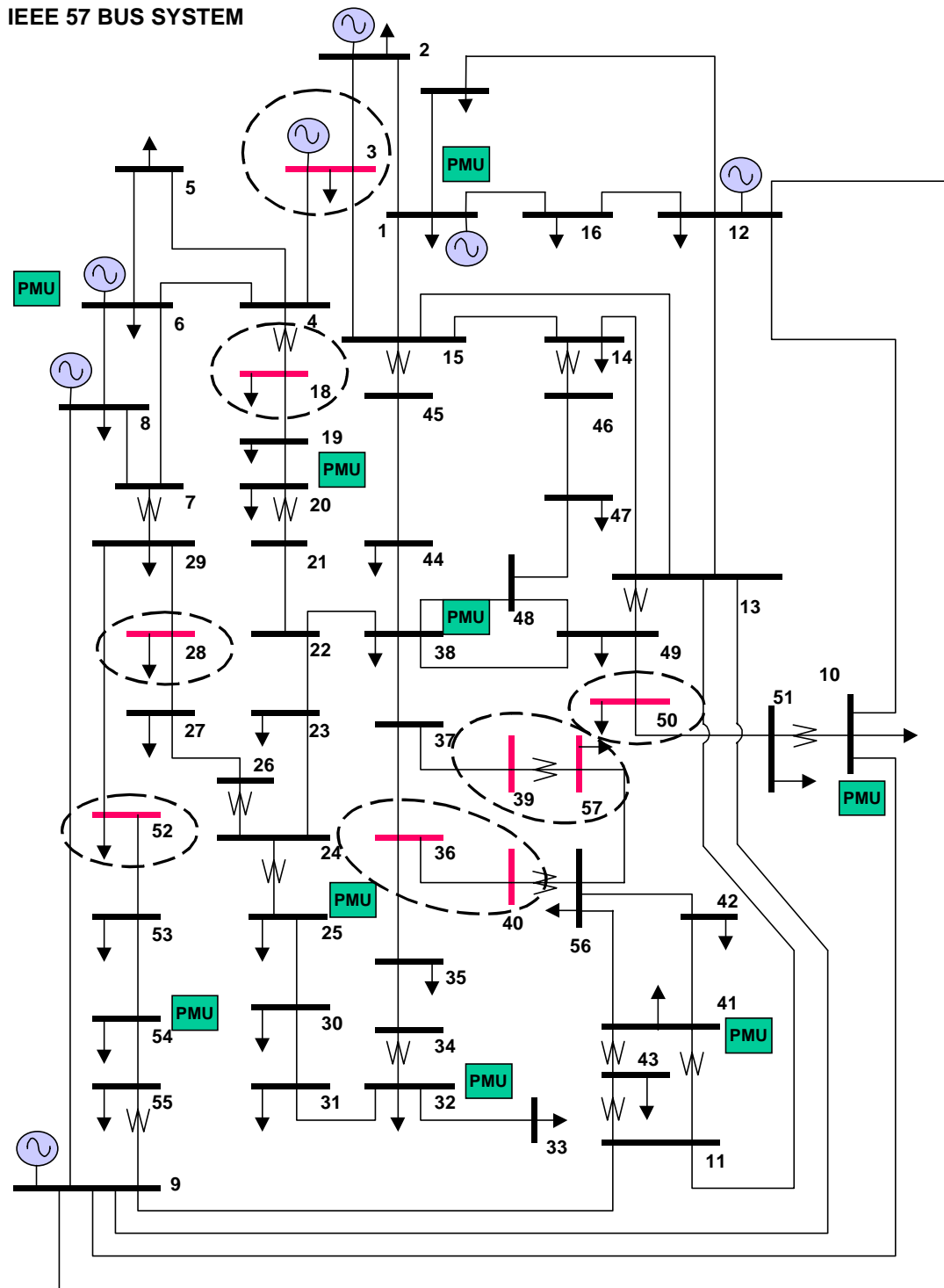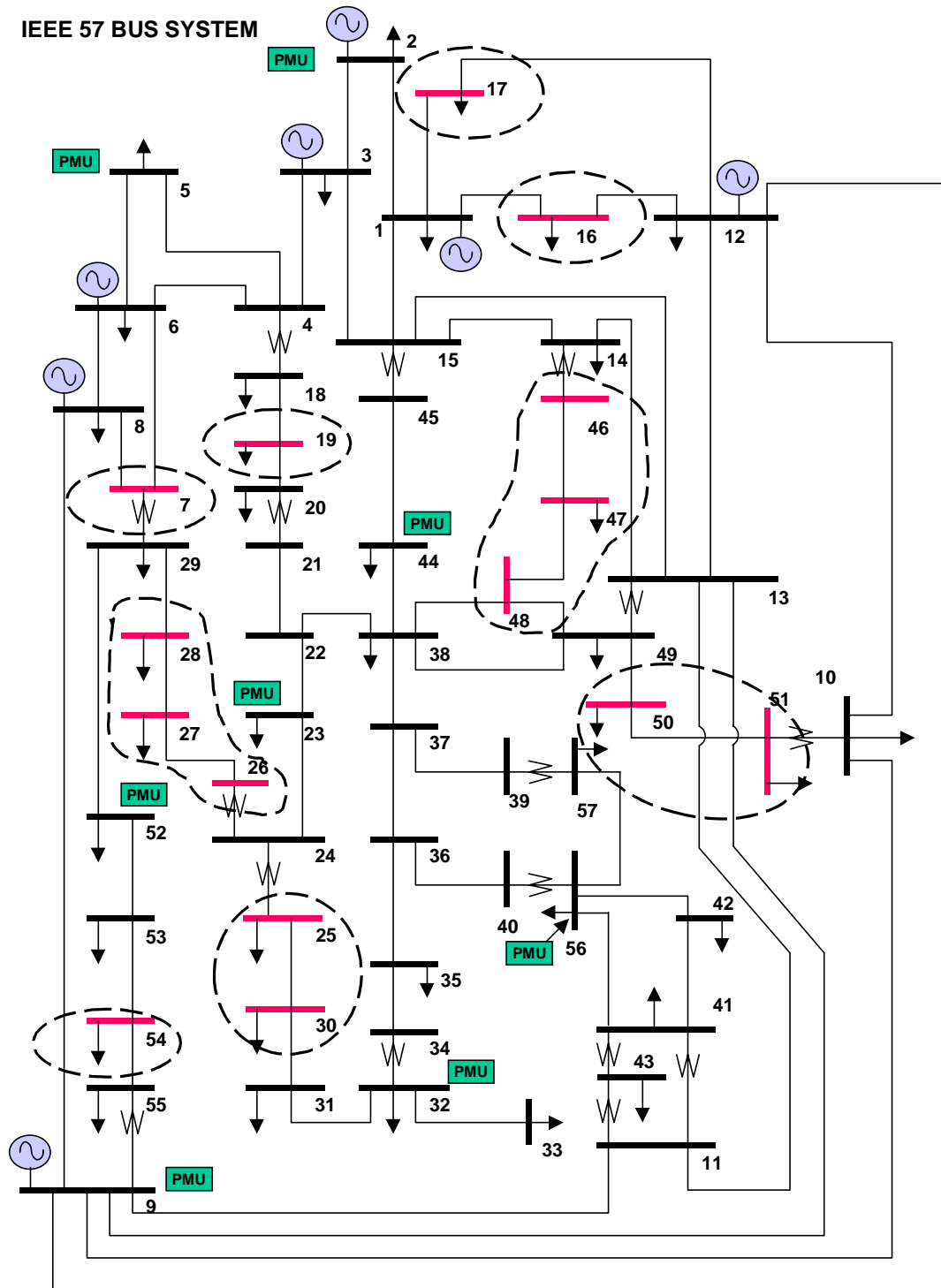
IEEE 57 BUS SYSTEM

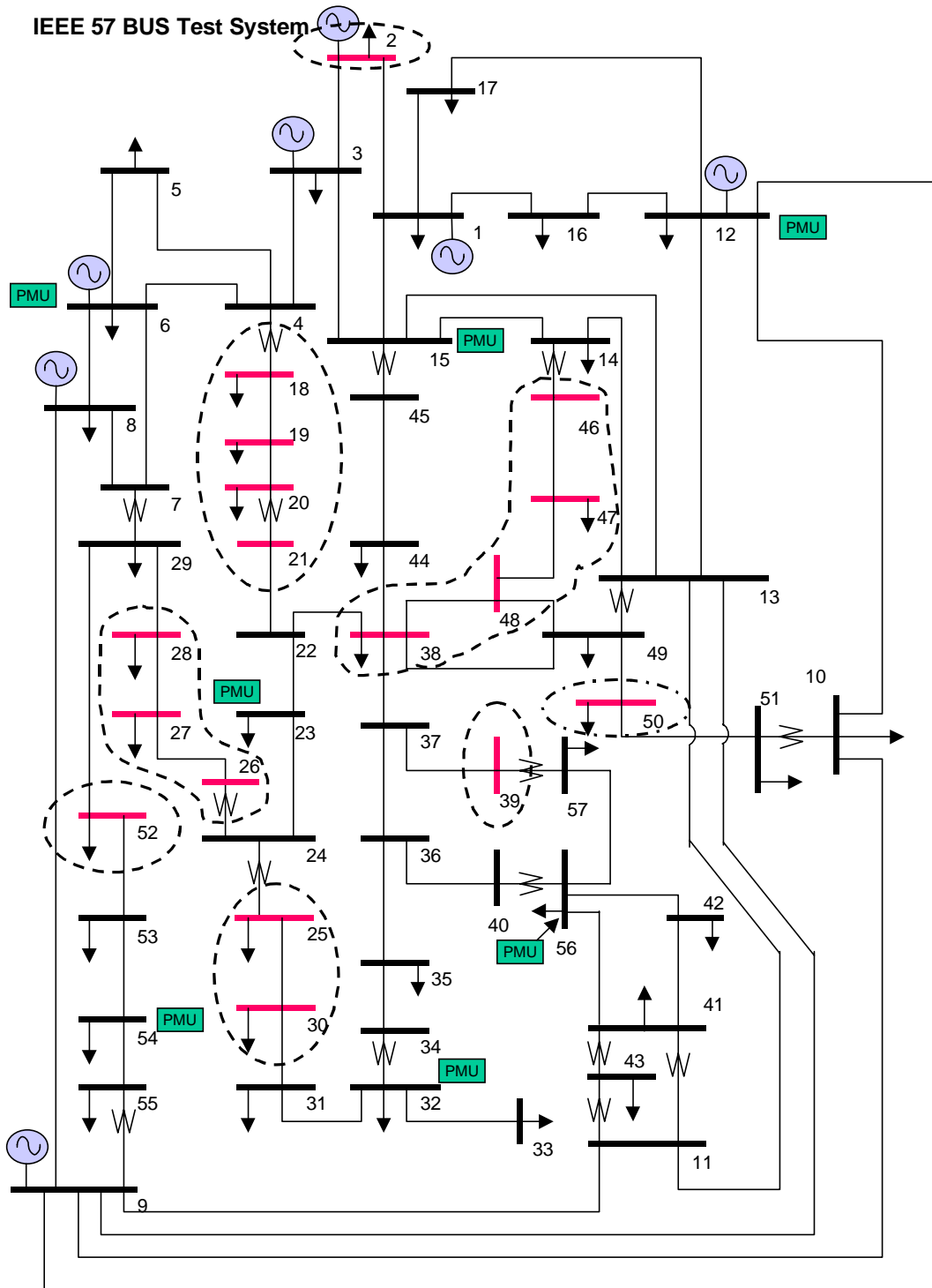Figure 2-8. Depth of Two Unobservability Placement on the IEEE 57 Bus Test System

Figure 2-9. Depth of Three Unobservability Placement on the IEEE 57 Bus Test System

2.5 Phased Installation of Phasor Measurement Units

The problem of phased installation of PMUs is dealt with in this section. The problem is how to progressively install PMUs in a network such that the minimum number of PMUs is always installed at any point in time. A further requirement of the problem should be that a given depth-of-unobservability is maintained at each point in time.

The way we approach this problem was to determine first an optimal placement for a depth-of-1 unobservability. We assume that this will be the ultimate scheme. Then, we remove a set of PMUs from this placement to achieve a depth-of-2 unobservability. We continue the process until no more PMUs can be removed from the network. This is seen as backtracking of the PMU placement through time.

The constraint of this problem is that initial PMU placements cannot be replaced at another bus locations. We modeled this as an optimization (minimization) problem with a pseudo cost function as being dependent on the target depth of unobservability.

The cost function is modeled in a way such that a PMU placement that violates a target depth of unobservability is penalized, while a PMU placement that achieves a target depth of unobservability result in a lowered value of cost function.

Cost Function Model

The pseudo cost function z with cost parameter c_j's is modeled as

$$z = \sum_{j=1}^{n} c_j d_j \tag{2.15}$$

where

$c_j < 0$, if $d_j \leq \mu_p$

$c_j >> 0$, if $d_j > \mu_p$

μ_p is the highest distance of any unobserved bus for a target depth of unobservability υ.

Here d_j is the distance of a bus from the nearest PMU; n is the number of buses. For a depth-of-1 unobservability, $\mu_p = 2$, which is the highest distance of any bus from its nearest PMU. It follows that, $\mu_p = 3$, for a depth-of-2 unobservability and so on. The cost function coefficients are picked in such a way that they obey the relationship, $c_j < c_k$ if $d_j < d_k$, so that buses with lower depths of unobservability are preferred over those with higher depths of unobservability.

Search Space

The search space in this optimization problem is the set of bus PMU locations that are candidates for removal. Strictly speaking, the size of this search space is large and is equal to 2^P -1points, P being the ultimate number of PMUs in the system. Each point is a binary assignment, $0 \vee 1$, for each of the buses with PMUs. Thus, the search point **x=[1000…0]** corresponds to a removal of the bus located at position one in the list. Exhaustive enumeration and search on all the points in the problem is prohibitive from the computational point of view however.

The way we attack this optimization problem is based on a random search on the search space. A search point is randomly generated by a permutation of bus locations with PMUs $\{b_1 b_2 b_3 \ldots b_P\}$. Then, a test is made if removing a PMU in this permutated list will improve the value of our cost function. If z is improved, its value is stored as the upper bound for z. Then, the same test is made for each of the elements in the list. The test is terminated when all elements in the list are exhausted. At this point, we have an upper bound on the pseudo cost function. Another permutation of PMU bus locations is again initiated and the value of the cost function is evaluated for this list. Our observation is that we attain convergence after 20 unique sets of permutated list of PMU bus locations.

Results of PMU Phased Installation Study

The aforementioned model and solution algorithm was applied to a study region within a big utility we identify as "Utility System A." This system consists of 270 buses and 326 lines. Its line p-q list is reported in the Appendix A. The phased installation strategy discussed here comprised six stages starting at depth-of-six unobservability and

culminating at depth-of-one unobservability. There are 3 PMUs whose locations are fixed.

Table 2-2 presents the bus locations of PMU placements at Utility System A. For all stages of the phased installation, we have relaxed the observability requirement at all terminal buses with net injection of less than 20 MW. There are 6 such buses. Placing a PMU to a bus with non-zero injection, which is linked to a terminal bus whose injection is less than 20 MW, is not a requirement. In other words, the only way with which a PMU is placed in these buses is because it is an optimal location with respect to the rest of the PMU placements.

Table 2-2. Results of PMU Phased Installation Exercise for Utility System A

Stage ID	Desired Depth of Unobservability	Number of PMUs	Number of' Unobserved Buses
6	6	27	160
5	5	35	129
4	4	40	114
3	3	45	94
2	2	56	67
1	1	62	57

It was mentioned in before that the target depth of unobservability is rarely achievable to all unobserved buses in the network. The reason lies on the network connectivity.   To provide us a picture of the depths of unobservability of the network with this placement, the network was mapped. Mapping here refers to the determination of the **distances** of the network buses from the nearest PMU(s). From this the depth of unobservability of each bus in the network can be determined. Table 2-3 shows a tabulation of the depths of unobservability of the buses for all stages of the phased installation.

Table 2-3. Distribution of Depths of Unobserved Buses Resulting from Phased
Installation of PMUs

Stage ID	Target Depth	Depth-of-Unobservability						
		1	2	3	4	5	6	7
1	7	11	12	74	13	34	2	14
2	5	18	23	58	15	15		
3	4	28	22	49	15			
4	3	38	23	33				
5	2	47	20					
6	1	53	4					

Chapter 3. Simulated Annealing Solution to the Phasor Measurement Unit Placement Problem with Communication Constraint

3.1 Introduction

The PMU placement algorithm presented in Chapter 2 assumes that each substation in the power system is capable of transmitting the PMU measurements to a central location such as a control center. The substation could have existing communication facilities. It is possible that the PMU placements could justify the installation of new communication facilities.

However, we seldom find ubiquitous communication infrastructure in power systems for economic reasons. The communication facilities needed to support a phasor measuring system costs a lot more than the PMU devices themselves. It must be recognized that oftentimes only the existing communication facilities will support new PMU installation. Hence, the constraint posed by inadequate communication facilities must be taken into account in the PMU placement problem.

The location and number of buses with communication facilities greatly affect PMU placements. It might be possible to extend the graph theoretic PMU placement algorithm to account for "no placement" buses. This can be the subject of future research.

In this thesis, the method of simulated annealing is used to solve the PMU placement problem constrained by deficiency of communication facilities. The two required elements of simulated annealing - the cost function and transition moves will be developed in Sections 3.3and 3.5, respectively. The objective is to minimize a scalar cost function that measures the energy of a particular PMU configuration. Designed so that strategies that recognize the communication constraint will carry less energy, the cost function likewise respond to parameters such as depth of unobservability, and the size of PMU placements.

3.2 Brief Review of Simulated Annealing

In the 1980's two authors, Kirkpatrick and Cerny, independently found that there exist a close similarity between minimizing the cost function of combinatorial optimization problems and the slow cooling of solid until it reaches its low energy ground state. It was termed simulated annealing (SA). Since then research on applications of the SA algorithm permeated the field of optimization. In electrical engineering, for example, SA found applications in computer-aided circuit design, mainly on layout problems for VLSI circuits. One of the objectives was to minimize the area of the VLSI chip by optimally placing its modules such that the wires connecting the modules occupy the least area as possible (see Chapter 7 of [41]). In the area of energy systems, SA was successfully applied to the unit commitment problem [68].

Simulated annealing (SA) belongs to a class of optimization techniques akin to solving combinatorial optimization problems requiring the solution of optima of functions of discrete variables. Many large-scale combinatorial optimization problem can only be solved approximately mainly because they have been proved NP-complete, meaning the computational effort required to solve the problem is not bounded by polynomial function of the size of the problem. A combinatorial optimization problem seeks to minimize (or maximize) an objective function C by searching for the optimal configuration s_{opt} among the often countably infinite set of configurations S (3.1). The cost function C, $C: S \rightarrow \mathbf{R}$, assigns a real number to each configuration in S.

$$z = C(s_{opt}) = \min_{s \in S} C(s) \tag{3.1}$$

In most situations, the solution to (3.a) is an approximation that is always taken as the best solution within an allotted computation time. The iterative improvement algorithm is one of such solution technique. Iterative improvement is a local or neighborhood search mechanism that incrementally analyzes neighboring configurations of the incumbent solution. Given a configuration j, its neighborhood S_j is a collection of configurations in proximity to j. Iterative improvement proceeds as follows. Given a configuration j and its neighborhood S_j, iterations are made on each of the members of the neighborhood. If the cost of a neighboring configuration is less than the current

configuration, then it becomes the incumbent solution. Otherwise, another neighbor is
sampled and tested. The process is repeated until the point when the cost of the current
configuration is approximately the same as its neighbors. The iterative improvement
suffers from several drawbacks. First, it was observed that the optimal solution is a
function of the initial configuration. Second, the algorithm is sometimes trapped in a
local minimum from which it is impossible to escape. An alternative solution approach
must be capable of escaping local minima and should be autonomous from the initial
configuration. Simulated annealing is one such popular technique that embodies these
desirable characteristics of a combinatorial optimization algorithm.

A profound analogy between the simulation of annealing of solids and the
optimization of combinatorial optimization problems exists. But first the physics of
annealing is analyzed. In annealing, a solid in a heat bath is heated by increasing the
temperature of the bath to a maximum value so as to allow the particles of the solid to
arrange randomly in the liquid phase. Subsequent gradual cooling then makes the
particles arranged in a lattice. The Boltzmann distribution (3.2) characterizes the
probability of being in a state x_i with energy $E(x_i)$ when the solid is in thermal
equilibrium at temperature T.

$$\Pr\{X = x_i\} = \frac{\exp(-E(x_i)/k_BT)}{\sum_{j=1}^{N}\exp(-E(x_j)/k_BT)} \tag{3.2}$$

The factor $exp(-E/k_BT)$ is known as the Boltzmann factor. As the temperature
decreases, (3.2) implies that only those states with the lowest energy have a high
probability to exist. Furthermore, when the temperature approaches the absolute
minimum, only those states with the *minimum energy* will have a non-zero probability of
occurrence. The evolution to thermal equilibrium of a solid is simulated using the
Metropolis algorithm. This Monte Carlo method generates a sequence of states in the
following manner. First, given the current state characterized by the position of the
particles a small perturbation is induced by randomly displacing a particle. Then, the
difference in energy ΔE between the current state and the perturbed state is compared. If
$\Delta E < 0$, then the new perturbed state is accepted. If $\Delta E > 0,$ then the probability of
acceptance of the perturbed state is given by $exp(-E/k_BT)$ – the *Metropolis criterion*. The

process of perturbing the current state continues until the point when the system achieves thermal equilibrium. Simulated annealing can be viewed as a sequence of Metropolis algorithms applied to gradually decreasing temperature.

There is a profound analogy between the Metropolis algorithm and a combinatorial optimization problem. The configurations S can be likened to states of the solids x; the energy E is likened to the cost function C while the temperature T is likened to the control parameter c (we have decided to retain the notation T for the control parameter to be consistent with what has been used in literature) . Simulated annealing can now be viewed as a series of Metropolis algorithms evaluated at decreasing values of control parameter c. The SA algorithm presupposes the description of configurations, cost function, and transition. A configuration defines an arrangement of the objects whose position is to be optimized. A configuration in S is generally a permutation; for example in the popular TSP (traveling salesman problem) it is the sequence of cities to be visited. Similarly, in the unit commitment problem, configuration is a permutation of unit statuses, that is, a sequence of binary variables describing which units are committed or de-committed.   The cost function is a mapping $C: S \rightarrow R$, which assigns a real value to a configuration. Transitions are a collection of configurations S_i in the neighborhood of configuration i$\subset S$. A mechanism to generate these transitions is an integral component of the SA algorithm. The SA algorithm in pseudo-code is described in Figure 3-1.

INITIALIZATION. Set initial value of control parameter T_t. Generate initial

configuration *i*.

DO

       m=0;

       M = f(T); The upper bound on the number of considered configurations at *T*

       DO

              Perturb configuration *i*. $i{\rightarrow}j; \Delta C_{ij} = C(j) - C(i)$

              If $\Delta C_{ij} < 0,$ then *j* becomes new state else

                     If $\exp(-\Delta C_{ij}/T)>$ *random[0,1]* then *j* becomes new state;

              m$\rightarrow$m + 1;

       while m < M

       $T{\rightarrow} T_{new}{=}f(T);$

while (stop criterion is not met)

END.

Figure 3-1. The Simulated Annealing Algorithm in Pseudo-Code

3.3 Modeling the Communication Constrained PMU Placement Problem

    We now formally define the variables in the PMU placement problem within the
context of the simulated annealing algorithm. Let $\mathbb{S}$ be the search space consisting of
finite collection of PMU placement strategies. Each configuration $S \subset \mathbb{S}$ refers to the
vector of PMU bus locations.

The Cost Function

      We propose a cost function (3.3) that seeks to maintain the desired depth of
unobservability while recognizing that certain buses cannot have PMU placements. It is
important to admit the fact that the depth of unobservability cannot be faithfully
recognized throughout the power system. A skew in the revised PMU placement strategy
is to be expected. However, the cost function is modeled such that it minimizes
deviations from the desired depth of unobservability. The cost function must model two

important elements: minimizing deviations from the desired depth of unobservability and ensuring placement only on buses with communication facilities. Hence, we model the cost function as a scalar valued real function that measures the level of energy of a placement strategy S. It has three components; (a) $C_1(S)$ - associated with cost of deviation from the desired depth of unobservability; (b) $C_2(S)$- associated with a penalty for placement on buses without communication facilities and; (c) $C_3(S)$- associated with cost for the number of PMU placements.

$$C(\mathbf{S}) = C_1(\mathbf{S}) + C_2(\mathbf{S}) + C_3(\mathbf{S}) \tag{3.3}$$

We now proceed to develop the three components.

<u>The Cost Component to Minimize Deviations from the Desired Depth of Unobservability</u>

To maintain the desired depth of unobservability associated with a particular PMU placement strategy, each unobservable region need to be checked for its size and structure. The *observability algorithm* presented in Section 2.3 can identify the unobserved buses U_S associated with a meter placement scheme. From this set we can identify the collection of unobserved regions $(\mathbf{R_u}) = \{\mathbf{R_1}, \mathbf{R_2}, \ldots, \mathbf{R_N}\}$. Each unobserved region $\mathbf{R_i}$ is a set of buses that forms a connected graph, that is, each one is a sub-graph by itself. The algorithm presented in Section 3.4 shows how to generate these unobservable regions. Any unobservable region $\mathbf{R_i}$ with structure conforming to the desired depth of unobservability presented in Section 2.2 will not affect the cost function. Otherwise, $\mathbf{R_i}$ will increase the cost of its associated strategy to a degree dependent on its size and structure.

Let $(\mathbf{R_x})$ be the collection of unobserved regions that does not conform to the desired structure for a depth of unobservability υ. To determine this collection, it is necessary to determine first the number of buses $|\mathbf{R_i}|$ of each unobservable region in $(\mathbf{R_u})$. That region with number of buses greater than υ, $\{\mathbf{R_i} : |\mathbf{R_i}| > \upsilon\}$, is a member of $(\mathbf{R_x})$. Having determined $(\mathbf{R_x})$, the subject cost component can now be posed and is mathematically stated in (3.4).

$$C_1(\mathbf{S}) = K_1 * \sum_{\forall \mathbf{R}_j \subseteq (\mathbf{R}_x)} \sum_{x \in \mathbf{R}_j} \max(0, d(x) - \upsilon + 1)^2 \tag{3.4}$$

where $d(x)$ is the distance of any unobserved bus x. The constant K_1 is a tunable

weighting factor that varies depending on how faithfully we wish to maintain the depth of

unobservability υ. In essence, (3.4) will result in higher cost for regions with depths of

unobservability greater than υ. The square term will amplify this cost exponentially for

greater violations.

Penalty Term for Placement on Buses without Communication Facilities

Another component in the cost function is a cost penalty term for placements on

buses without communication facilities contained in the set $\mathbf{W_O}$. It is desirable to model

this penalty term as varying on the location of the bus so as to allow SA to differentiate

penalty amongst buses. Since initially we are given a placement that includes PMUs on

'no-placement' buses, it would be helpful for SA to accelerate the location of a bus with

communication. Here, we introduce a vector A whose elements $A(x)$ defines the distance

of any bus x without communication from its nearest bus neighbor $B(x)$ that has

communication. Both of these vectors are generated by algorithm 'Minimum Distance

from Buses with Communication Facilities' developed in Section 3.4. Mathematically

depicted in (3.5), the penalty for placement on prohibited buses is once more a quadratic

function of the minimum distance vector A. It consists of a weighting factor K_2 that scales

the extent of violation for placement on prohibited buses.

$$C_2(\mathbf{S}) = K_2 * \sum_{x \in Wo} A(x)^2 \tag{3.5}$$

K_2 must be set such that $K_2 >> K_1$ to force the SA algorithm to initially prioritize

movement towards the set of buses with communication facilities.

<u>The Cost Component Due to Size of PMU Placements</u>

A component of the form (3.6) is incorporated to the overall cost function to allow SA to distinguish between strategies with differing number of PMUs so as to minimize the number of PMUs. This model should be viewed within the context of *transition moves* (see Section 3.5) in simulated annealing. It implies that removal or moving a PMU to a redundant location is a SA possible transition move. Normally, violation of the desired depth of unobservability should not be compromised with decreasing the size of the PMU placement set. Hence, the $K_3 << K_1$ must hold true at all times.

$$C_3(\mathbf{S}) = K_3 * |\mathbf{S}| \qquad\qquad (3.6)$$

3.4 Graph Theoretic Algorithms to Support SA Solution of the Constrained PMU Placement Problem

The cost function model for the proposed simulated annealing solution requires two graph theoretic routines. Figure 3-2 depicts the pseudo-code to generate the sub graphs of unobservable regions created by the PMU placement strategy **S.** This algorithm basically groups the unobservable buses *U* into several regions, each of which is connected. A test is made if any unobserved bus *x* is a member of any one of the existing regions, that is, if any of its adjacent bus y$\in \ell_x$ is also a member of the same region. If the test fails, a new unobserved region is created.

Figure 3-3 generates the vector of 'minimum distance' *A* wherein an element *A(x)* contains the minimum distance from bus *x* to the nearest bus with communication facilities.

Step 1. Given the set of unobserved buses **U**, initialize the number of
   unobservable regions $n \leftarrow 0$; Assign **V** = **U**
Step 2. While **V** is not NULL
Step 3. $n \leftarrow n + 1;$ Create new region
Step 4.   Initialize first element of region $n;$ **R_n** $\leftarrow$ *{U(1)};* Set **Bin** $\leftarrow$ *{U(1)};*
   B is NULL;
Step 5.   While **Bin** is NOT NULL
Step 6.   Loop for all $x \in$ **Bin**
Step 7.   **B** = **B** $\cup$ {y: y$\in \ell_x$ and y$\notin$ **R_n**}; Bus links to set **Bin** not in
   current region **R_n**;
Step 8.   CONTINUE loop for all $x \in$ **Bin**
Step 9.   **Bin** $\leftarrow$ **Bin** $\cup$ **B**;
Step 10.   IF **Bin** is not NULL THEN: Put **Bin** in **R_n**; **R_n** $\leftarrow$ **R_n** $\cup$ **Bin**; GO
   TO Step 5; ELSE GOTO Step 11
Step 11.   **V** = **V - R_n**.
Step 12.   IF **V** is NOT NULL THEN: GOTO Step 2; ELSE STOP a collection
   of n regions **R_1, R_2, $R_{3...}$ R_n** is complete.

Figure 3-2. Algorithm to Build Sub-Graphs of Unobservable Regions

Step 1.   Given the vector of buses without communication facilities Wo, the
vector of buses with communication facilities Wi, initialize the
'minimum distance' vector: $A(\text{x}) \leftarrow 0, \forall \text{x} \in Wo$; $A(\text{x}) \leftarrow \infty, \forall \text{x} \in Wi$;
Step 2.   Loop for all $x \in Wo$
Step 3. $Basket$ is NULL
Step 4. $Bin \leftarrow \{x\}$
Step 5.   Loop for all $y \in Bin$;
Step 6.   Loop for all $z \in \ell_y$;
Step 7. $A(z) = minimum(A(z), A(y)+1)$;
Step 8.   IF $z \in Wi$, THEN $A(x) \leftarrow A(z)$; GOTO Step 10; ELSE
Step 9. $Basket \leftarrow Basket \cup \{z\}$; GOTO Step 10
Step 10.   CONTINUE z
Step 11.   CONTINUE y
Step 12. $Bin \leftarrow Basket$
Step 13.   GO TO Step 5
Step 14. CONTINUE x

Figure 3-3. Algorithm to Find the Minimum Distance to a Bus with Communication
Facilities

3.5 Transition Techniques and Cooling Schedule for the PMU Placement Problem

This section presents a transition technique used to perturb a given configuration. A transition is a small perturbation of the existing PMU strategy *S*. In the literature, a number of generic transition strategies are proposed. The term 'length' is used to measure the distance between any two strategies or states in the SA. Here we have adopted a length of one, which means that any perturbation around the existing state differs by only one PMU bus position. Formally, if $\Phi(S)$ denote the collection of strategies of length one with respect to *S*, and $V_\Phi{}^j$ denotes the set of buses of j_{th} strategy in Φ, then (3.7) is a necessary condition for length one perturbations around *S*. It is assumed that the size of some of the perturbations can be smaller than the existing strategy, that is, $|\,V_\Phi{}^j\,|-|S| = 1$ can hold true $\exists\ V_\Phi{}^j$. This describes the condition when $V_\Phi{}^j$ is generated when an *x* PMU in *S* is *fused* (merged) with another *y* PMU in *S*. The way this is implemented in the algorithm is to make a move from a PMU bus in *S*, termed as the *root bus* to another bus.

$$\left| \mathbf{S} - \mathbf{V}_\Phi^j \right| = 1, \quad \forall \mathbf{V}_\Phi^j \subset \mathbf{\Phi}(\mathbf{S}) \tag{3.7}$$

A computational issue that needs to be resolved is the size of the collection of perturbations around *S.* Significant savings in computation time can be realized by carefully choosing the transition strategies. Baldwin, et al, [5] treated each non-PMU bus in *S* as a feasible bus to visit; the choice is made at random. In effect, this approach slows down the convergence process because all buses, even those of significant distance from the root bus, are considered equally likely to be visited. Although theoretically we can investigate |*S*|-1 transitions associated with a PMU in *S*, the computational requirement may be decreased if the set of possible moves is reduced. It is conjectured that the upper bound on the number of transitions rooted on a PMU consists only of movements to buses of distance less than or equal to γ as defined by (3.8). The rest of the PMUs provide cover for the other regions in the system. If we let Γ_x be the set of buses with distance less than or equal γ from a PMU bus *x,* then the transition move

consist of replacing PMU bus x to any bus in Γ_x. Figure 3-4 illustrates how a transition move rooted in x is made.

$$\gamma = \upsilon + 2 \qquad\qquad (3.8)$$
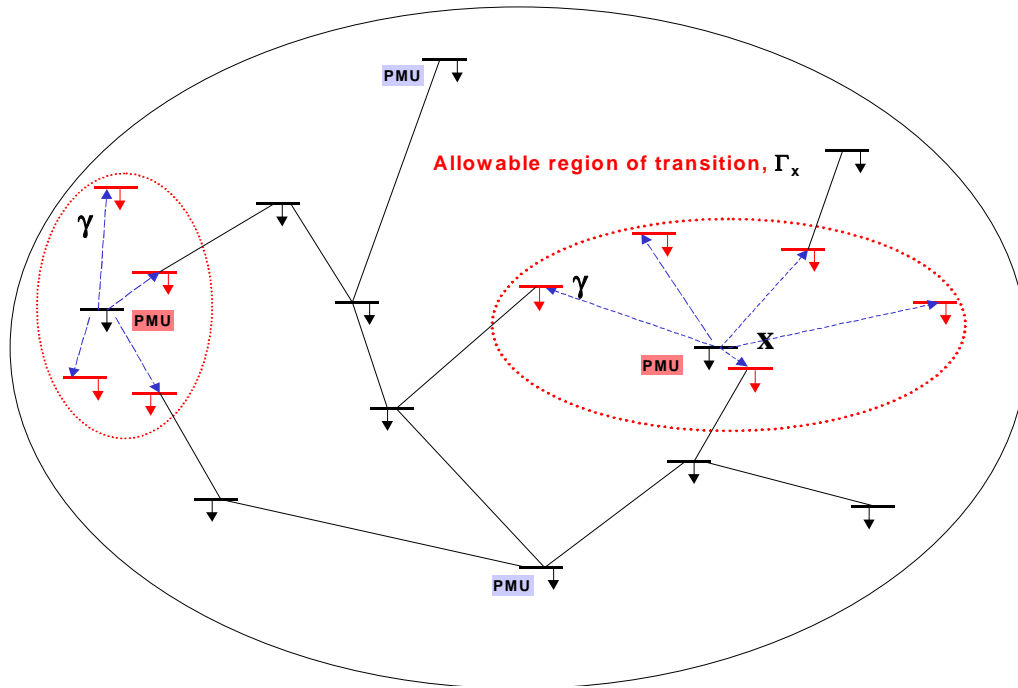
Figure 3-4. Regions of Transition: Extent of transition moves rooted at a PMU bus x

In some instances, a transition from a PMU bus x to another PMU bus y may happen in which case a merging of two PMU locations will occur. As long as this transition would not result in significant increase in the value of the cost function C, that is, the desired depth of unobservability is not violated, the size of the accepted strategy will be reduced by one PMU. This was the motivation behind including cost component associated with of PMU size in C (3.6). Therefore, this transition strategy both optimizes the location of PMUs and reduces their required number.

The Initial PMU Placement Strategies

Theoretically, we can randomly generate an initial placement strategy to jumpstart the SA algorithm. Although any random permutation of PMU locations with number not exceeding the least upper bound defined by (2.2) may be a candidate, a prudent choice

will be an initial strategy that has a well defined structure conforming to the depth of unobservability. Thus, we used the graph theoretic placement technique to generate the initial placements for all the exercises in this study.

Cooling Schedule for The Constrained Placement Problem

The cooling schedule is a prescription for the following parameters:

- Initial value of control parameter;

- Final value of control parameter;

- Upper limit on the number of strategies considered at each stage c_k of the annealing process; and

- Decrement of the control parameter.

In literature it is frequently suggested to choose an *initial value of the control parameter c_o* such that a large percentage (80% to 90%) of the proposed transitions are accepted. However, it is accepted that there is no general prescription in choosing the value of c_o because it is problem specific. Zhuang and Galiana [68] conducted numerical experimentations to find the suitable value. We found out that numerical experimentation works best for the placement problem and have adopted this approach in the thesis.

Similarly, in choosing the *final value of the control parameter c_f* there is no specific rule. In this thesis, the search process is normally terminated in 50 iterations on the control parameter c. However, the search is terminated early when there is no more reduction in the cost function at any given value of control parameter.

The number of *transitions considered per control parameter M* should be related to the size of the problem. Here we have assumed M to a multiple of the size of the number of PMUs, *M=k|S|,* with k normally set to 5.

The *control parameter is decremented* according to (3.9). Numerous experimentations were done to determine the optimal value of α. For the PMU placement, this value is 0.8.

$$c_{k+1} = \alpha \cdot c_{k+1} \tag{3.9}$$

3.6 Results on Study Systems

The model is tested using a real world power system. This power system is a study region inside a major electric power utility in the US. The test system used to test and validate the model in the previous sections was taken from a big electric power utility. Lists of 306 buses whose voltages need to be monitored were initially provided. Of this list, 165 buses have communication facilities and links to the control center in the form of radio, and fiber optic lines. The 306 buses is not a connected system by their own, rather it is a collection of several isolated islands. At first a minimum of 76 buses was integrated to the study system to make it connected. It became clear in this initial exercise that there were a high number of terminal buses. Since placements are mandatory on the terminal buses or on their bus links, this interim study system will skew the placement process by requiring significant number of non-movable PMUs. As a consequence, the final system model was built from the interim model by identifying and connecting paths in the external system emanating from terminal buses and looping back to the study region. The study system (Utility System B) is summarized in Table 3-1. It has 444 buses and 574 lines.

Table 3-1. Basic System Data for Utility System B

Number of Buses	444
Number of Lines	574
Number of Terminal Buses	46
Number of Buses with Communication	165
Number of Buses without Communication	275
Number of Buses without Active Injections	48
Number of Boundary Buses	271

The buses without communication facilities are formed in several groups of contiguous buses distributed throughout the study region. Figure 3-4 shows that a significant number of these buses form 2 to 14 buses per group. There are 92 bus groups without communication facilities; 35 of these bus groups have 2 or more buses.
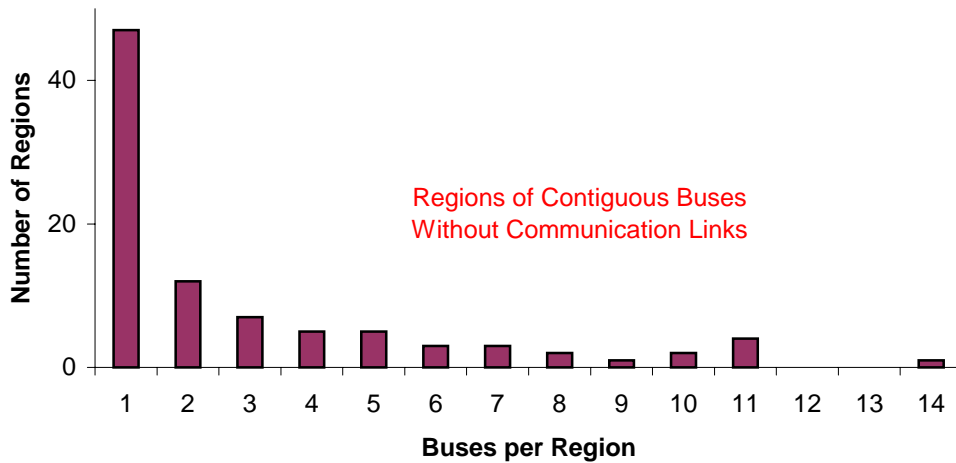
Figure 3-5 Distribution of Buses without Communication Links in the Utility System B

Table 3-2 shows a tabulation of the PMU placement strategies for Utility System B using the graph theoretic algorithm of Chapter 2. As there are only 165 buses with communication links, the total misplaced[1] PMUs (in column 4) are proportionately high. The mandatory placements on the 46 terminal buses were also a significant factor that affected the listed placements.

The locus of convergence in terms of the cost function value (3.3) is shown in Figure 3-8. The choice of weighting constants are $K_1 = 10^3$; $K_2 = 10^6$; and $K_3 = 150$. A total of 40 time steps were used. Numerical experimentations were done to determine the optimal value of the initial control parameter T_0. The summary of these experiments is plotted in Figure 3- 6 from which the optimal initial value of the control parameter is $T_0=5x10^3$. With the assigned weights SA prioritized reassignment of misplaced buses, followed by optimization on location of all PMUs.

---

[1] Set of PMUs assigned to buses without communication facilities
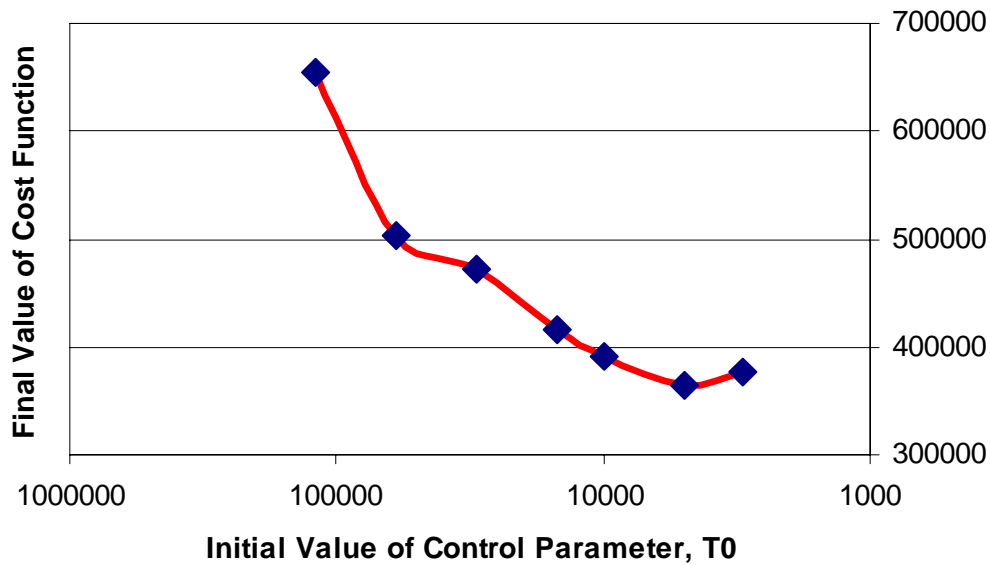
Figure 3- 6. Variation of Cost Function with Initial Value of Control Parameter,T0

Table 3-3 shows the results of the simulated annealing solution to the communication- constrained problem. Note the reduction in the number of PMUs due to non-placements on the 275 buses. Thus, in the process of moving the misplaced PMUs to substations with communications some were moved to redundant[2] locations or replaced to more effective locations by SA. A noticeable consequence of the PMU replacements is the increase in the number of unobserved buses. However, for depths of 3 and higher the change in number is only incremental. This is because at higher depths the difference in the required number of PMUs is small. SA was able to move misplaced PMUs without significantly reducing the size of the observable buses. Concerning the number of unobservable regions, two observations can be made. First, for depths of one and two unobservability the lower numbers for the SA solution are accompanied by an increase in the number of regions with higher than the desired depth of unobservability (see Figure 3-7). However, depths of three and higher placement resulted in higher number of regions. With very close number of unobserved buses, this means that SA split a significant number of the original unobserved regions into smaller size regions by the optimal replacements of the PMUs.

---

[2] Redundant position here refers to placing a PMU to another PMU bus

Table 3-2. Initial PMU Placement Strategies for Utility System B

Case ID	Desired Depth of Unobservability	Number of PMUs	Total Misplaced PMUs	Number of' Unobserved Buses	Number of Unobservable Regions
A	1	97	54	99	82
B	2	83	52	150	82
C	3	68	39	191	73
D	4	60	34	216	57
E	5	55	34	238	44

Table 3-3. Simulated Annealing Solutions to the Communication Constrained PMU Placement Problem for Utility System B

Case ID	Desired Depth of Unobservability	Number of PMUs	Total Misplaced PMUs	Number of' Unobserved Buses	Number of Unobservable Regions
A	1	78	0	136	63
B	2	67	0	174	72
C	3	59	0	194	74
D	4	52	0	223	72
E	5	50	0	233	66

Figure 3-7 illustrates the distribution of the depths of unobserved buses for a depth of one placement (with and without communication constraints). The figure illustrates that for a constrained placement some buses relegate to higher depths. Note the existence of depths of 8 to 10; this peculiarity is more the result of big regions of contiguous buses without the required communication facilities rather than the effect of replacement of buses by SA.
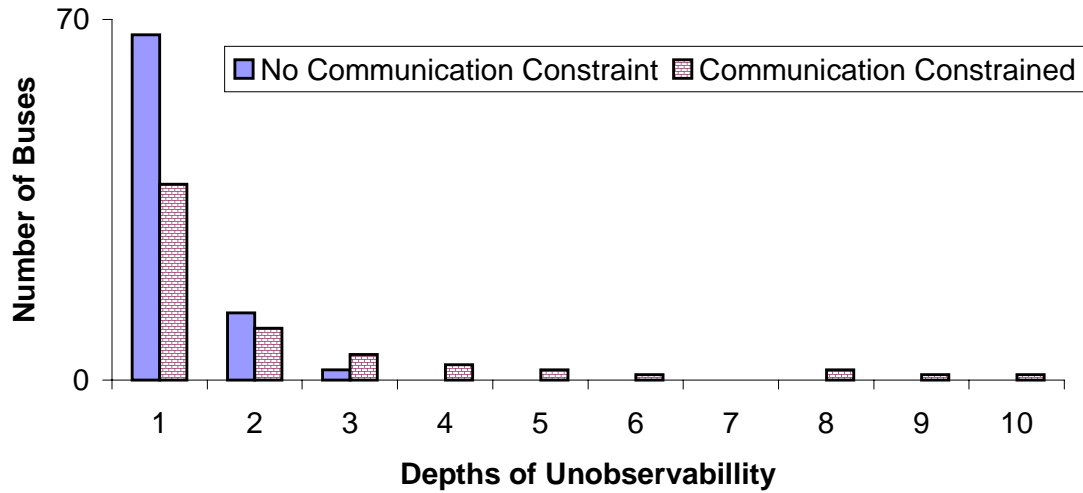
Figure 3-7. Adjustments in the Depths of Unobservability of Buses Due to
Communication Constraints

The convergence record for depth-of-one unobservability placement for Utility System B is shown in Figure 3-8, plots A, B and C. The points in the locus correspond to accepted transitions or successful cost function changes. Note the monotonically decreasing cost function.   The overriding spikes are the result of accepting higher costs with probability $exp(-\Delta C_{ij}/T)$. The sharp drop in the locus halfway implies that all PMUs are positioned on buses with communication facilities. The blown up plots B and C contains the same number of points in the locus; plot B corresponds to only the initial value of the control parameter T_0=5000 while plot C corresponds to seventeen (17) decrement changes in T. Both plots validate the expected behavior of simulated annealing: higher value of control parameter T (plot B) there is higher probability of accepting strategies with higher cost.
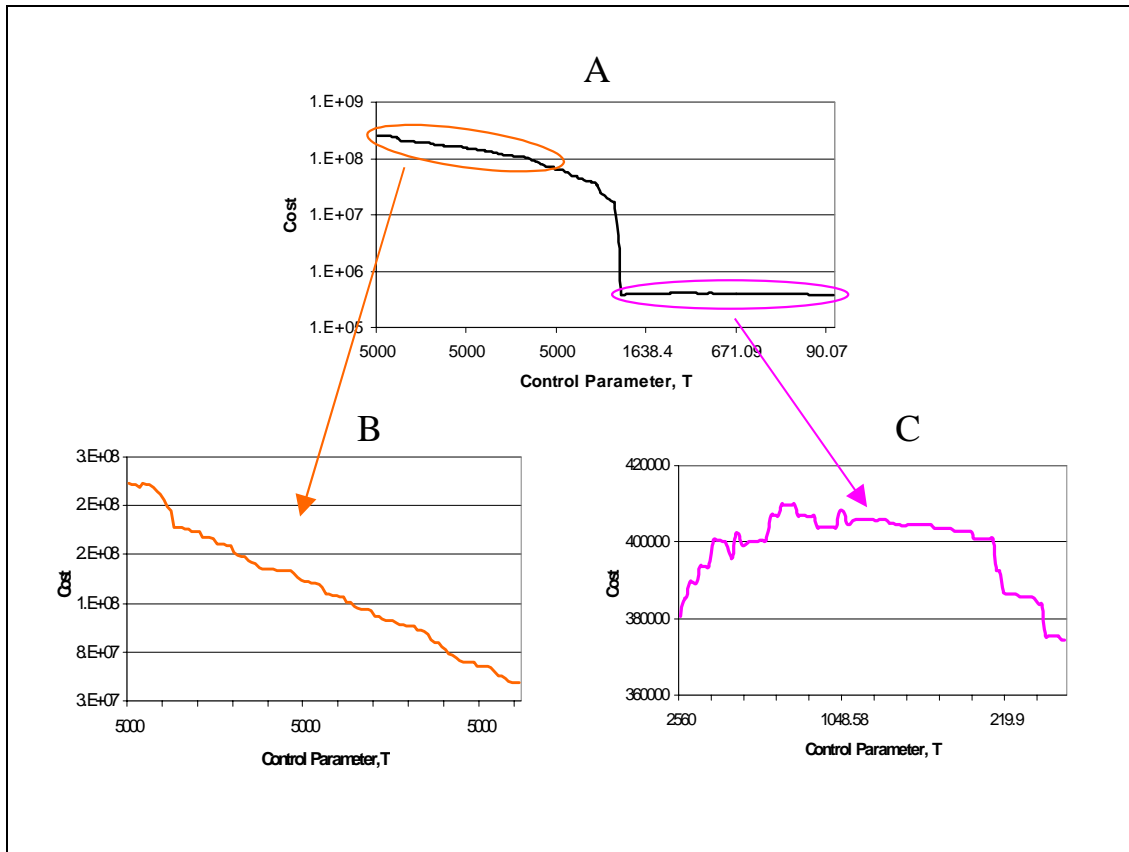
Figure 3-8. Simulated Annealing: Convergence Record of Communication Constrained Placement for Utility System B

Phased Installation of PMUs with Communication Constraints

Another variant to the SA modeling of the PMU placement problem is the phased installation of the PMUs. This is a strategy whereby PMUs are installed in batches starting from a high depth of unobservability to lower depths. The primary constraint is not being able to move earlier installed PMUs to another bus locations. This problem can be approached in the following way: do a SA starting with depth of one but with transitions confined to only the merging of the root bus and another PMU (in effect vanishing the root PMU) with desired depth of unobservability relaxed to depth of 2. The depth-of-2 placement is then again submitted to SA to find the new PMU bus locations for a depth-of-3 placement, and so on. The constraint of non-replacement of previously installed PMUs makes the search space of this technique small. As such, higher number of PMUs is expected for the various depths of unobservability compared with the non-

phased installation technique. The larger placement sets are evident from Table 3-4, which summarizes the results of the phased installation technique. One suggested phased strategy is to start with depth-of-5 placement (Stage 1: 59 PMUs), followed by depth-of-3 (Stage 2: 69 PMUs), and finally depth-of-zero (Stage 3: 88 PMUs). Here desired depth-of-zero connotes placement for complete observability. Figure 3-9 shows a graphical distribution of the unobservable regions at each stage. Observe the gradual reduction in the depth of the unobserved regions as we move towards the later stages of the phased installation scheme.

It is important to recognize that the desired depth-of-unobservability structure cannot be realized in all unobserved buses; thus, the term *desired* has been ubiquitously used in this thesis. This aberration in the placement process has been discussed in Section 2.3, but it becomes more evident in the presence of communication constraints. A fitting example is Phase ID 0 in Table 3-4. Because of communication constraints, there still exist unobservable regions in spite of the *desired* depth-of-zero placement strategy. The location and number of communication facilities for Utility System B simply cannot achieve complete observability. It further implies that positioning PMUs in all buses provided with communication facilities simply cannot observe the system.

Table 3-4. Results of PMU Phased Installation Exercise for Utility System B: Limited Communication Facilities

Phase ID	Desired Depth of Unobservability	Number of PMUs	Total Misplaced PMUs	Number of' Unobserved Buses	Number of Unobservable Regions
0*	0	88	0	114	49
1	1	86	0	115	50
2	2	78	0	135	61
3	3	69	0	158	64
4	4	61	0	188	71
5	5	59	0	196	72
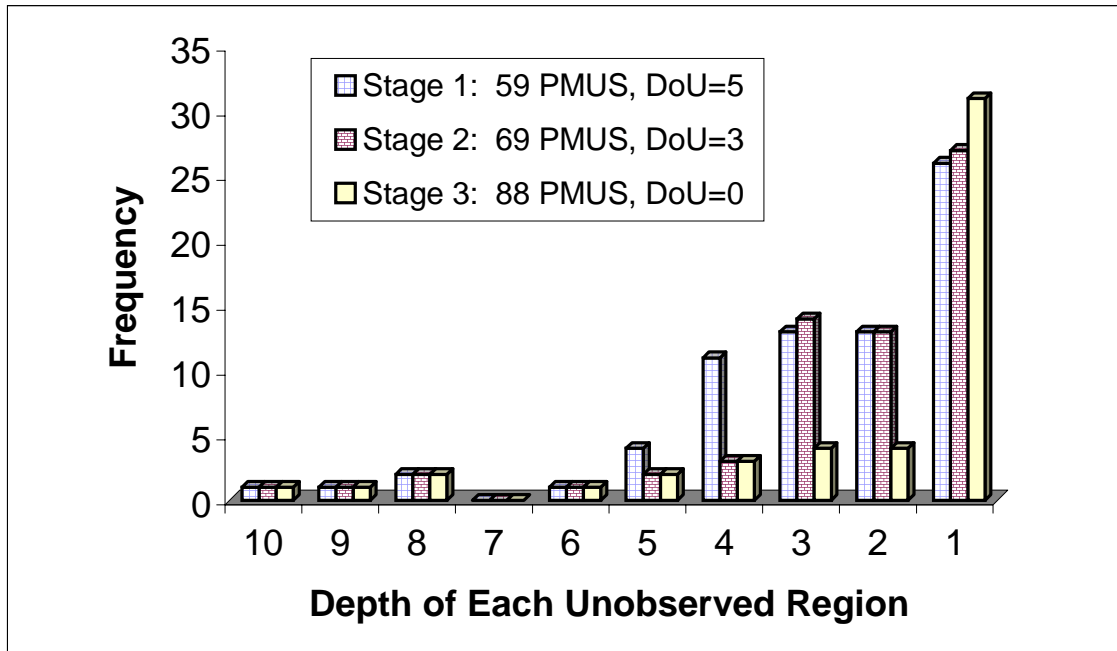
Figure 3-9. Distribution of Size of Unobserved Regions for a 3-Stage Phased Installation
of PMUs for Utility System B

<u>Towards Complete Observability: Identifying Location of New Communication Facilities</u>

To achieve complete observability, new communication facilities must be put in

place together with new PMUs. However, construction of such facilities entails cost

higher than the PMUs themselves. As such, several other cost-effective factors drive the

commissioning of new communication facilities. Nonetheless, we will simplify the

problem by allowing PMUs to completely observe the system as sole justification for the

associated communication. A cost-benefit analysis is outside the scope of this thesis, but

striving for complete observability has its merits. For example, direct state measurement

can now be possible. A PMU-based state estimation will result in more accurate state

estimate. This will then benefit the advanced power applications programs in the control

center's Energy Management Systems.

To identify locations for new communication facilities, we will identify the

optimal placement and number of additional PMUs to make the system observable and

associate communication facilities with this set. We assume that existing PMUs cannot

be replaced or moved to another locations (the phased installation problem). We propose

a solution within the framework of simulated annealing algorithm. The strategy is to place PMUs in all buses U_{S0} left unobservable because of the insufficient communication facilities and subsequently use simulated annealing to remove any redundant PMUs. It is easy to prove that the new candidate set of PMUs will make the system observable since $d(x) = 0$, $\forall x \in U_{S0}$. The communication constraint is then relaxed so that these PMUs are free to move to optimal locations. Mathematically, we first relax the communication constraint and allow SA to search for the optimal positions of the new PMUs. Then, new communication facilities can be provided on the bus locations of the additional PMUs.

The algorithm is formally implemented as follows. First, we classify all PMUs S^0 in the depth-of-zero placement as non-movable. Then, this set of PMUs is augmented by the set of unobserved buses $S^+ = U_{S0}$ to generate the initial strategy S^{init}, that is, $S^{init} = S^0 \cup S^+$. With communication constraint relaxed, the latter candidate placements S^+ are now free to move. As simulated annealing generates the transitions rooted at each bus $x \in S^+$ a redundant PMU in bus x will be eliminated by fusion with another PMU bus y, $y \in S^0$ in the non-movable set, or another PMU bus z in the candidate set, $z \in S^+$. This process will reduce the number of PMUs with observability guaranteed by the penalty term (3.4) in the cost function.

Summary of the results is shown in last row of Table 3-5. New communication and PMUs will be installed at a total of 57 substations. Note that at 'Stage 3' of this table there are a total of 49 unobservable regions; that is, groups of contiguous unobserved buses. It is to be expected that the number of new PMUs required for full observability can never be less than 49 since at least one PMU is needed to observe each unobserved region. The placement of 57 new PMUs and their associated new communication facilities is the optimal solution to the problem.

Table 3-5 shows a proposed migration path towards complete observability of Utility System B. It consists of 4 stages; each stage involves installing a batch of PMUs to move to a lower depth of unobservability. The ultimate stage achieves full observability but only with installation of new communication facilities.

Table 3-5. Proposed PMU Phased Installation Strategy for Utility System B

Stage	Desired Depth of Unobservability	Total Number of PMUs	Number of Unobserved Buses	Number of Buses with Communication	Number of Unobservable Regions
1	5	59	196	165/ Limited	72
2	3	69	158	165/ Limited	64
3	Placement to Observe Maximum # of Buses	88	114	165/ Limited	49
4	Complete Observability	145	0	222/ New Communication Facilities on 57 Buses	0

The phased installation strategy for Utility System B is illustrated in Figures 3-10,3-11 and 3-12. Figure 3-10 illustrates the first stage wherein 59 PMUs will be installed for a desired depth of five unobservability. (PMUs are shown in pink). The second stage requires an additional of 10 PMUs for a desired depth of three unobservability. Finally, stage 3 represents the final stage where 19 more PMUs shall be installed bringing the number of PMUs to 88. The placement strategy is the optimal given the communication constraints; it results in maximum coverage of the study system with the least number of PMUs.
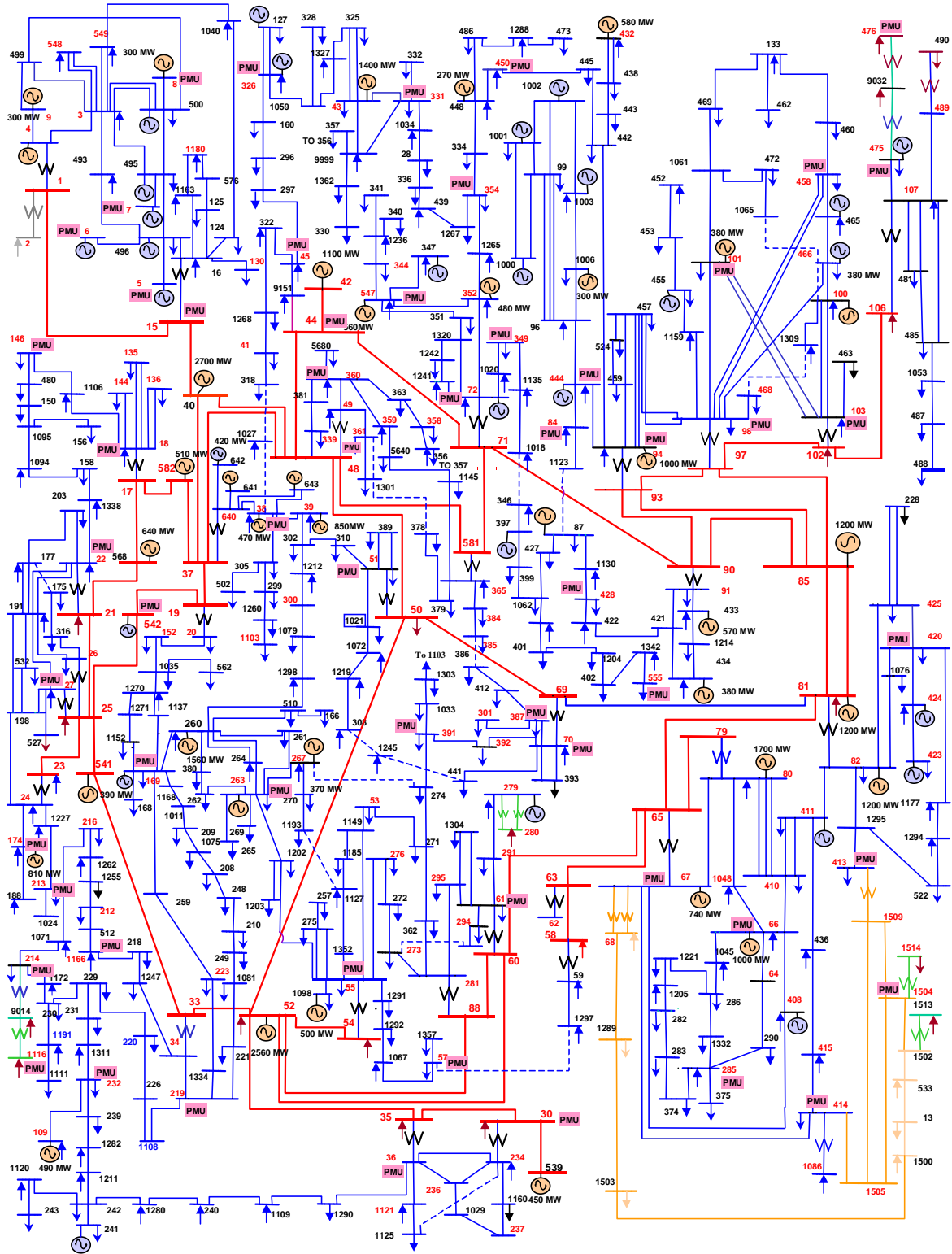
Figure 3-10. Stage 1 of PMU Phased Installation in Utility System B: 59 PMUs
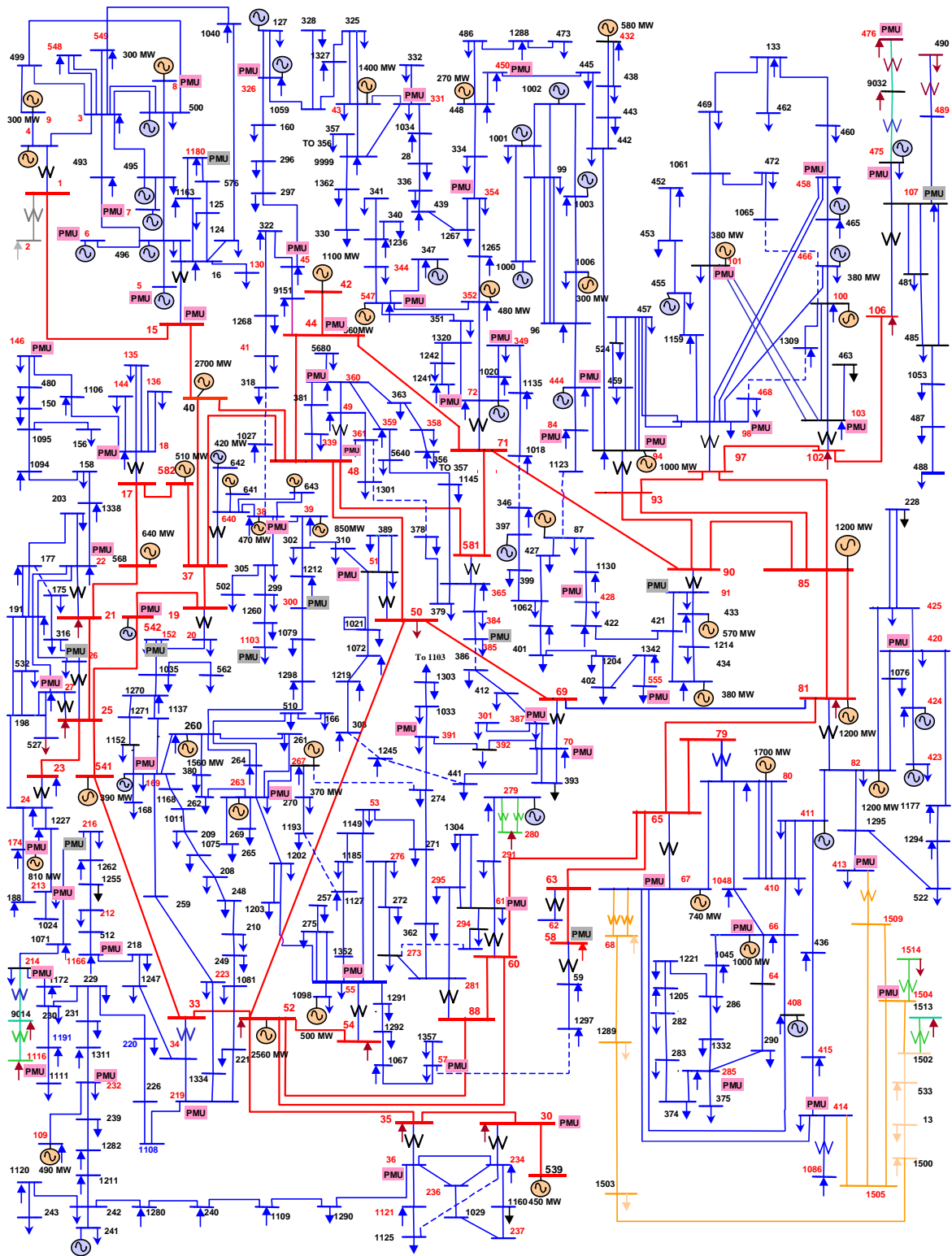
Figure 3-11. Stage 2 of PMU Phased Installation on Utility System B: 69 PMUs
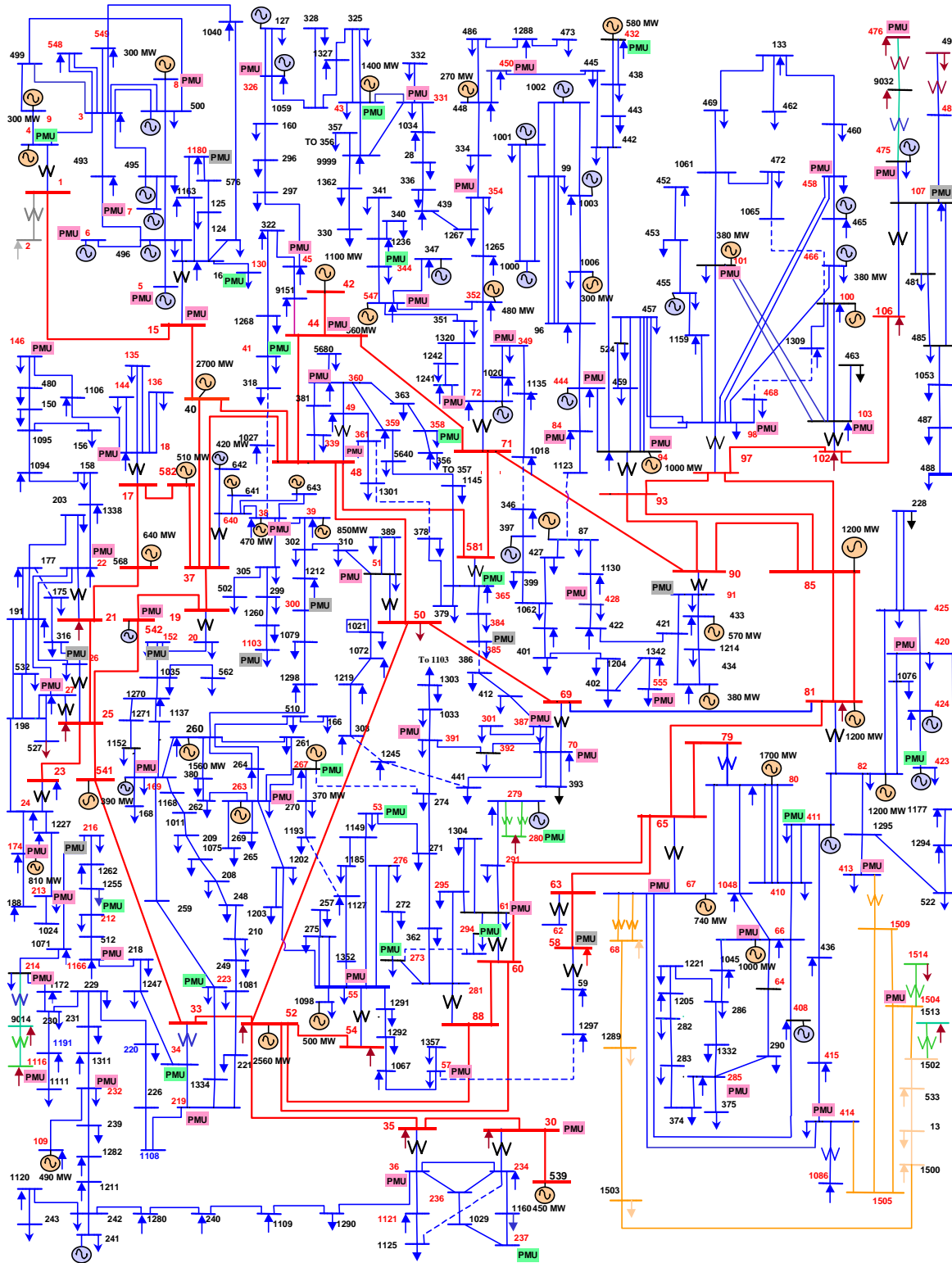
Figure 3-12. Stage 3 of PMU Phased Installation at Utility System B: 88 PMUs

Chapter 4. Voltage Security Monitoring Using Synchronized Phasor Measurements and Decision Trees

This thesis investigates the feasibility of utilizing synchronized phasor measurements to determine the real-time voltage security status of the power system. An algorithm for on-line voltage security monitoring utilizing PMU direct phase angle and/or generator var measurements to quickly determine the voltage security status of a power system. Synchronized phasor measurements can greatly enhance voltage security monitoring through fast, accurate and true real time measurement of the state of the power system.

A steady state power flow model was used to derive indicators of voltage security. Decision trees provide voltage security status given the current operating condition as represented by the PMU measurements. Large numbers of offline power flow simulations were used to generate the database required to build and test the decision trees. A PMU placement technique to increase the reliability of predicting voltage security status will be discussed.

4.1 Introduction

An on-line voltage stability monitoring system should provide three basic functions: 1) contingency selection and screening, 2) voltage security evaluation, and 3) voltage security enhancement [34]. Contingency selection and filtering consists of quickly identifying the contingency or narrowed-down list of contingencies most likely to cause voltage stability problems. This list of contingencies may quite different from those causing transient stability, thermal overload, or voltage decline. Voltage security evaluation is an assessment if the system meets the voltage security criteria when subjected to these contingencies. Voltage security enhancement consists of operator initiated preventive and control actions to create sufficient margin if the voltage security criteria is violated.

Voltage stability indices indicate how far an operating point is from voltage instability. There are two types: state based indices and large deviation indices (see Chapter 9 of [20]). Computationally, state based indices are easier to calculate although large deviation indices provide more precise measures of security margins.

Voltage security and voltage stability literature and books permeates the industry [2][20][40] [65]. Voltage security (VS) is the ability of the power system to maintain voltage stability following one of the lists of "credible events." A credible event could be a line or generator outage, load ramp, or any other event stressing the power system. Voltage security analysis associates the current system conditions to its voltage security status. The system condition is characterized by its state (complex bus voltages) or parameters calculated from it such as real and reactive flows, active injections, etc. VS status could be secure or insecure. If a system is insecure, *preventive action* must be initiated to reinstate voltage security. Preventive action includes capacitor-switching, generator re-dispatching, locking of tap-changer, etc. If the system is secure, useful information is its distance from voltage insecurity. "Distance" here could be in terms of quantities such as total system load, or tie line flows. For example, a MW margin to voltage collapse is the distance in MW of load from the current state to the maximum loadability point assuming a direction of load increase. The direction of load increase is the chosen path on how the system load increases as characterized by the participation of each bus real and reactive load to the system load increase. Clearly, the margin is equally a function of this direction as it is of the current operating point.

When does voltage instability occurs? This fundamental question was addressed using bifurcation analysis (see Chapter 5 of [20]). A dynamical nonlinear system can be characterized by a family of smooth differential equations of the form in (4.1)

$$\dot{x} = f(x, \rho) \tag{4.1}$$

where **x** is the state vector and ρ is the vector of parameters. The equilibrium points of (4.1) for a given parameter ρ are found by setting the left hand side equal to zero, thus

$$0 = f(x, \rho) \tag{4.2}$$

The bifurcation points correspond to loadability limits of the power system. A power system is described in steady state by the algebraic equations:

$$g(u, \rho) = 0 \tag{4.3}$$

At the loadability limit, the Jacobian of the steady state equations of (4.3) is singular. Thus,

$$\det g_u = 0 \tag{4.4}$$

Since at least one eigenvalue is zero in a square matrix, then it can be inferred that at the loadability limit the system Jacobian has a zero eigenvalue.

A family of state-based indices can be derived from the singularity condition of the Jacobian matrix. These are the *determinant, smallest singular value* [42]*, and eigenvalue closest to the origin*. Sensitivity factors form another class of state-based indices. As the loadability limit is approached, sensitivity factors tend to go to infinity. This property results in various types of sensitivity indices proposed in literature [8][26].

Large deviation indices on the other hand searches for the bifurcation points leading to more precise indicators of voltage security. Both the continuation power flow [1] and the point of collapse method [14] belong to this family of indices. Voltage security margins are a direct output of these methods.

Any of the aforementioned voltage stability indicators could be used to determine the real time voltage security status. Speed must be a primary consideration however to leave more time for operators to react. Accuracy is another requirement although it is affected by numerous factors such as 1) direction of stress, 2) initial operating point, etc. The continuation method is the most desirable from the operator's point of view because of the physical significance of the calculated margin. In [23] a curve-fitting method was proposed to speed-up the continuation method to make it attractive for control center application. The margin can also be approximated based on its sensitivity to changes in the parameters of the system [28][29].

Most voltage security indicators are based on complete state information as derived from the traditional state estimator. This thesis proposes direct state measurement using PMUs to increase the speed of the on-line voltage security

monitoring system. We propose an on-line voltage security monitoring system using decision trees (DTs) model that accepts real-time direct measurements from PMUs to predict voltage security status. Both DTs and PMUs satisfy speed requirements to give operators ample time to do preventive or corrective control actions given a potentially risky voltage collapse situation. Bus voltage magnitude was not made part of the measurement set primarily due to inherent bias errors on CCVTs. In time, their effective turns ratio drift due to aging or short-circuiting of capacitor stacks. Another argument against using voltage magnitudes is that they become poor measures of voltage security when reactive support is continuously switched into the system to maintain the voltage profile. Furthermore, PMUs are usually installed at EHV buses where voltage magnitudes are quite insensitive to the loading of the system.

An alternative to voltage measurements is angle measurements. The voltage phase angle is a good indicator of stress or loading in the system [16][24]. Figure 4-1 shows the power-angle (P-δ) bifurcation diagram of a simple radial system used to illustrate the voltage stability problem. Stable operation is on the lower branch of this bifurcation diagram. Note the increasing angle difference between the load and the infinite bus as the load real power parameter P is increased. For large-scale systems, the maximum angle difference mimics this property of the small test system. Figure 4-2 shows the maximum angle difference in the test subsystem as a function of the total real power load. The quadratic characteristic of the angle near the maximum loadability point is observed. The more stressed double outage contingency case results in greater angle difference and lower limit on maximum loadability compared to the no-outage base case condition.
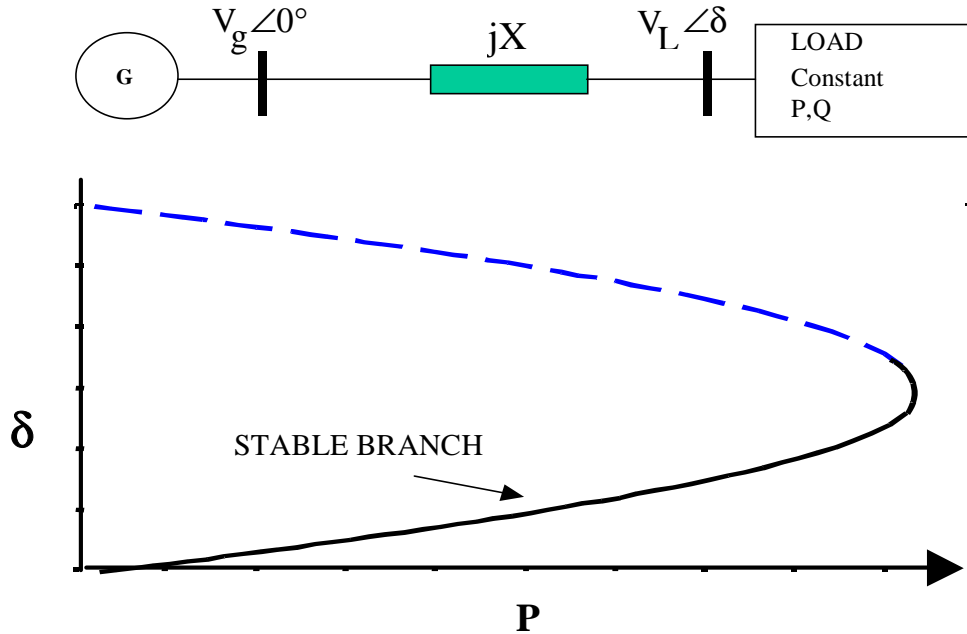
Figure 4-1. Power-Angle Bifurcation Diagram

PMUs output phase angle information of both bus voltages and line currents. The reference angle is that angle associated with the timing signal. To have a physical meaning, this angle measurement must be referred to a physically existing reference measurement point (the load flow slack bus, for example). However, this set-up would be unreliable because the angle difference measurement set is rendered useless whenever there is failure of the referencing PMU. We decided then that angle difference measurements between PMUs constitute the measurement set for the decision tree.

The study system is a 360-bus problem area illustrated in Figure 4-3. The voltage levels are 765 kV, 345 kV, and 138 kV. Not all 138 kV buses are shown.
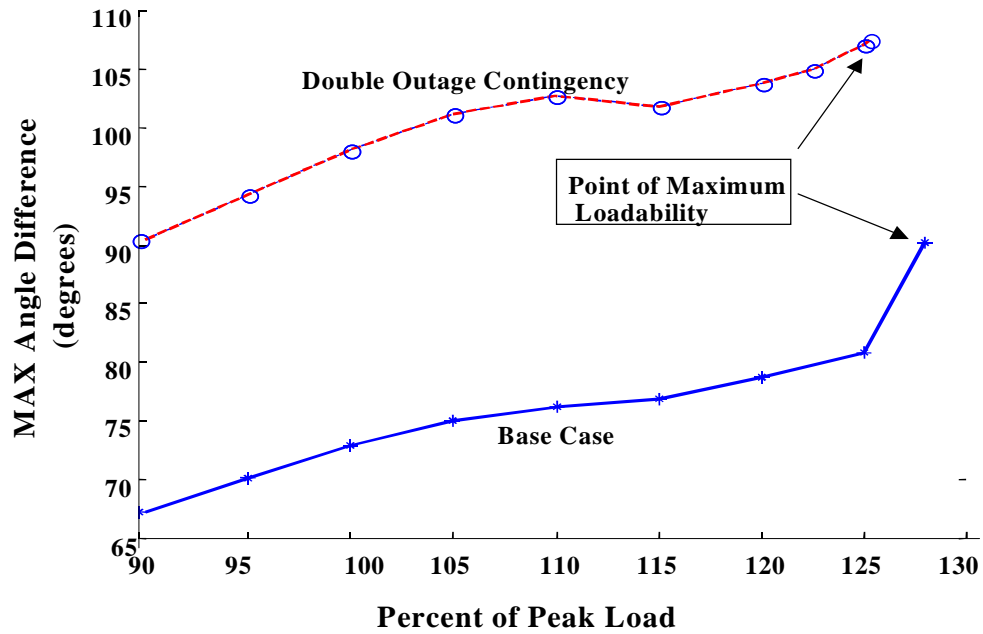
Figure 4-2. Loading-Maximum Angle Difference Diagram of the Study System
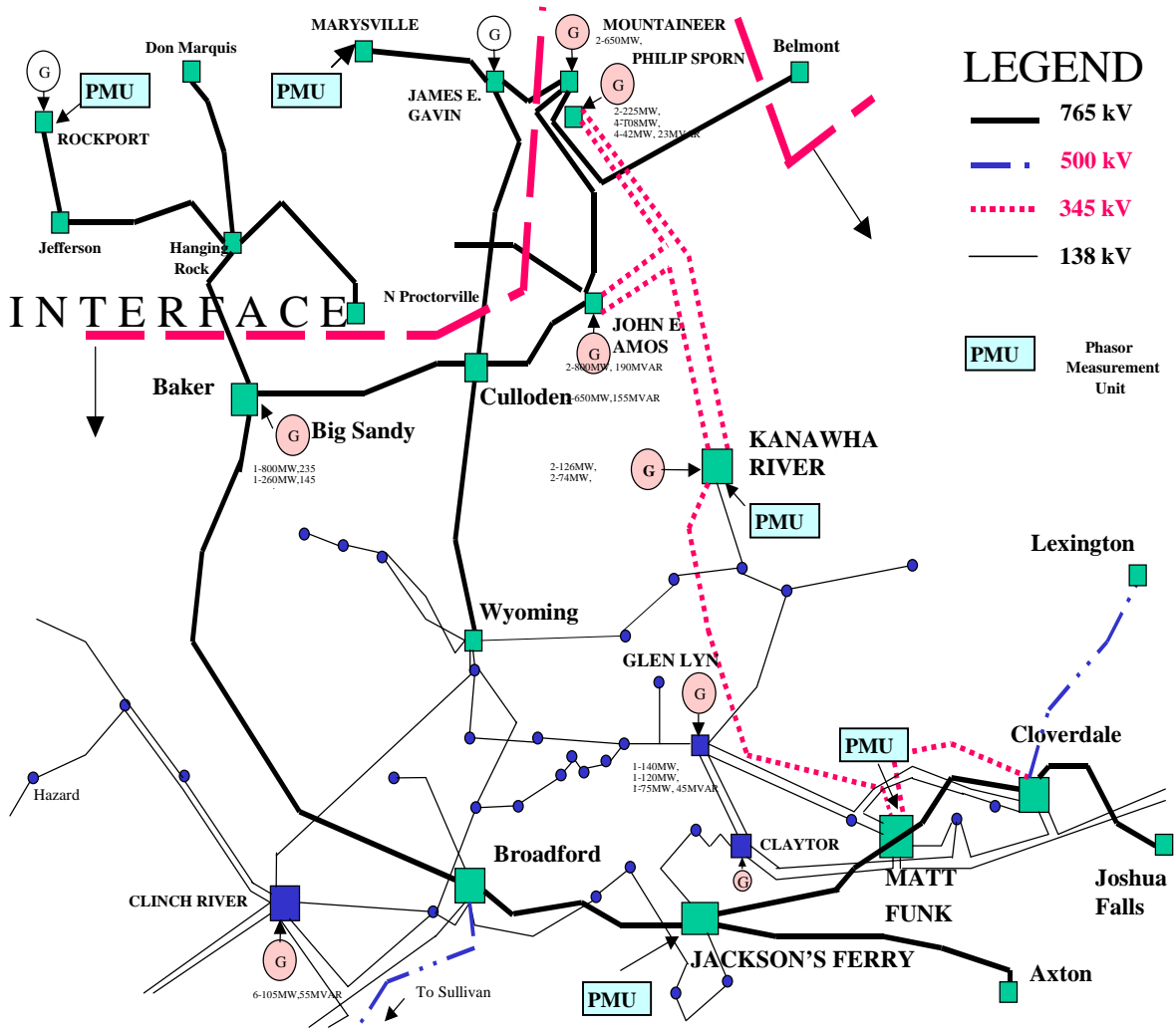
Figure 4-3. Study Region for Voltage Security Monitoring

4.2 Voltage Security Monitoring Using Decision Trees

Decision trees (DT) models were chosen to carry out the classification function of the on-line voltage security monitoring system. Decision trees belong to a class of "Automatic Learning" techniques used to derive information from database. For exhaustive treatment of decision trees as applied to power system security assessment see [66]. A practical application of DTs for voltage security assessment is explored in [19]. A decision tree partitions the input space of a data set into mutually exclusive regions, each region being assigned a label or a value. Within the context of voltage security assessment, we can have two types of trees:

1. Classification type – classifies an operating point as "secure" or "insecure" based on a pre-defined voltage security criterion; and

2. Regression type – associates an operating point with margin to collapse.

Figure 4-4 shows a classification type DT for voltage security monitoring. A real-time measurement vector is dropped down the root node of the tree. Each node in the tree is associated with a dichotomous test, **Is $x(a) \leq s(a)$?,** on one measurement variable (henceforth, called an attribute). Depending on the result of this test the measurement vector will "drop" down into one child node, wherein another test is made, and so on. The classification process stops when the measurement vector settles into a terminal node. A terminal node has a voltage security class pre-assigned to it during the tree-building process. The voltage security class of the terminal node (where the real-time measurement vector will finally lie) will then be the voltage security class of the current operating condition as characterized by the measurement vector.

The decision tree is built in a top-down fashion from the root node to the terminal nodes using sample training data or measurement space consisting of N cases, L={(x_1, j1),, (x_N, j_N)}. Each x_i is an M-dimensional measurement vector with a pre-assigned class j_i. The measurement space associated with each node t is searched for that attribute which when split (resulting in the dichotomous test) will result in the most "class-pure" segregation of that measurement space. This is accomplished by determining the optimal splits s_a* for each attribute a, and then choosing that optimal attribute a* which results in greatest reduction in the impurity of node t (see Appendix C). Splitting is terminated on

a node if it is class pure; that is, all measurement vectors arriving at that node belong to the same class. The whole process of node splitting is repeated until such point when: 1) all terminal nodes are class-pure or 2) the maximum misclassification rate of all terminal nodes is lower than a pre-defined bound. The reliability of a decision tree is a measure of the robustness of its classification ability on a separate test data set. This reliability is expressed as the ratio of misclassified cases to the total number of test cases submitted to the decision tree.

A method exists to find the "right-size" tree [9][35], which is that DT with the highest reliability. It involves generating the maximum possible sized tree during the node splitting process, and then pruning back this big tree by eliminating its weakest links (see Appendix C). Decision trees are fast and efficient models to the voltage security monitoring function. Only rules or tests arranged in a binary tree like fashion are used and there are no floating-point calculations. Decision trees blend naturally with fast measuring capability of PMUs.
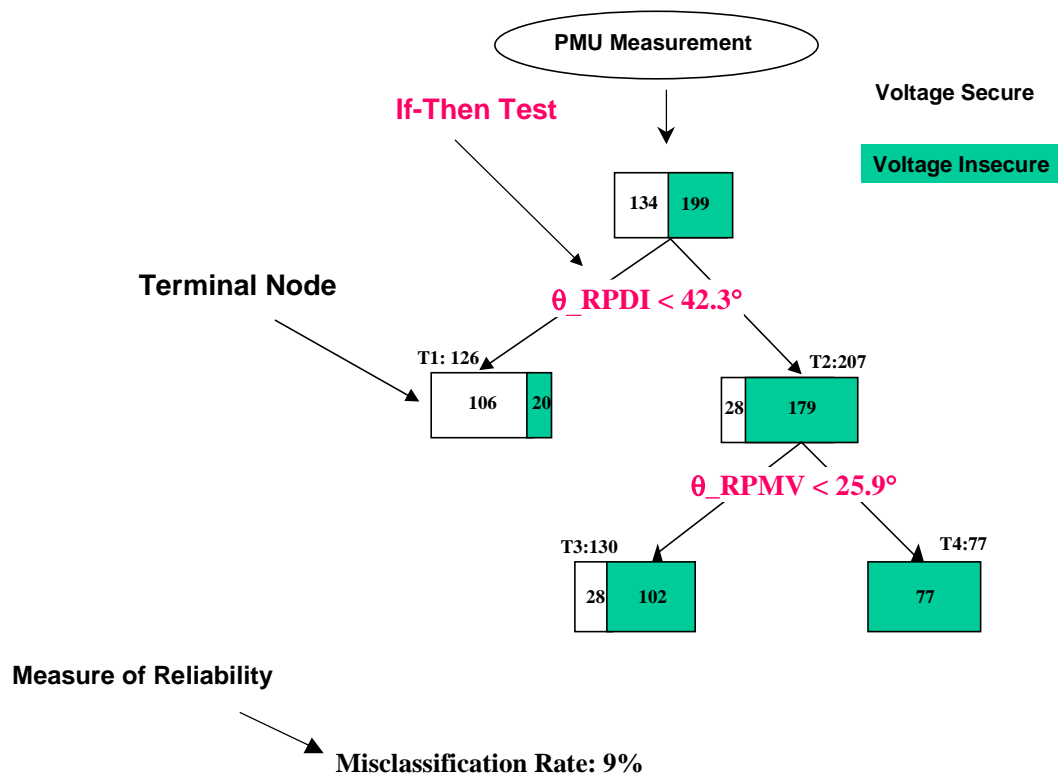
Figure 4-4. Classification-Type Decision Tree for Voltage Security Monitoring

4.3 Data Generation

Decision trees require training points encompassing realistic operating conditions, in terms of load, generation mix, and outages. The credible outages encompass single (n-1) and double (n-2) outage contingencies on 765 kV, 500 kV, and 345 kV systems. The 500 kV line outages are externally located from the study region. Our load flow model has 6000+ buses and covers the neighboring systems as well. The contingency list also includes single unit outage of the selected power plantsThe next step is to assign a margin to voltage collapse for each of this operating point. Here we have adopted the theory linking voltage collapse to saddle-node bifurcation [20][13]. At the saddle point, the system loses a stable operating point and the ensuing dynamics leads to voltage collapse. We used a power flow model and the method described in [25] to calculate the critical point (the knee) of the P-V curve. Voltage instability caused by Hopf bifurcation and fast transient due to inductive motor loads are not captured by this method and are the subject of future work.

With regards to the direction of load increase, we assumed a uniform participation of each of the load buses on the total load increase; load power factor maintained constant. Thus, if we specify a loading factor λ to get a corresponding real power load increase $\lambda * P_{so}$ from the base case real power P_{so}, each bus j's load will increase according to (4.5) and (4.6).

$$P_{Lj} = P_{LjO} + P_{NLj} * \lambda * P_{SO} \tag{4.5}$$

$$Q_{Lj} = Q_{LjO} + Q_{NLj} * \lambda * P_{SO} \tag{4.6}$$

Here P_{NLj} and Q_{NLj} are load distribution factors, with $P_{NLj}=P_{Lj0}/\sum P_{Lj0}$ and $Q_{NL}=P_{NLj} \tan \theta_j$. The *direction of generation increase* is expressed in terms of generator pick-up factors DP_{Gi} as defined by (4.7). Each generator will pick up load based on its installed capacity with maximum generating capacity check enabled. Generators inside the subsystem rank high in the merit order, followed by external generators if available capacity is exhausted.

$$P_{Gi} = P_{Gi0} + DP_{Gi} * \lambda * P_{SO} \tag{4.7}$$

Figure 4-4 graphically demonstrates how to detect the nose of the PV curve using power flow simulations. Point A represents the current state of the system without contingencies. The user will stress the system by increasing its real/reactive power demand, which will bring the next operating point to B. This process is continued, each time saving the solved loadflow case, until point O is reached (the nose). The distance A-O is the margin for point A; similarly the distance B-O is the margin for point B, and so on.

The lower dashed curve represents an outage condition occurring at the state A. Thus, if a contingency happens at A, the resulting PV curve will have a nose at point O'. To generate this curve, retrieve the solved load flow cases at points B, C, etc. and initiate the same outage. The nose of this curve occurs at point O' which corresponds to point C at the upper PV curve. The margin to collapse for this outage condition, given the current operating point A, is then the distance A-C.
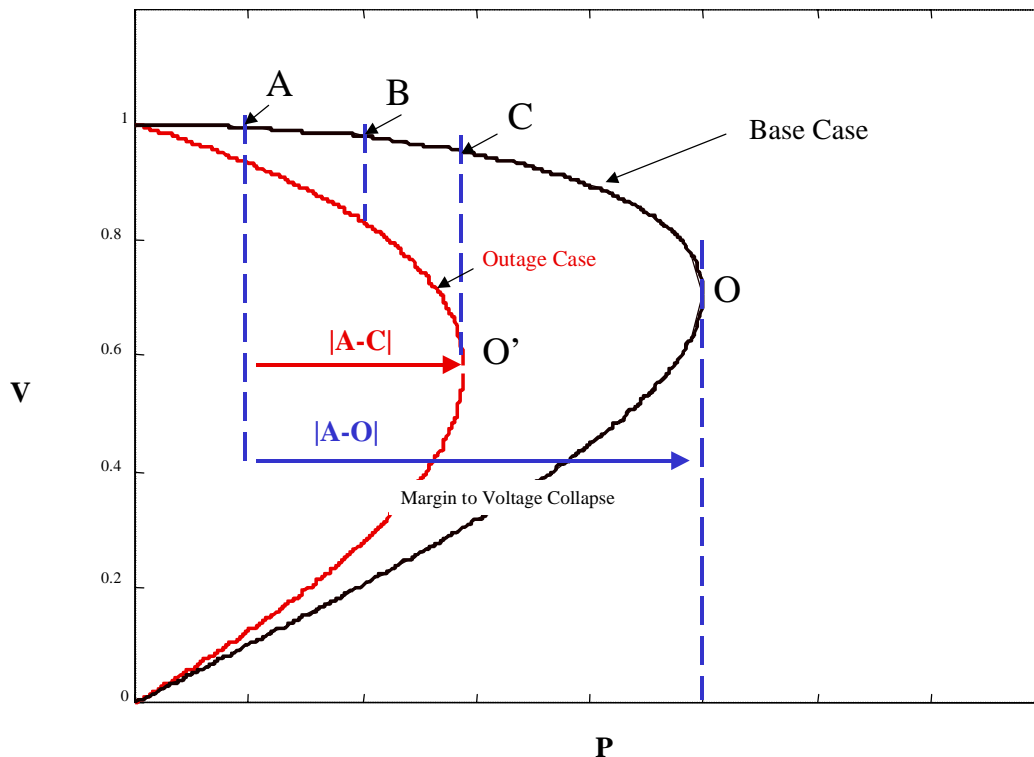
Figure 4-5. Determining the Nose of the PV Curve Using Successive Power Flow
Simulations

4.4 Voltage Security Criterion

Determining the voltage security criterion is another important aspect. This is equivalent to assigning a class (secure or insecure) to an operating condition. There is not a universal approach to voltage security classification however. Nonetheless, we impose the following conditions that must be met for the study system to be voltage secure:

1. A minimum voltage > 0.92 per unit in the study system.
2. Minimum operating voltages at identified power plants.
3. A margin to voltage collapse of 12.5% of base case peak load.

Item 1 above simply imposes a minimum acceptable operating voltage in the study system. Item 2 is a product of previous operator's experience wherein it was observed that in cases when voltages fell below these values the units in question trip due to low voltage on their auxiliary system. Note that bus voltage magnitude information is derived from power flow simulation and will only be used to classify voltage security status. Item 3 is the subject of further study. Margin to voltage collapse simply is the additional amount of real power load that can be carried by the system before collapsing. From the operator point of view, load growth is closely monitored together with load ramping rate. Thus, the total load growth during the ramping period can be estimated. If this change in load exceeds the margin to voltage collapse of the system, then the current operating condition must be classified as voltage insecure so that preventive actions can be done to avert possible collapse conditions.

It then becomes a question of how fast the operator will realize these actions. When the operator reaction time is ascertained and assuming the preventive actions are effective, a time margin can be added to this reaction time, which together with the ramping rate will give the estimated load growth from the time the operator detects the problem to the time it is corrected. This load growth can then be taken as the voltage security criterion.

4.5 Numerical Results

A. Existing PMUs

Figure 4-6 shows a DT classifier built from 11 angular difference attributes (10 ($_5C_2$)
angular differences and 1 maximum angular difference) corresponding to the five
existing PMUs. The misclassification rate is 18% on the test data. The tree was the
optimal from a set of 16 candidate "pruned trees" taken from a 77-node maximum tree.
Figure 4-7 shows plots of the misclassification rates of each pruned tree (indexed by the
number of terminal nodes) in the pruning process. Note that for the training data the
maximum tree gave the best classification, however, using test data the most reliable tree
is smaller. Note the location of the "right-size" tree. The tree in Figure 4-6 has 25 nodes -
12 internal or test nodes and 13 terminal nodes. Following the convention in [66], the
number on top of each node corresponds to the number of training cases falling into it.
Each node is shaded in proportion to the voltage secure (light shade) and insecure
(shaded) cases into the node. Any terminal node that is not class pure implies there was
pruning of its descendant nodes. The most relevant attribute identified by this tree is the
angular difference between two PMUs situated near the boundary of the study system.
We observe that the angles of these 2 PMUs are less sensitive to the line outages. Line
outages create shifts in angles on neighboring buses and this skew has to be accepted. For
example, a line outage case far from collapse may result in higher angular difference
among the neighboring buses compared to a non-outage case very close to collapse.
Figure 4-8 shows the DT input space partitioned by the two most relevant attribute tests.
In terms of the decision tree in Figure 4-6, L1 divides the input space according to the test
at the root node. Note that all points above L1 are insecure, they are the set of points
going to node T2 of Figure 4-6. Line L2, which corresponds to the test of attribute 2 at
node T1, further divides the remaining portion of the input space (points below L1) into
two separate groups. Each is as "class pure" as possible, although no class assignment is
done on this node.
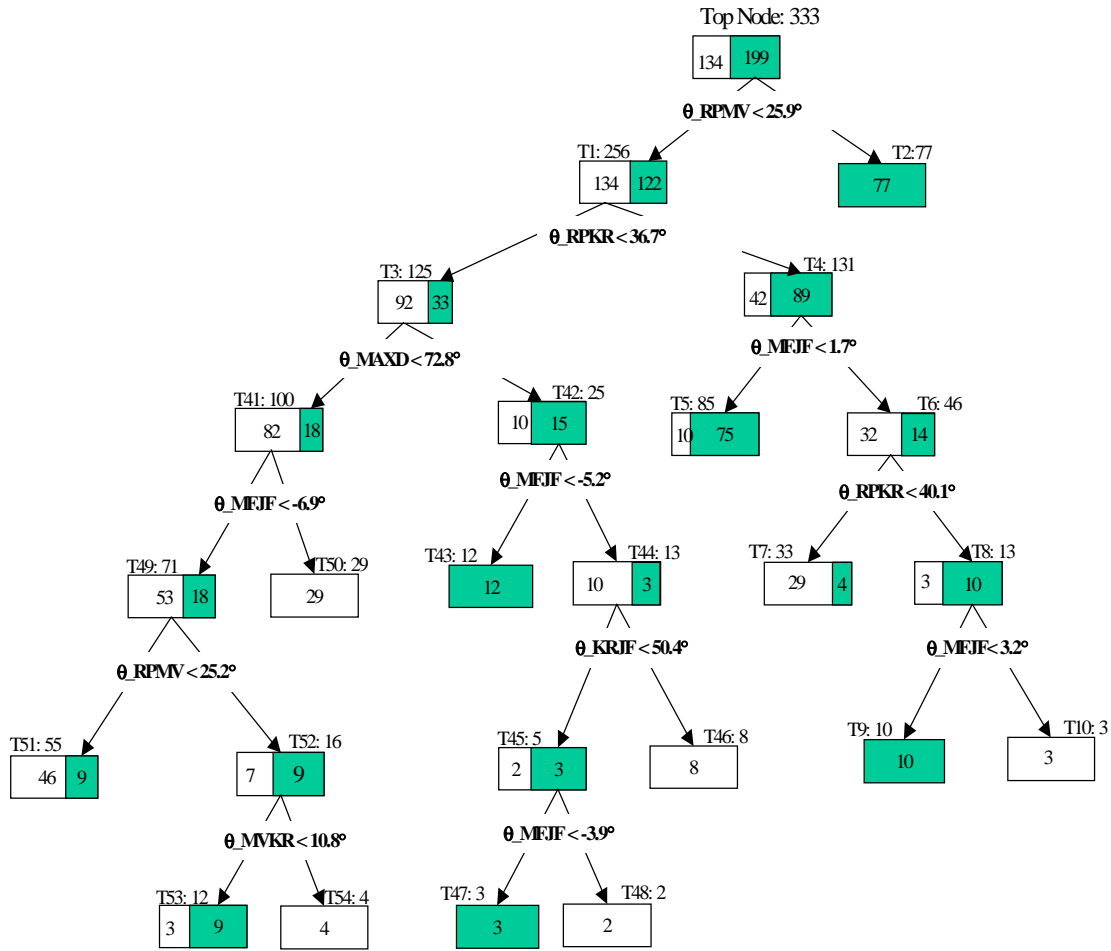
Misclassification Rate: 18%

Figure 4-6. Decision Tree for Voltage Security Assessment Using Existing PMUs: Angle Difference Attributes
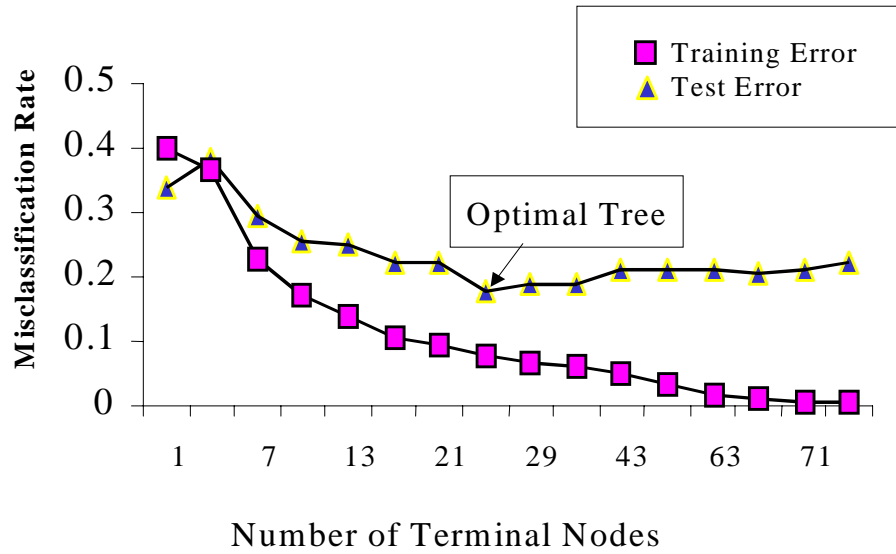
Figure 4-7. Misclassification Rate with Respect to Tree Size
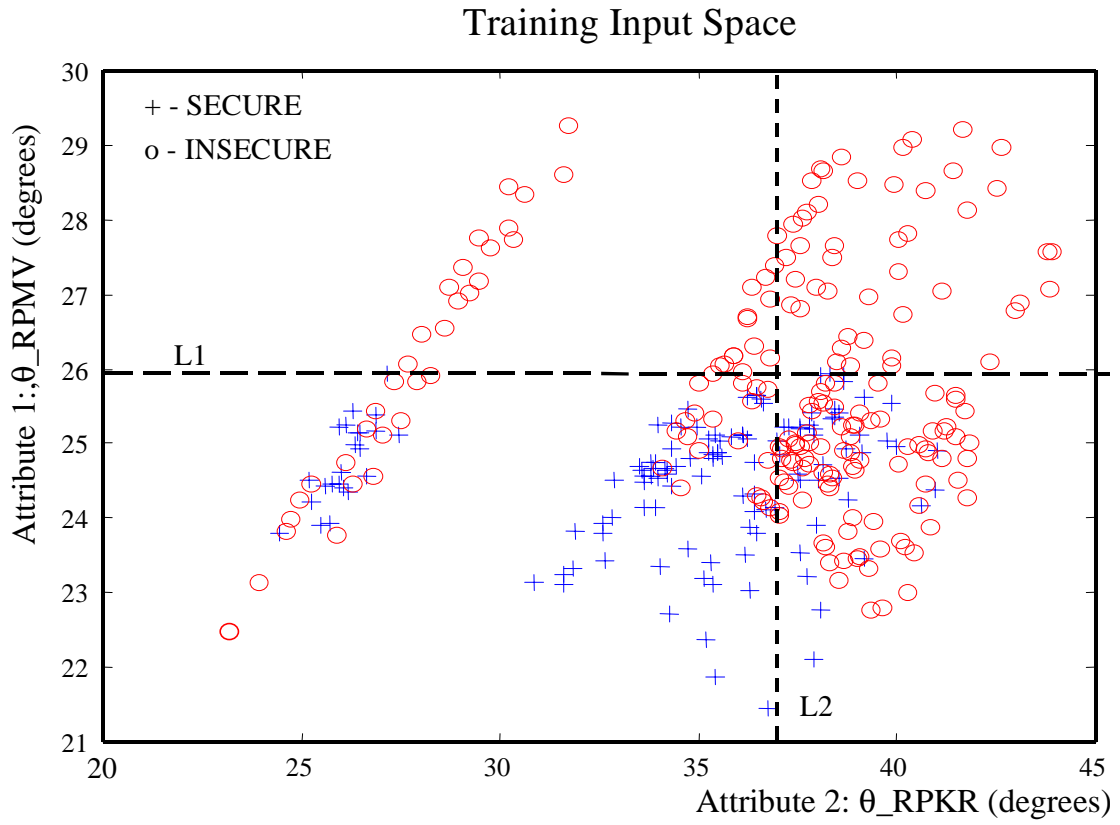
Figure 4-8. Input Space Partitioning by Decision Tree

B. Additional PMU Placement for Improved Monitoring

The objective here is to find one other PMU location that will increase the reliability of classification. We have taken the approach of adding angle information from a new PMU bus resulting in 6 additional angular difference attributes (maximum angle difference included). This is done for each bus in the study system. A decision tree is built for each of these systems of 6 PMUs with misclassification rate calculated each time. Figure 4- 9 shows the distribution of the calculated reliabilities. Not every new PMU location resulted in better classification, but most candidate PMUs increase the overall accuracy. A total of 32 new PMU locations achieved 10% or better misclassification rates. It is important that the tree can be easily interpreted, and a compromise is made between accuracy and complexity, complexity being defined as the number of nodes. Figure 4-10 shows a possible DT classifier with one new PMU location at a 138 kV bus, two buses away from a 765 kV substation. The misclassification rate is 9% for a size of 13 nodes. The best calculated classification is 4.8% from a new PMU location at a 765 kV bus. However, the resulting tree with its 33 nodes is too complex to be drawn here.
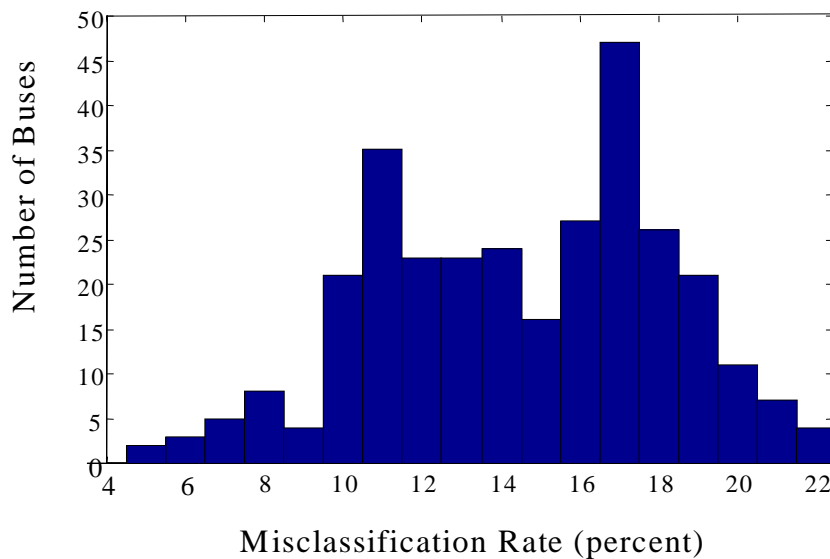
Figure 4- 9. Distribution of Misclassification Rates with One New PMU

C. Using Generator Vars

Var generation is another critical indicator of voltage security. This information can be accessed using new PMUs installed at critical power stations or from the Energy Management System inside the control center. The latter can be accomplished at the expense of speed. In building the decision trees, we used ten pre-selected var attributes all coming from generators inside the study system: nine from var outputs of major power plants and one total Mvar attribute. Figure 4-11 shows the decision tree with 13 nodes using var generation. The misclassification rate of 10.8% is slightly less accurate than the angular difference DT (with one new PMU) of Figure 4-10. Of the 10 attributes, this DT selected just 5 generator vars as relevant.

Table 4-1 presents a summary of the optimal decision trees resulting from each of the design options considered. The results suggest that adding one more PMU to the existing 5 would improve the reliability of classification of voltage security significantly. The table also suggests that the angle attribute fared at par with a more widely accepted attribute - the generator vars- in detecting voltage insecurity.

Table 4-1. Comparison of Classification Type Decision Trees for Voltage Security Monitoring

DT ID	Misclassification Rate	Number of PMUs	Complexity (# of nodes)	Attribute
1	**18.0%**	5 (Existing)	25	Angle
2	**9.0%**	6 (One Additional)	13	Angle
3	**10.8%**	5 (All new PMUs)	13	Generator VAR
4	**4.8%** **Most Reliable**	6 (One Additional)	33 (Most Complex)	Angle

Top Node: 333

θ_RPDI < 42.3°

T1: 126

T2:207

θ_RPMV < 25.9°

T3:130

T4:77

θ_RPDI < 55.2°

T5: 88

T6: 42

θ_MFDI < -9.68°

T7: 77

T8: 11

θ_MFMV < -41.9°

T9: 29

T10: 48

θ_MRKR < 13.9°

T11: 43

T12: 5
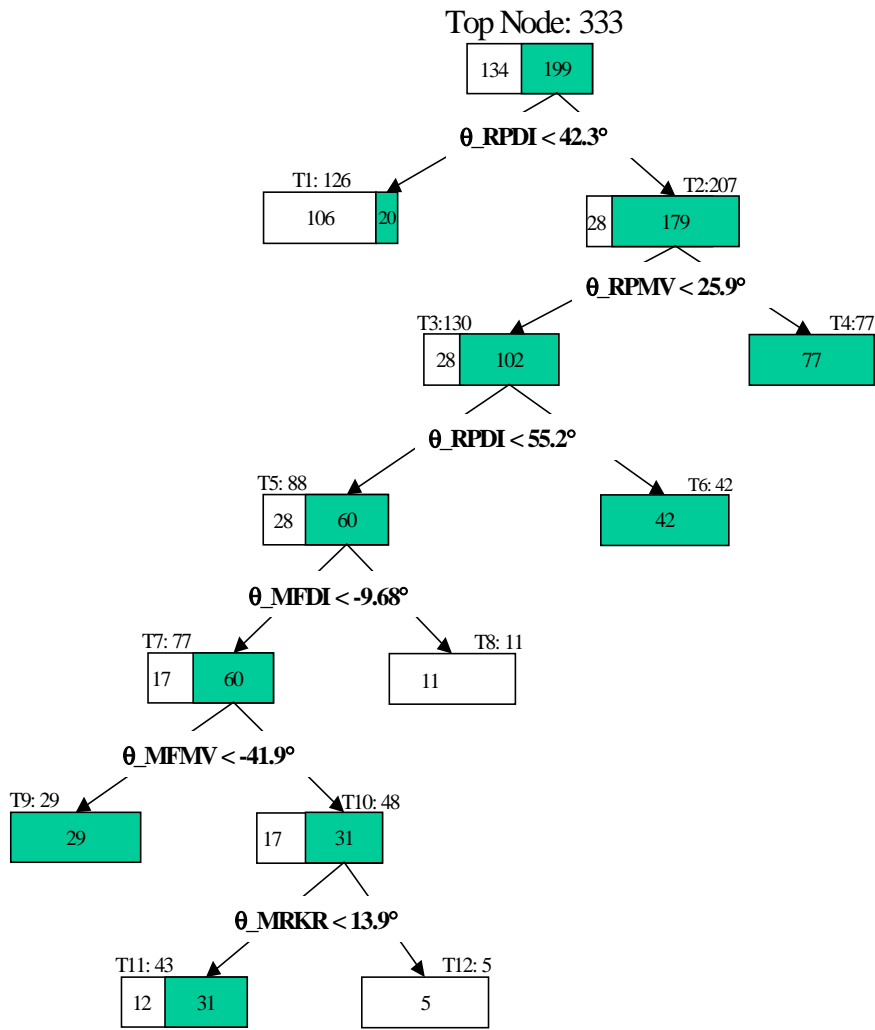
Misclassification Rate: 9%

Figure 4-10. Decision Tree with One New PMU: Angle Difference Attributes

Top Node: 333

134	199

Q_CLNCHR < 177

T1: 150

122	28

T2:183

12	171

Q_SPORNB < 111.2

T15:89

87	2

T16: 61

35	26

Q_AMOS765 < -117.5

T17: 11

11

T18: 50

24	26

Q_BSAND < 40

T19: 44

24	20

T20: 6

6

Q_MOUNT < 31.8

T21: 44

24	20

T22: 12

11	1

Q_SPORNB < 122.5

T25: 12

11	1

T26: 17

2	15

Misclassification Rate: 10.8%
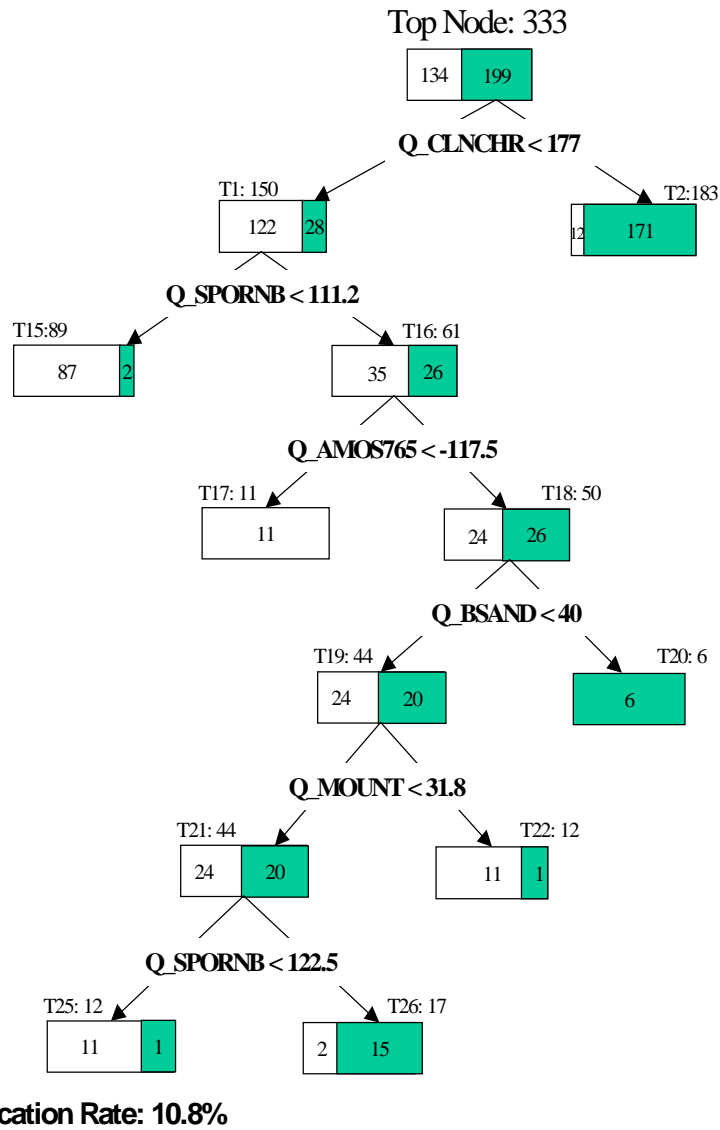
Figure 4-11. Decision Tree Using Generator Var Attributes

4.6 Conclusion

We presented a proposed on-line voltage security monitoring system using phasor measurements and decision trees. Its functional representation is illustrated in Figure 4-12. Using phasor measurement units offer the distinct advantages of speed, accuracy, and synchronized sampling of bus voltage angles and var generation. Quick voltage security classification by decision trees blends with the fast measuring capability of PMUs.

We have introduced the concept of using bus voltage angle differences as indicator of voltage security. Based on our simulation, there is a general pattern of increasing angular separation as loading or line outage contingencies stress the system. The stress is associated with decreasing margins to voltage collapse. The pattern is non-linear and decision trees are effective tools in extracting the complex relationship between the angle differences and voltage security class. We have also used decision trees to identify the critical location of another PMU for improved accuracy of classification.

A separate DT was built using var generation attributes from selected power plants inside the study system. Numerical results on the test data showed that angle difference attributes performed better classification than var attributes. This observation depends largely on the generated database, but we expect this result to be sustained, as long as the new PMUs are placed at critical monitoring buses.

The proposed voltage security monitoring system shows great promise. Further enhancements can be realized by: (a) generating a larger number of off-line data base, (b) using dynamic simulations to further screen insecure cases that are otherwise loadflow solvable, and (c) incorporating several directions of load increase such as the "shortest distance" to collapse.
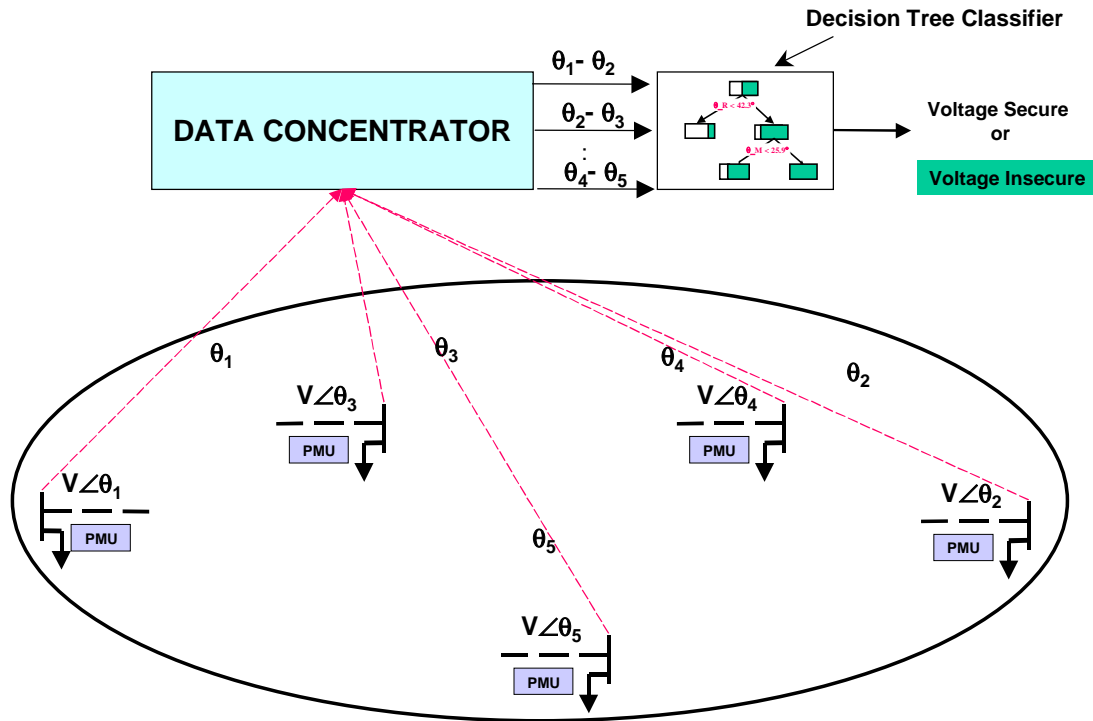
Figure 4-12. Functional Representation of Proposed Voltage Security Monitoring
System

Chapter 5. State Estimation Using Synchronized Phasor Measurements

In this chapter, several models for state estimation using synchronized phasor measurements will be presented. The basic premise is that the power system is not fully observable with the PMUs.

We start by revisiting the WLS state estimation algorithm and devise a model wherein PMU measurements could enhance the accuracy of the state estimate (Section 5.2). PMU measurements are utilized much like the conventional remote terminal units (RTUs) but the main difference is that here PMU voltage and currents are not part of the WLS measurement set. Rather, they are utilized after WLS has calculated the state. Our model for the enhanced WLS algorithm is non-invasive in the sense that PMU measurements are not direct inputs to the WLS.

A technique for interpolating the voltages of the unobserved buses based on the bus admittance matrix will be developed and presented (Section 5.3). Interpolation errors shall be analyzed vis-à-vis the number of PMU placements. We refer to the interpolation coefficients as the linear state estimators (LSEs). A model for updating these LSEs using sensitivity factors will be presented. The goal is to achieve a more accurate estimation of the unobserved buses. The accuracy of the model will be tested on a real world utility system and will involve comparison of estimated power flows to the ideal flows.

Finally, a hybrid state estimator using the classical WLS estimator and the linear PMU based estimator will be developed (Section 5.4). This is a streaming-type state estimator and is designed primarily to utilize the estimate from the traditional WLS SE to update the LSEs. This model addresses the need of utilities to implement PMU-based SE alongside their traditional SE. This model includes effects of transducer errors on both PMU measurements and RTU measurements.

5.1 Introduction

We anticipate a gradual migration towards full PMU implementation for utility power systems. The logical direction is towards full observability with PMUs. Due to

cost considerations, it is expected that utilities will execute a phased installation of PMUs, that is, batches of PMUs being installed through time. As discussed in Chapters 2 and 3, phased installation can be systematically accomplished using the concept of incomplete observability wherein PMUs are installed so as to lower the depth of unobservability. The technique implied that existing PMUs must be subset of the ultimate set required for observability.

Utilities will expect advanced applications attached with the PMUs. In the past, a number of PMU alpha site installations were made for the purpose of gaining experience, R&D, or post mortem analysis of disturbances [11][57][12]. Applications on adaptive protection [15] and state estimation enhancements [62] although small scale and time limited confirmed the validity of deploying PMUs for network applications.

State estimation provides the platform for advanced security monitoring applications in control centers. For example, contingency analysis and optimal power flow rely on the state of the power system derived from the state estimator. For this reason, state estimation is perhaps the most important PMU application. Also, the positive sequence voltage and line current phasor output from PMUs are consistent with the positive sequence network model in state estimation. It should be expected that communication links might not be sufficient to fully observe the network. The challenge therefore is to develop a functioning state estimator with synchronized phasor measurements when the system is not yet observable by PMUs.

Phadke et al [52] is to be credited with the seminal paper introducing the theory behind a linear state estimator utilizing synchronized phasor measurements. The revolutionary idea stems from making direct state measurement and involving state estimation to improve on the accuracy of the state. PMUs were assumed to be located in every bus; variations in the number of line current measurements were considered in testing the model. The measurement vector is shown in (5.1) where E_B and I_L are the vectors of true values of complex bus voltages and line currents and ε_B and ε_L are the corresponding measurement error vectors respectively.

$$M = \begin{bmatrix} E_B \\ I_L \end{bmatrix} + \begin{bmatrix} \varepsilon_B \\ \varepsilon_L \end{bmatrix} \qquad\qquad (5.1)$$

By expressing the true line current vector I_L in terms of true bus voltage vector E_B, a transformation of (5.1) into (5.2) results, which now expresses the measurement in terms of the state of the system E_B. Here y is the diagonal matrix of primitive line series admittances, A is the current measurement to bus incidence matrix, and y_S is the diagonal matrix of primitive admittances at the metered end.

$$M = \begin{bmatrix} \mathbf{I} \\ yA^t + y_S \end{bmatrix} E_B + \begin{bmatrix} \varepsilon_B \\ \varepsilon_L \end{bmatrix}$$

$$= BE_B + \varepsilon \tag{5.2}$$

The measurement errors were assumed to be un-correlated with zero mean defined by the diagonal covariance W. The weighted least square solution for E_B shown in (5.3) is linear with the gain matrix $G = B^*.W.B$.

$$G \cdot E_B = B^* \cdot W \cdot B \tag{5.3}$$

Being linear, the state can be directly calculated in a single matrix computation. G is factorized and stored in compact form and needs to be updated only after occurrence of significant topology changes in the system. The linearity complements the fast measuring capability of PMUs.
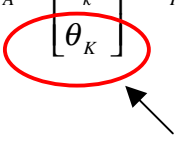
Other state estimation models were based on integrating the PMU angle measurement in the traditional measurement set [53] [62] resulting in a balanced set of measurement equations (5.4)(5.5). The state of the system here is still the average over the traditional SE scan period however. Originally, there was the skew in the measurement equations because the angle could not be measured. PMUs directly measure the angle although with respect to the timing reference of the GPS. A detailed analysis of the effects of angle measurement errors in the quality of solution can be found in [62].

$$z = \begin{bmatrix} z_A \\ z_R \end{bmatrix} \longleftarrow \text{measurement vector} \tag{5.4}$$

where

$$z_A = \begin{bmatrix} P_{km} \\ P_k \\ \theta_K \end{bmatrix} \quad z_R = \begin{bmatrix} Q_{km} \\ Q_k \\ V_K \end{bmatrix}$$ (5.5)

direct angle measurement with PMU

Aligning the angle reference between the traditional state estimator and the PMUs is a fundamental problem that needs to be resolved in this class of algorithms. A direct approach could be to install one PMU in the reference bus of the traditional SE. It was realized that this arrangement was unreliable because the angle measurements become invalid when the reference angle becomes unavailable for reasons such as communication failure. The angle reference was solved in a partitioned state estimator proposed in [38]. Here the authors envisioned using a linear PMU-based SE on the low voltage side of the system and traditional weighted least square (WLS) SE on the high voltage side. On the boundary buses the phasor measurements were assumed to be available both estimators. In addition the WLS estimator also calculates the angles at the boundary as referred to one of the high voltage buses. Therefore, the angle difference between the two references can be found by averaging the difference between these two angles. That is,

$$\phi = \frac{1}{NB} \sum_{i=1}^{NB} (\theta_i^{WLS} - \theta_i^{LSE})$$ (5.6)

Traditional State Estimation

The traditional state estimator will be used to update the interpolation coefficients in the hybrid state estimator. It is logical at this point to present a brief review of this method that has since its inception have grown to be the most important function in the real time monitoring of today's power systems.

State estimation is considered perhaps the most important control center function. It provides the information necessary to execute such advanced control computer programs on security analysis, on line load flow, etc. The book by Monticelli [46] provides a comprehensive treatment of basically all theoretical aspects of state

estimation. Wood and Wollenberg [67] devoted a whole chapter on this very important topic.

State estimation grew out of need to accurately estimate the state of the power system in real time. Before this technology, the only way operators can monitor the system is through a load flow simulation. However, errors in the measurement were not explicitly taken into account resulting in gross errors in the state estimate. It was recognized that a different model must be in place that will recognize the errors in the measurements.

Schweppe is to be credited for significant fundamental contributions to the theory behind the AC static state estimator [58][59][60][61]. The qualifying term 'static' signifies that the estimator will not provide the true dynamic state since the measurements are not collected simultaneously. The measurements are collected every scan period resulting in a state estimate that is static or pseudo-steady state. The measurement model is an expression relating the measurement vector z with the state x of the system given the meter error ε.

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{\varepsilon} \tag{5.7}$$

The errors in the measurements are assumed to have zero-mean, uncorrelated as modeled by the diagonal covariance matrix R.

$$\mathbf{R} = E\{\mathbf{\varepsilon}^{\mathbf{T}}\mathbf{\varepsilon}\} \tag{5.8}$$

The state estimate can be found by solving minimizing the scalar function J(x), thus

$$J(x) = [z - h(x)]R^{-1}[z - h(x)] \tag{5.9}$$

With the term [*z-h(x)*] equal to the residual vector *r*.

Because **h** is a nonlinear function of **x** the state estimate x can be found via numerical solution of the following equation.

$$\Delta x = (H^{T}R^{-1}H)^{-1}H^{T}R^{-1}\Delta z \tag{5.10}$$

The gain matrix $G=H^{T}R^{-1}H$ is not inverted. Rather, (5.10) is solved using sparsity technique involving factorization of *G*.

It is clear from (5.10) that large residues $r_i=[z_i-h_i(x)]$ due to gross measurement errors carries lots of weight and thus severely affect the accuracy of the state estimate. A good state estimator must be able to recognize and mitigate this aberration in measurements. Bad data detection is one of the most important functions of a state estimator. One approach was to reformulate (5.10) such that residual terms weighing heavily are scaled down [43]. Still another method bad data detection involves analyzing the residuals r_i evaluated at the solution, normalizing the residuals by their individual standard deviation, and executing statistical tests to identify residuals that are suspiciously high. The state estimator is rerun with these suspect measurements removed with the state estimate declared acceptable if the remaining residuals are permissible [22] [31]. The reader is referred to Mili [44] for a comparative study of various bad data detection techniques.

The system must be observable by PMUs for the SE to estimate the state. Observability ensures that the iterative equation (5.10) is solvable. If the system is not observable the gain matrix is singular and there is no unique solution to (5.10). This is called numerical observability. Topological observability is concerned with the determination of observability through a graphical analysis of the metering equipment and topology of the network. References [39][18] provide excellent material on topopogical observability.

5.2 Enhancing the WLS State Estimator Using Phasor Measurements

Several methods have been proposed to integrate phasor measurements to the classical WLS algorithm. Basically, the methods involved accommodating the direct angle measurements to the classical measurement set [53][62]. These were serious attempts towards integrating phasor measurements in the monitoring functions of power systems control center. It should be reiterated at this point that these methods still belong to the class of traditional state estimators characterized by unsynchronized measurements.

Most functioning state estimators in modern control centers are purchased as part of the large software package belonging to the Energy Management Systems functions of the control center computer. It is unlikely that the state estimation software can

accommodate phasor V and I measurements in its input set unless they are converted into equivalent P and Q measurements.

The model we conceive in this section is based on enhancing the output of the WLS estimator by modeling a linear measurement set composed of PMU measurements and WLS state estimate. Thus, the proposed measurement model is

$$\mathbf{M} = \mathbf{H}^c \mathbf{V}_{\text{est}} + \varepsilon \tag{5.11}$$

The superscript 'c' is used to uniquely identify the coefficient matrix H in this formulation since we will encounter the same symbol used in a different fashion in the subsequent sections.

The measurement vector is composed of 1) the direct state output from WLS 2) PMU voltage measurements and 3) PMU current measurements. Thus, our measurement model is shown below.

$$\mathbf{M} = \begin{bmatrix} \begin{bmatrix} \mathbf{V}_R \\ \mathbf{V_I} \end{bmatrix}_{SE} \\ \begin{bmatrix} \mathbf{V}_R \\ \mathbf{V_I} \end{bmatrix}_{PMU} \\ \begin{bmatrix} \mathbf{I}_R \\ \mathbf{I_I} \end{bmatrix}_{PMU} \end{bmatrix} = \begin{bmatrix} \mathbf{H11} & \mathbf{H12} \\ \mathbf{H21} & \mathbf{H22} \\ \mathbf{H31} & \mathbf{H32} \\ \mathbf{H41} & \mathbf{H42} \\ \mathbf{H51} & \mathbf{H52} \\ \mathbf{H61} & \mathbf{H62} \end{bmatrix} \begin{bmatrix} \mathbf{V}_R \\ \mathbf{V_I} \end{bmatrix} + \begin{bmatrix} \varepsilon_{VR}^{SE} \\ \varepsilon_{VI}^{SE} \\ \varepsilon_{VR}^{PMU} \\ \varepsilon_{VI}^{PMU} \\ \varepsilon_{IR}^{PMU} \\ \varepsilon_{II}^{PMU} \end{bmatrix} \tag{5.12}$$

Where the subscript R and I denotes the real and imaginary components of the voltages and current measurements. The errors are assumed to have zero-mean and un-correlated with a diagonal covariance matrix R defined by

$$\mathbf{R} = diag\{[\sigma^2_{|VR\_SE|} \ \sigma^2_{|VI\_SE|} \ \sigma^2_{|VR\_PMU|} \ \sigma^2_{|VI\_PMU|} \ \sigma^2_{|IR\_PMU|} \ \sigma^2_{|II\_PMU|} \ ]\} \tag{5.13}$$

Each element inside R is a diagonal sub matrix by itself, for example,

$$\sigma^2_{|VR\_SE|} = diag\{[\sigma^2_{|VR|\_1} \ \sigma^2_{|VR|\_2} \ \sigma^2_{|VR|\_3} \ ....\sigma^2_{|VR|\_N}]\} \tag{5.14}$$

The state estimate can be solved directly without iteration using the following weighted least square solution of (5.11).

$$\mathbf{V}_{\text{est}} = (\mathbf{H}^{\text{cT}} \mathbf{R}^{-1} \mathbf{H}^c)^{-1} \mathbf{H}^{\text{cT}} \mathbf{R}^{-1} \mathbf{M} \tag{5.15}$$

Elements of the H^c coefficient matrix

The following sub matrices of the coefficient matrix follow from the structure of the measurement model (5.12).

H11 = I;   H12 = 0

H21 = 0;   H22 = I;

Where **I** is an NxN identity matrix and N is the number of buses in the system. **0** is the NxN null matrix.

H31 is a *PxN* matrix, *P* being the number of PMUs. Each row *i* corresponds to pmu *i* and has all zeros except at the *jth* column corresponding to the index of V_{Ri}^{pmu} in the state vector.

$$\mathbf{H31}_i = [\ 0\ 0\ \dots\ 0\ .\ (1)\ ..\ 0\ ]$$

H32 = 0$_{(PxN)}$

H41 = 0$_{(PxN)}$

H42 is a *PxN* matrix, *P* being the number of PMUs. Each row *i* corresponds to pmu *i* and has all zeros except at the *jth* column corresponding to the index of V_{Ii}^{pmu} in the state vector.

$$\mathbf{H42}_i = [\ 0\ 0\ \dots\ 0\ .\ (1)\ ..\ 0\ ]$$

At this point we will develop the elements of H corresponding to the PMU current measurements. Given that we have the pi model of a transmission branch (line or transformer) in Figure 5- 1, the current Ipq is expressed as

$$Ipq = EpYpo + (Ep - Eq)Ypq \qquad (5.16)$$

Converting everything into rectangular components yields

$$Ipq_R = (G + Gpo)Ep_R - (B + Bpo)Ep_I - GEq_R + BEq_I \qquad (5.17)$$

$$Ipq_I = (B + Bpo)Ep_R + (G + Gpo)Ep_I - BEq_R - GEq_I \qquad (5.18)$$
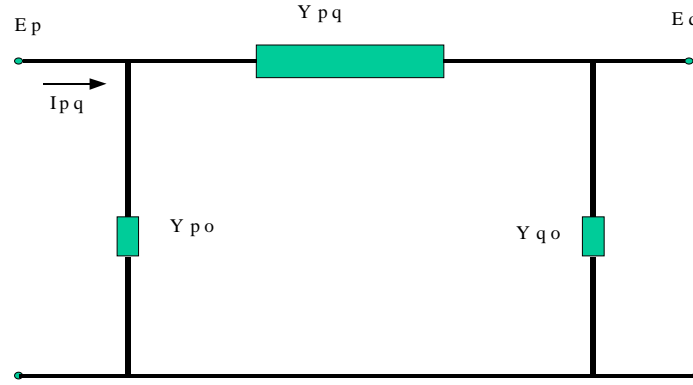
Figure 5- 1. Transmission Branch Pi Model

Therefore, **H51** is an LxN matrix (L being the number of PMU current measurements) whose elements of will be purely conductances while **H52** is an LxN matrix that has purely susceptance elements. Conversely, **H61** is an LxN matrix that has purely susceptance elements and those of the LxN matrix **H62** will be purely conductances.

Elements of the Covariance Matrix

The variances of meter measurements are usually given in terms of variance or standard deviation on the magnitude and angle. Our model requires covariance matrix elements in corresponding to phasor rectangular components. Thus, the following transformations are necessary

$$\sigma^2_{|VR|} \approx \cos^2\theta\sigma^2_{|V|} + |V|^2 \sin^2\theta\sigma^2_{|\theta|} \tag{5.19}$$

$$\sigma^2_{|VI|} \approx \sin^2\theta\sigma^2_{|V|} + |V|^2 \cos^2\theta\sigma^2_{|\theta|} \tag{5.20}$$

$$\sigma^2_{|IR|} \approx \cos^2\phi\sigma^2_{|I|} + |I|^2 \sin^2\phi\sigma^2_{|\phi|} \tag{5.21}$$

$$\sigma^2_{|II|} \approx \sin^2\phi\sigma^2_{|I|} + |I|^2 \cos^2\phi\sigma^2_{|\phi|} \tag{5.22}$$

Where θ and ϕ are the phasor angle of complex voltages and currents respectively. The $\sigma_{|V|}$ and $\sigma_{|\theta|}$ from the classical state estimator can be evaluated directly from the gain matrix at the convergence point. Reference [46] provides an interesting discussion on how to derive these sigmas.

Aligning The State Estimator And PMU Angle References

The reference of PMU measurements is the reference associated with the timing signal from the GPS, say δ_{ref}, while the state from the traditional SE are referred to the angle of one of the buses, say θ_{slack}. It is necessary to align these two references; otherwise the results would not make sense. Figure 5-2 shows how to align these references. In this figure, it is shown that if we align the WLS and the PMU references we should measure the same angle of the voltage phasor at bus i. To accomplish this, we must adjust the PMU angle reference by a certain amount Φ to align it with the SE reference. From the figure it is obvious that

$$\theta_i + \theta_{slack} = \delta_i + \delta_{ref} \tag{5.23}$$

This implies that the angle required to align the references is

$$\Phi = \delta_{ref} - \theta_{slack} = \theta_i - \delta_i \tag{5.24}$$

If we have m sets of PMU measurements then we can take the average so that a good estimate of Φ (the phase shift between PMU reference and the slack bus angle) is

$$\Phi = \sum_{i=1}^{m} \frac{\theta_i - \delta_i}{m} \tag{5.25}$$
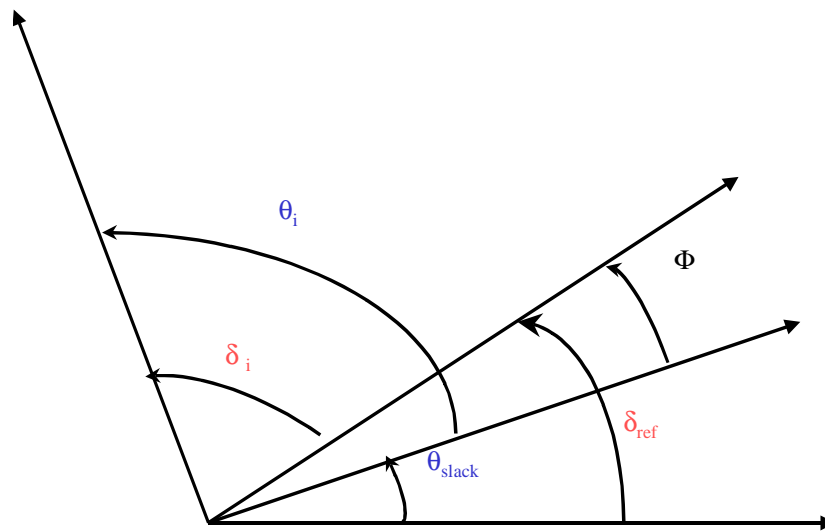
Figure 5-2. Aligning the PMU and WLS estimator reference

Numerical Results

This algorithm was tested on the New England 39 Bus system shown in Figure 5-3. The accuracy of the proposed model was tested against several indices. Here we coded a program designed to execute the WLS state estimation (see Appendix B). The model for the diagonal entries σ_{ii}^2 of the noise covariance matrix **R** are taken from the standard deviation of metering error assumed in [22][3] thus,

$$\sigma_i = (0.02m + 0.0052 f_s)/3 \tag{5.26}$$

Where

$$m = \begin{cases} \sqrt{P_{pq}^2 + Q_{pq}^2} \\ V_p \end{cases} \tag{5.27}$$

For flow and voltage magnitude measurements respectively and f_s is the full-scale value of the instrumentation.

Injection measurements were not made part of the WLS measurements set. The accuracy of the transducers for the PMUs was taken from the ANSI standards [4] and summarized in Tables 5-1 and 5-2. The corresponding standard deviations of the measurements were then calculated using the relationship $\pm\sigma = \pm$Error (Magnitude or Phase). The same standard deviations were used to calculate the variances in rectangular form 5.19-5.22. For simulation, the measurements with random errors were generated using the formula

$$z^{meas} = z^{true} + \sigma^{meter} * r \tag{5.28}$$

Where r is a normally distributed random number.

Table 5-1. Maximum Magnitude and Phase Error for ANSI class type CTs

ANSI CT Type	Load Current	Max Magnitude ± Error pu	Max Phase ± Error degrees
Relaying	100 to 2000%	0.10	NA
Metering 1.2	10%	0.024	2.08
	100%	0.012	1.04
Metering 0.6	10%	0.012	1.04
	100%	0.006	0.52
Metering 0.3	10%	0.006	0.52
	100%	0.003	0.26

Table 5-2. Maximum Magnitude and Phase Error for ANSI class type PTs

ANSI PT Type	Max Magnitude ± Error pu	Max Phase ± Error degrees
Relaying	0.10	NA
Metering 1.2	0.012	2.08
Metering 0.6	0.006	1.04
Metering 0.3	0.003	0.52

Basically, two groups of tests were conducted. The first battery of tests correspond to full P&Q line flow measurements, that is, all lines are metered on both ends totaling 184 flow measurements. Also, 39 bus voltages are metered. The second group of tests corresponds to partial P&Q line flow measurements. In this set of exercises, there were only 106 P&Q line flow measurements and 21 bus voltage measurements. A total of 100 Monte Carlo runs were executed for each problem case. The standard deviation of power flow errors and voltage errors were monitored.
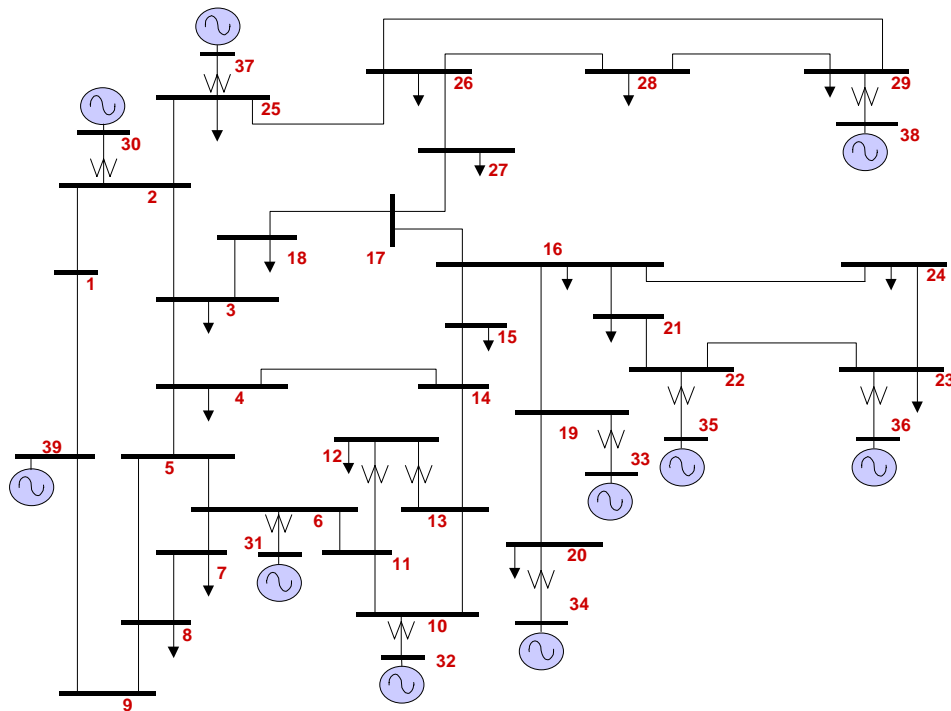
NEW ENGLAND 39-BUS SYSTEM Single Line Diagram

Figure 5- 3. The New England 39 Bus Test System
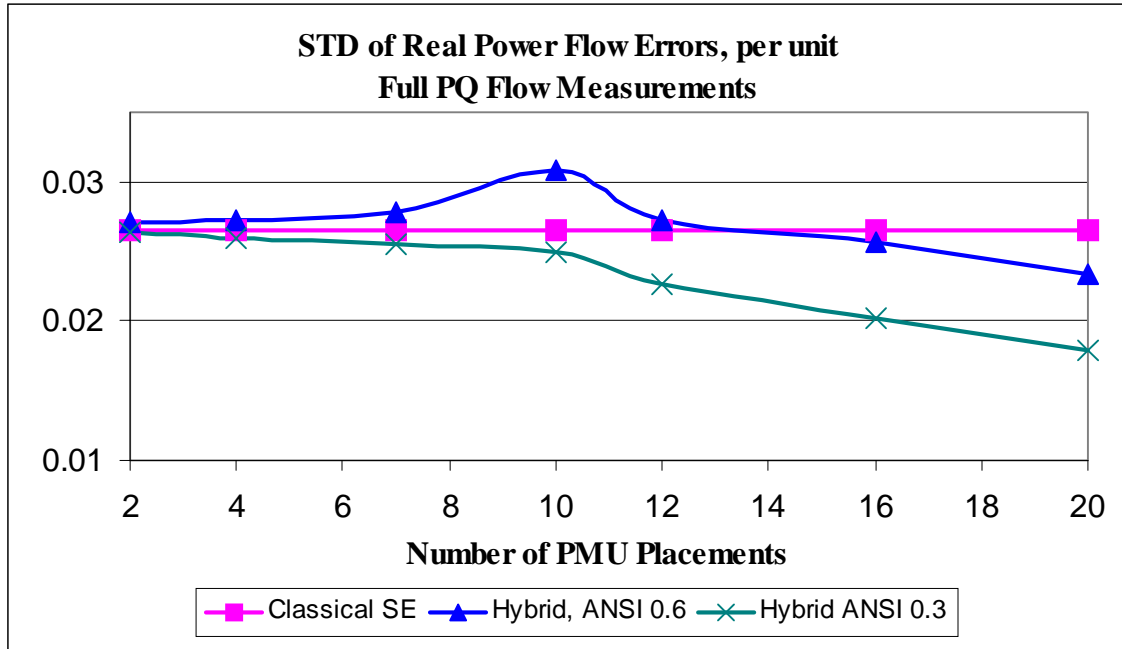
Figure 5- 4. Standard Deviation of Real Power Flow Errors, per unit: Full PQ Flow
        Measurements
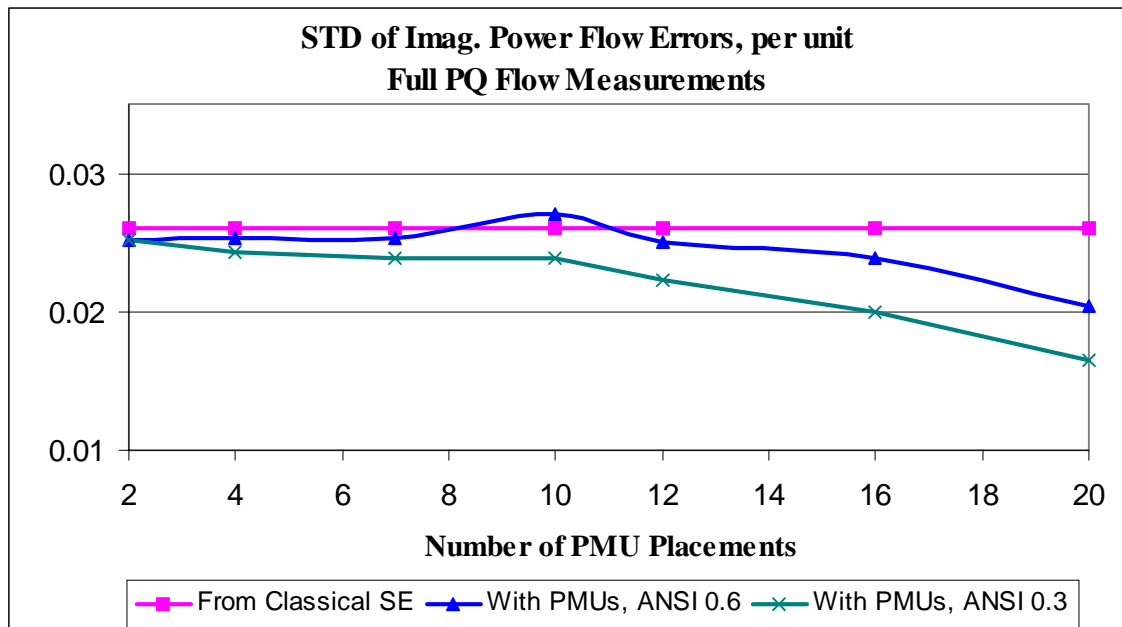
Figure 5- 5. Standard Deviation of Imaginary Power Flow Errors, per unit: Full PQ Flow
        Measurements
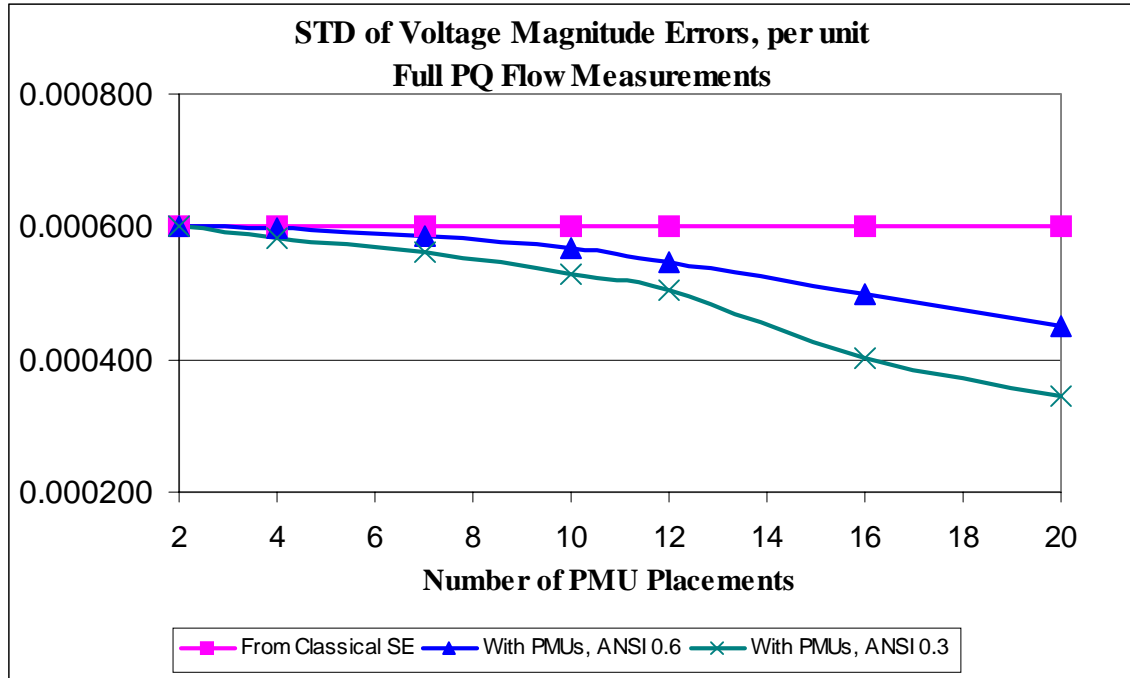
Figure 5-6. Standard Deviation of Voltage Magnitude Errors, per unit: Full PQ Flow
Measurements
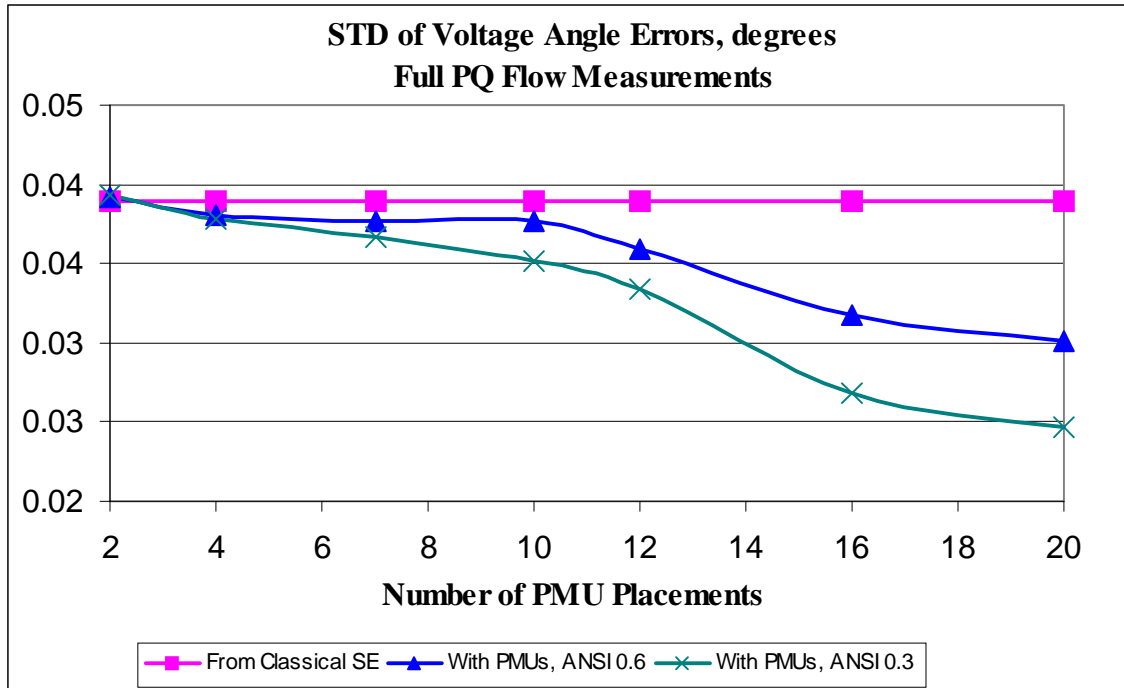
Figure 5- 7. Standard Deviation of Voltage Angle Errors, degrees: Full PQ Flow
Measurements
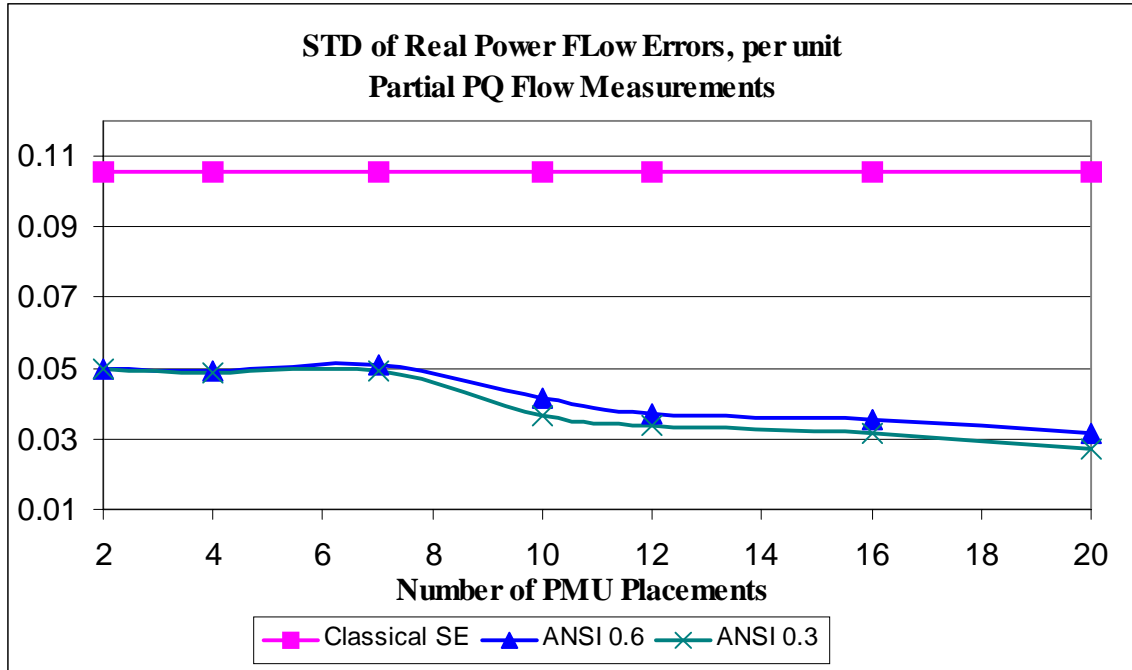
Figure 5- 8. Standard Deviation of Real Power Flow Errors, per unit: Full PQ Flow Measurements
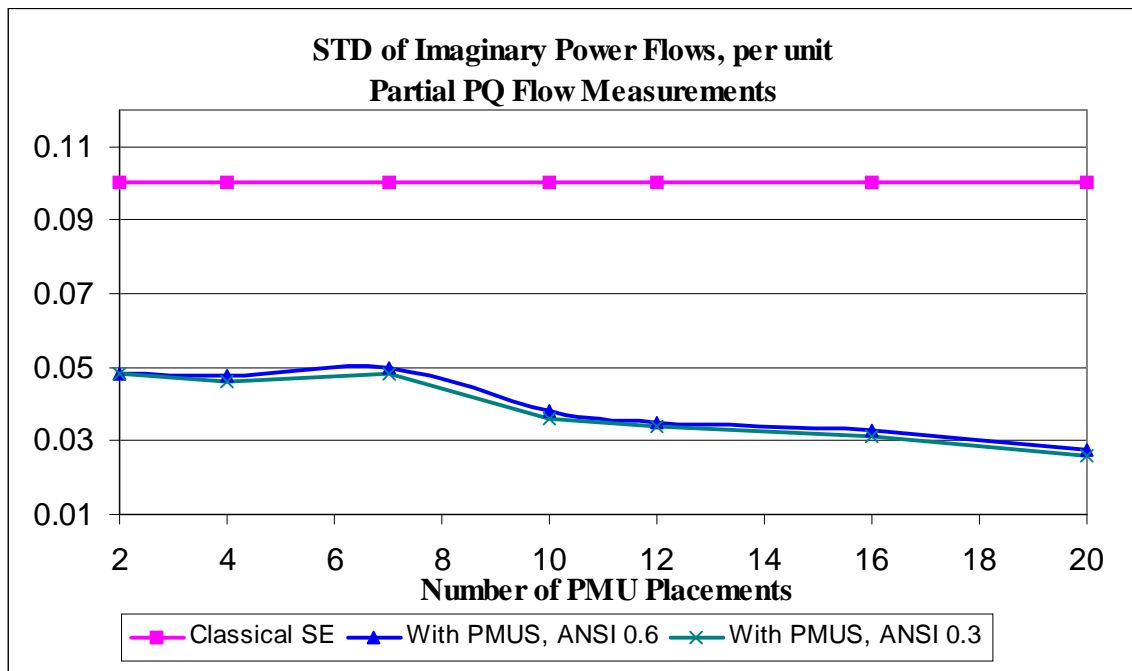
Figure 5- 9. Standard Deviation of Imaginary Power Flow Errors, per unit: Partial PQ Flow Measurements
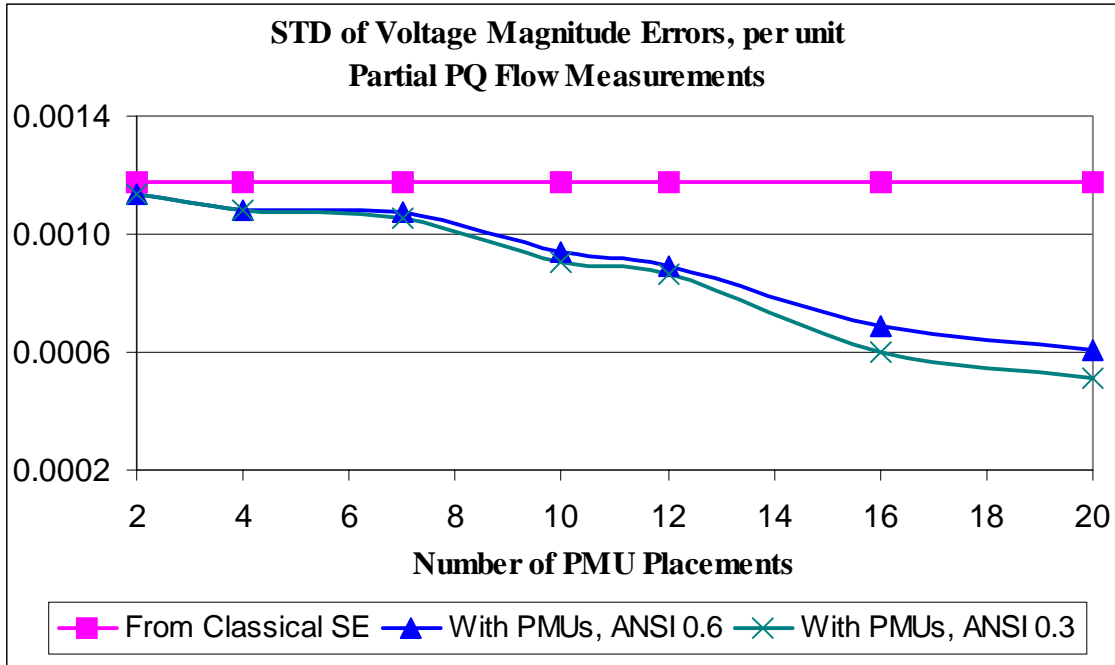
Figure 5- 10. Standard Deviation of Voltage Magnitude Errors, per-unit: Partial PQ
Flow Measurements
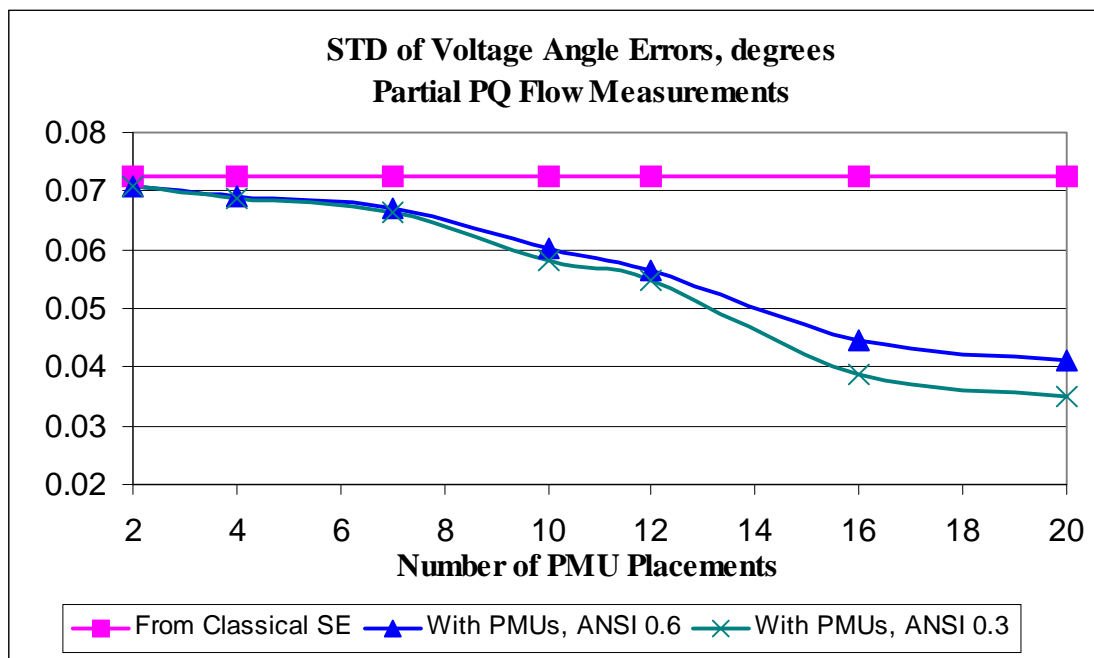
Figure 5- 11. Standard Deviation of Voltage Angle Errors, degrees: Partial PQ Flow
Measurements

The following conclusions can be derived from the simulations as illustrated by the preceding figures:

- When the system is fully metered, the improvements due to PMUs on accuracy of real and reactive flows become significant only with a large number of PMUs (close to 2/5 of the number of buses). However, using more accurate PMU transducers would result in less PMUs to achieve the same improvement in estimation.

- The impact of PMU measurements is significant when the system has only partial real and reactive flow measurements. Figure 5- 8 and Figure 5- 9 show that even a small number of PMUs could significantly improve the accuracy of the line flow estimates.

5.3 Interpolation of State of Unobserved Buses Using Bus Admittance Matrix

As discussed in Chapters 2 and 3 PMU placements for incomplete observability intentionally creates pockets of unobservable regions around the network. These unobservable regions are composed of one or more unobserved buses depending on the desired depth of unobservability. Thus, a PMU placement strategy S results in the set of unobserved buses U and observed/directly measured buses O. The interpolation problem is concerned with interpolating the voltages of unobserved buses from the observed ones. Formally: given a power system with PMU placement set S of size *v* satisfying a desired depth of unobservability υ determine relationship between the vector of complex voltages of unobserved buses $\mathbf{E}_U$ and the vector of complex voltages of observed buses $\mathbf{E}_O$.

Kabra [36] assumed the linear relationship $\mathbf{E}_U=\mathbf{H}\mathbf{E}_O$ holds with the $\mathbf{H}$ being full, that is, each of the unobserved bus voltage becomes a linear combination of the network's measured and calculated bus voltages. The full dependency of each voltage in $\mathbf{E}_U$ to each voltage in $\mathbf{E}_O$ created errors in the model. Using Kirchoff's Current Law, it was inferred in Section 2.2 that only the state of neighboring buses affects the voltage of any unobserved bus.

Barber [7] investigated the errors in estimation of the load portion of the network given that PMUs are installed at the generating stations. The estimate of the load portion

was based on the bus admittance matrix model as in this thesis but with constant interpolation coefficients stored in matrix form. This thesis will prove that with PMU placed for incomplete observability the problem of estimation of the unobserved portions of the system can be done independently from each other.

    The following presents the theoretical background behind the interpolation equations for estimating the complex voltages of the unobserved buses. We start with the network model defined by the bus admittance matrix (5.29). From the bus frame of reference, the nodal equations in a power system expressed in matrix form is

$$[\mathbf{I}] = [\mathbf{Y}_{BUS}][\mathbf{E}_{BUS}] \tag{5.29}$$

    We then partition the bus admittance matrix $\mathbf{Y}_{BUS}$ into four sub-matrices, and reorder the rows so that the observed voltage vector $\mathbf{E}_O$ is listed first followed by the vector of unobserved buses $\mathbf{E}_U$ yielding

$$\begin{bmatrix} \mathbf{I}_o \\ \mathbf{I}_U \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{OO} & \mathbf{Y}_{OU} \\ \mathbf{Y}_{UO} & \mathbf{Y}_{UU} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{E}_o \\ \mathbf{E}_U \end{bmatrix} \tag{5.30}$$

    At this particular operating point, we model the injections $\mathbf{I}_U$ at the unobserved buses $\mathbf{U}$ as equivalent load admittances. If we let N_u to be number of unobserved buses, the active injections $\mathbf{I}_U$ at the unobserved buses are expressed as

$$[\mathbf{I}_U] = \left[\frac{P_i - jQ_i}{E_i^*} \right], \ i=1,2,\ldots N_u \tag{5.31}$$

    The equivalent load admittance vector $\mathbf{Y}_U$ shown in (5.32) follows from the relationship Y=I/E.

$$[\mathbf{Y}_U] = \left[\frac{P_i - jQ_i}{\left| E_i \right|^2} \right], i=1,2,\ldots N_u \tag{5.32}$$

    With the current injections at the unobserved buses converted into admittances, the original matrix equation (5.30) now becomes

$$\begin{bmatrix} \mathbf{I}_O \\ -- \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{OO} & \mathbf{Y}_{OU} \\ ------ & \\ \mathbf{Y}_{UO} & \mathbf{Y}_{UU}^{'} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{E}_O \\ -- \\ \mathbf{E}_U \end{bmatrix} \tag{5.33}$$

Where

$$\mathbf{Y}_{UU}^{'} = \mathbf{Y}_{UU} + \mathbf{Y}_L = \mathbf{Y}_T \tag{5.34}$$

The load admittance matrix $\mathbf{Y}_L$ is of the same dimension as $\mathbf{Y}_{UU}$ and can be expressed as

$$\mathbf{Y}_L = diag\left\{\frac{S_U^*}{\left|E_U\right|^2}\right\} \tag{5.35}$$

Performing the indicated operation on the lower row of matrix equation (5.33) yields

$$0 = \mathbf{Y}_{UO}\mathbf{E}_O + \mathbf{Y}_T\mathbf{E}_U \tag{5.36}$$

Which when solved for $\mathbf{E}_U$ will now express the relationship between the vector of unobserved buses with the vector of observed buses. Hence,

$$\mathbf{E}_U = -\mathbf{Y}_T^{-1}\mathbf{Y}_{UO}\mathbf{E}_O \tag{5.37}$$

Let's denote the number of observed buses to be N_o. The product $\mathbf{Y}_T^{-1}\mathbf{Y}_{UO}$ is sparse matrix with admittance elements of dimension $N_u x N_o$. Let us define this product to be $\mathbf{H}$

$$\mathbf{H} = -\mathbf{Y}_T^{-1}\mathbf{Y}_{UO} \tag{5.38}$$

Yielding our model for interpolation of unobserved buses in matrix form.

$$\mathbf{E}_U = \mathbf{H}\mathbf{E}_O \tag{5.39}$$

Where $\mathbf{H}$ is the matrix of interpolation coefficients; each row $\mathbf{H}i$ contains the interpolation coefficients of unobserved bus i. Thus, we can write interpolation equations for each unobserved bus *i* as the dot product,

$$E_i^U = \mathbf{H}_i \cdot \mathbf{E}^O \text{ , i=1,2,...Nu} \tag{5.40}$$

We have superscripted the identifiers U and O without loss of generality.

In large systems it is not computationally efficient to evaluate the product $\mathbf{Y}_T^{-1}\mathbf{Y}_{UO}=\mathbf{H}$ each instance of topology change. It would be helpful to know the structure of $\mathbf{H}$. Figure 5- 12 shows a sub-network with two unobserved regions labeled U1 and U2, each one containing 2 unobserved buses respectively. The rest of the buses shown in this figure are assumed to be observable by PMUs (not shown here); the dashed arrows indicate connection to other parts of the network. The matrix of interpolation coefficients H in this sub-network is

$$\mathbf{H} = -\mathbf{Y}_T^{-1}\mathbf{Y}_{UO} \tag{5.41}$$

We can divide the matrix $\mathbf{Y}_T$ to correspond to each unobserved region. Thus,

$$\mathbf{Y}_T = \begin{bmatrix} \mathbf{Y}1_T & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}2_T \end{bmatrix} \tag{5.42}$$

Both $\mathbf{Y}_1{}^T$ and $\mathbf{Y}_2{}^T$ are 2x2 sub-matrices and since U1 and U2 are not always contiguous, there are no $\mathbf{Y}_{BUS}$ elements coupling the two regions. Therefore, YT is block diagonal and its inverse is just the inverse of the block diagonal matrices $\mathbf{Y}_1{}^T$ and $\mathbf{Y}_2{}^T$. Thus, we can rewrite equation (5.41) in block form.

$$\begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix} = -\begin{bmatrix} \mathbf{Y}1_T^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}2_T^{-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{Y}1_{UO} \\ \mathbf{Y}2_{UO} \end{bmatrix} \tag{5.43}$$

Or

$$\begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \end{bmatrix} = -\begin{bmatrix} \mathbf{Y}1_T^{-1}\mathbf{Y}1_{UO} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}2_T^{-1}\mathbf{Y}1_{UO} \end{bmatrix} \tag{5.44}$$

Implying,

$$\mathbf{H}_1 = -\mathbf{Y}1_T^{-1}\mathbf{Y}1_{UO}$$
$$\mathbf{H}_2 = -\mathbf{Y}2_T^{-1}\mathbf{Y}2_{UO} \tag{5.45}$$

Now, the matrix $\mathbf{Y}1_{UO}$ contains the $\mathbf{Y}_{BUS}$ elements mutual to unobserved region U1 and the whole set of observed buses O. Similarly, the matrix $\mathbf{Y}2_{UO}$ contains the $\mathbf{Y}_{BUS}$ elements mutual to unobserved region U2 and the set O. Both $\mathbf{Y}1_T{}^{-1}$ and $\mathbf{Y}2_T{}^{-1}$ are full matrices. Performing the indicated multiplication in equation (5.45) for $\mathbf{H}_1$ would result in nonzero elements on $\mathbf{H}_1$ only on columns corresponding to the any bus in O linked to either Bus 1 or Bus 2 in U1. The same reasoning holds when the matrix multiplication is done for $\mathbf{H}_2$. Thus, we can derive the following conclusions: *To compute the interpolation coefficients for an unobserved bus we need only to consider the bus admittance matrix corresponding to the region containing this unobserved bus and any other unobserved buses contiguous to it together with their immediate neighboring observed buses.*

The aforementioned basically cuts the process of evaluating the interpolation coefficients into much smaller autonomous sub-networks resulting in computationally efficient calculations. In subsequent discussions though we still maintain the whole network when referring to $\mathbf{H}.$ Take into account that implied calculations for each unobserved bus i interpolation coefficients $\mathbf{H}i$ is done independently.
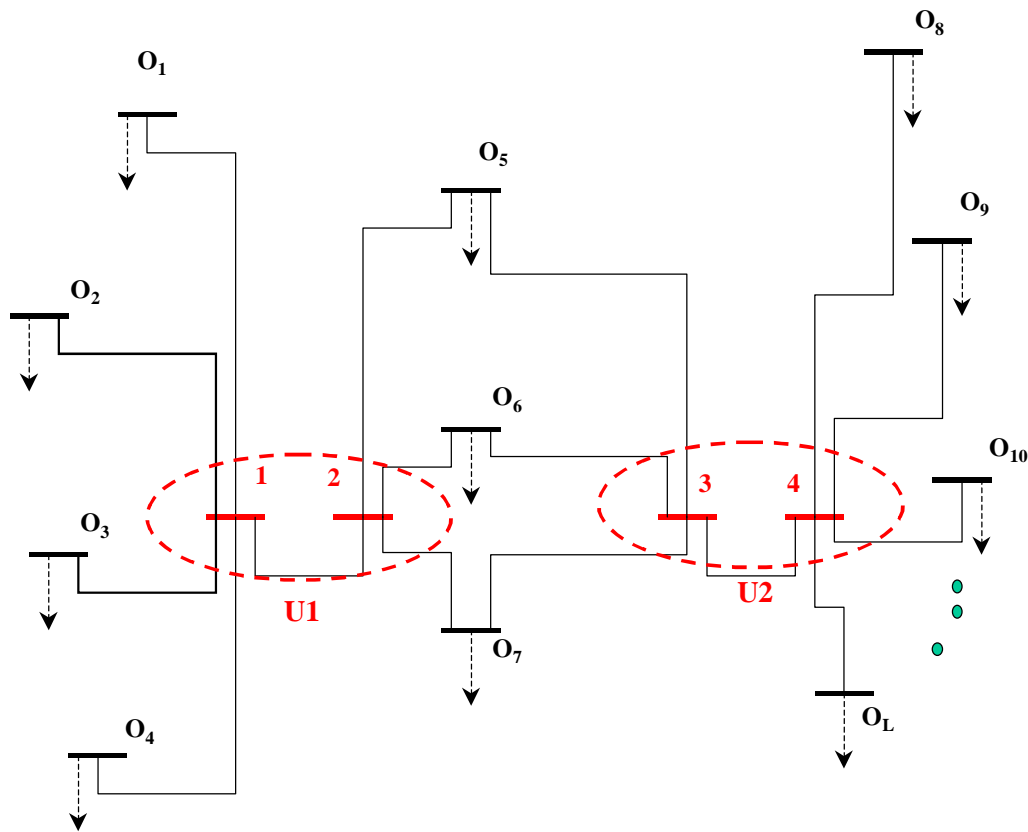
Figure 5- 12. Sub-Network With Two Unobserved Regions and Their Neighboring Buses

The vector of interpolators associated with unobserved bus i, $\mathbf{H}i$ can be expressed as the product $\mathbf{H}i=\mathbf{Z}_T\mathbf{Y}_{UO}{}^i$ wherein $\mathbf{Z}_T=-\mathbf{Y}_T{}^{-1}$. Now $\mathbf{Y}_{UO}{}^I$ is sparse with non-zero elements y_{ij}s if there's a link (line, transformer, etc.) between i and j. $\mathbf{Z}_T$ has a main diagonal because Y_T has a main diagonal composed of self primitive admittances $(\mathbf{Y}_{UU})_{ii}$ and the equivalent load admittances $\mathbf{Y}_U$. Therefore, the product $\mathbf{Z}_T\mathbf{Y}_{UO}{}^i$ ensures that the row vector $\mathbf{H}i$ will have non-zero element on position j if a link exists between i and j. Furthermore, nonzero elements also exist for position j in $\mathbf{H}i$ if such j is linked to another unobserved bus k that is also linked to i. Graphically, if we have an unobserved region containing I number of unobserved and contiguous buses, encircled by a set J of observed buses, then it is only necessary to consider the set J of observed buses in interpolating the set I of unobserved buses.

It has been proven that a more efficient way of calculating **H**i should then be based only on the admittances of the sub-graph or sub-network containing the unobserved buses and their observed neighbors! There will be two instances when the interpolators **H**i changes: 1) topology – line outage inside the sub-network, and 2) significant load change – note that the equivalent load impedances were calculated from a given operating point.

5.4 Updating the Linear State Estimators Using Sensitivity Factors

Any set of interpolation coefficients **H** calculated using (5.41) assumes a reference operating point, say '0'. That is,

$$\mathbf{H}^0 = -(\mathbf{Y}_T^0)^{-1}\mathbf{Y}_{UO} \tag{5.46}$$

Any deviation in operating point will create errors in the following interpolation equation.

$$\mathbf{E}_U = \mathbf{H}^0\mathbf{E}_O \tag{5.47}$$

A method of updating the interpolators **H** as the system changes its operating point is the subject of the foregoing discussion.

Given that we have

$$\mathbf{H} = -\left(\mathbf{Y}_{UU} + diag\left\{\frac{S_U^*}{\left|E_U\right|^2}\right\}\right)^{-1}\mathbf{Y}_{UO} \tag{5.48}$$

If we let

$$\mathbf{Y}_T = \mathbf{Y}_{UU} + diag\left\{\frac{S_U^*}{\left|E_U\right|^2}\right\} \tag{5.49}$$

Then we have the aforementioned expression of the interpolation coefficients.

$$\mathbf{H} = -\mathbf{Y}_T^{-1}\mathbf{Y}_{UO} \tag{5.50}$$

Or alternatively,

$$\mathbf{Y}_T\mathbf{H} = -\mathbf{Y}_{UO} \tag{5.51}$$

Taking the differential of both sides of this matrix equation yields

$$\Delta\mathbf{Y}_T\mathbf{H} + \mathbf{Y}_T\Delta\mathbf{H} = \mathbf{0} \tag{5.52}$$

Which when solved for Δ**H** yields

$$\Delta \mathbf{H} = -\mathbf{Y}_T^{-1} \Delta \mathbf{Y}_T \mathbf{H} \tag{5.53}$$

Now since

$$\mathbf{Y}_T = \mathbf{Y}_{UU} + diag\left\{\frac{S_U^*}{|E_U|^2}\right\} \tag{5.54}$$

Then the differential of $\mathbf{Y}_T$ is

$$\Delta \mathbf{Y}_T = diag\left\{\frac{\Delta S_U^*}{|E_U|^2}\right\} \tag{5.55}$$

Since $\mathbf{Y}_{UU}$ is a constant matrix of $\mathbf{Y}_{BUS}$ elements.

Finally, we have the following expression for the change in the value of the interpolation coefficient matrix $\mathbf{H}$.

$$\Delta \mathbf{H} = -\mathbf{Y}_T^{-1}\left[diag\left\{\frac{\Delta S_U^*}{|E_U|^2}\right\}\right]\mathbf{H} \tag{5.56}$$

Assume that we have an initial value of interpolation coefficients $\mathbf{H}^0$ calculated at the operating point $\mathbf{S}_U^0$ with

$$\mathbf{Y}_T^0 = \mathbf{Y}_{UU}^0 + diag\left\{\frac{(S_U^0)^*}{|E_U^0|^2}\right\} \tag{5.57}$$

If we define a change in operating point as

$$\Delta \mathbf{S}_U^* = \mathbf{S}_U^* - (\mathbf{S}_U^0)^* \tag{5.58}$$

Then,

$$\Delta \mathbf{H} = -(\mathbf{Y}_T^0)^{-1}\left[diag\left\{\frac{\Delta S_U^*}{|E_U^0|^2}\right\}\right]\mathbf{H}^0 \tag{5.59}$$

Which expresses the relationship between the incremental (sensitivity) change in the interpolation coefficient matrix with change in injected power at the unobserved bus. Hence, the interpolation formula for unobserved buses becomes

$$\mathbf{E}_U = (\mathbf{H}^0 + \Delta \mathbf{H})\mathbf{E}_O \tag{5.60}$$

Figure 5-13 illustrates the proposed scheme to implement interpolation coefficient update using sensitivity factors derived from the classical state estimator. It is to be stressed that this is a PMU-based estimator in the sense that the state is available at the rate PMUs measure the phasors. It is to be expected that the communication link will result in delay in receiving the phasor measurements. But since phasors are time-tagged and aligned at the data concentrator, the interpolated voltage of the unobserved buses will be in time synchronism with the rest of the phasor measurements.

In this proposed scheme, the classical state estimator will provide state estimate every scan period. The state will then be used to calculate the net bus injections in the system including at the unobserved buses. This injection will then be used to update the interpolation coefficients using the method described above.
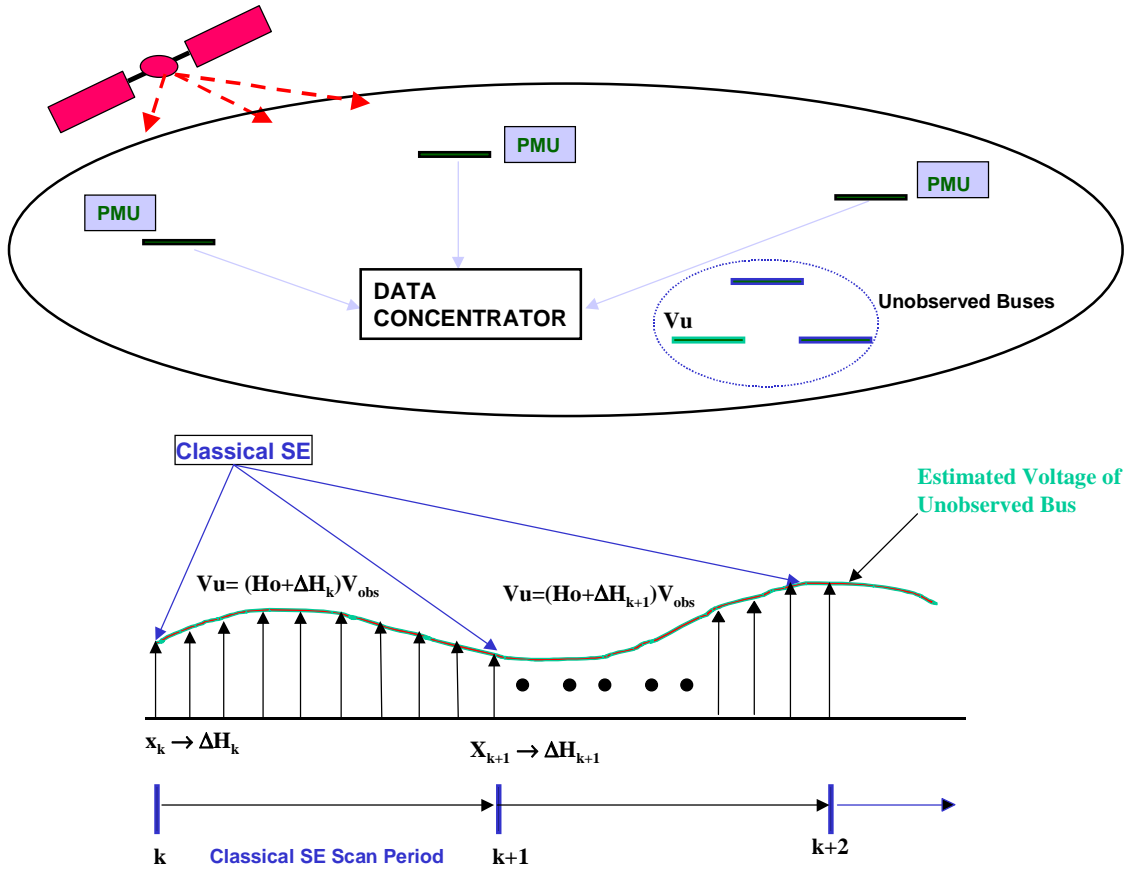
Figure 5-13. Schematic Diagram of a Hybrid State Estimator Utilizing the Classical SE to Update the Interpolation Coefficients H

5.5 Hybrid WLS and Linear State Estimator

In this section we formally apply the theory behind the concept of interpolating the state of unobserved buses developed in the Section 5.3. Basically, we propose a scheme wherein the interpolation coefficients are periodically updated using sensitivities with respect to the injection at the unobserved buses. Since we assume that the system is not observable with PMUs (else we can do direct state measurement), the traditional state estimator can be utilized to calculate these injections. This approach will be termed a hybrid state estimator; a method of executing PMU-based state estimation on an incompletely observable system with PMUs that utilizes the output of the traditional state estimators in a utility's energy management system. Figure 5-3 shows a schematic diagram of the hybrid state estimator. Basically, at each scan of the traditional SE the interpolation coefficients for the PMU-based SE are updated to reflect the most recent operating point of the system. Figure 5-4 illustrates the effect of this hybrid estimator. Shown here is the true state of an unobserved bus. The dashed curve reflects the expected value of the voltage when the interpolation coefficients are maintained constant. A gradually increasing error should be expected as the operating point moves farther away from the base operating point from which the interpolation coefficients were calculated. The dashed-dot curve illustrates the estimated voltage when the interpolation coefficients are updated using the output from the traditional state estimator. Note also that there will be errors in estimation as the system moves farther away from the base case operating point. However, the error is not as significant as when we use constant interpolators. It is assumed therefore that a fresh new set of interpolators must be used for significant changes in operating point. For example, in typical daily load changes we can store three sets of interpolation coefficients: each set corresponding to off-peak, morning peak and nighttime peak respectively. It is expected that this proposed hybrid state estimator scheme would persist in most power systems primarily due to limited communication facilities. In the next section, we will evaluate the accuracy of the hybrid estimator using a real world power system.
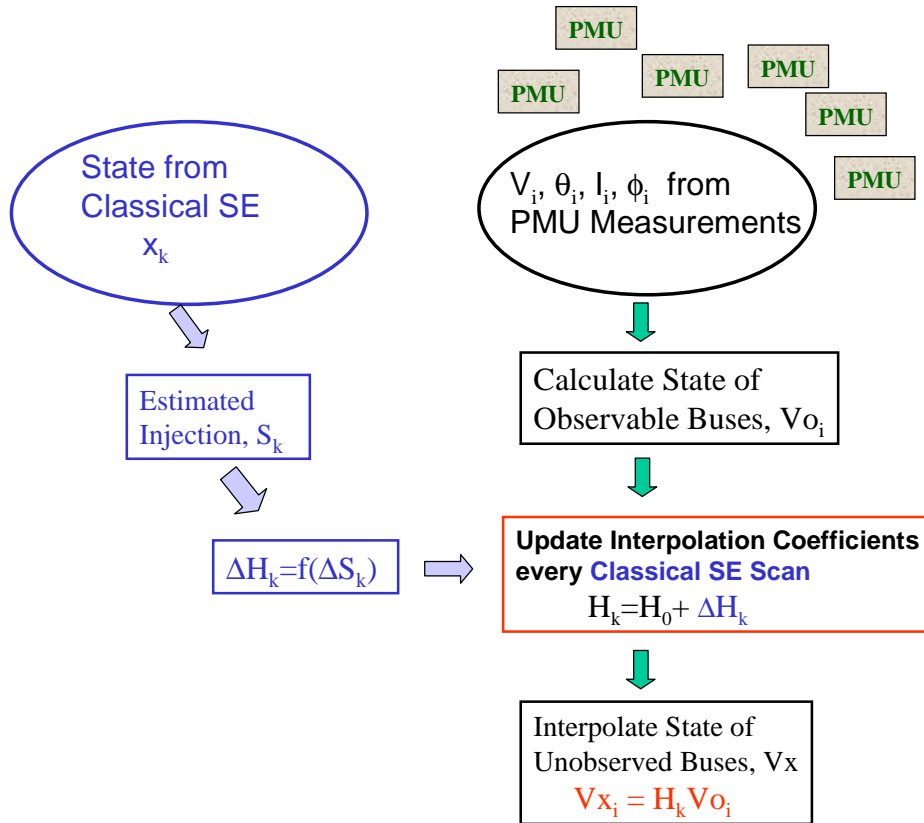
Figure 5-14. Flow Chart of the Hybrid State Estimator
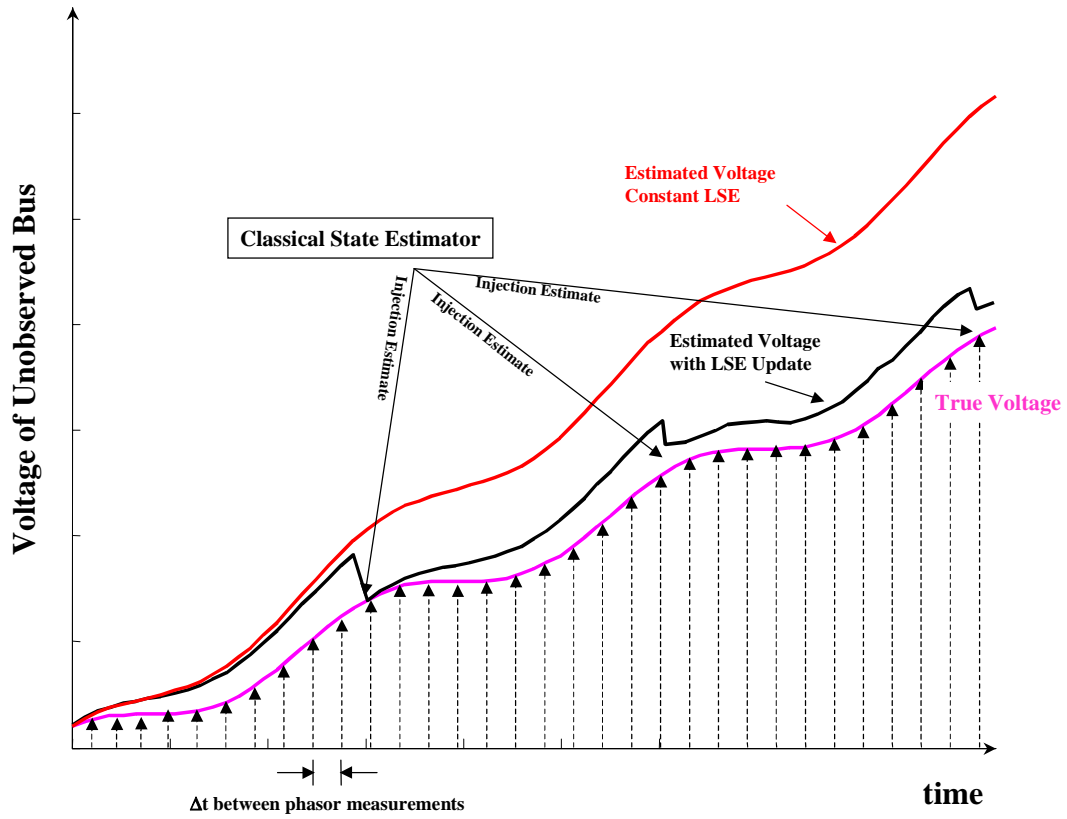
Figure 5-15. Traces of Voltage on Unobserved Bus: True Value Compared with
Estimated Value Using Hybrid SE or Constant LSE

5.6 Numerical Results
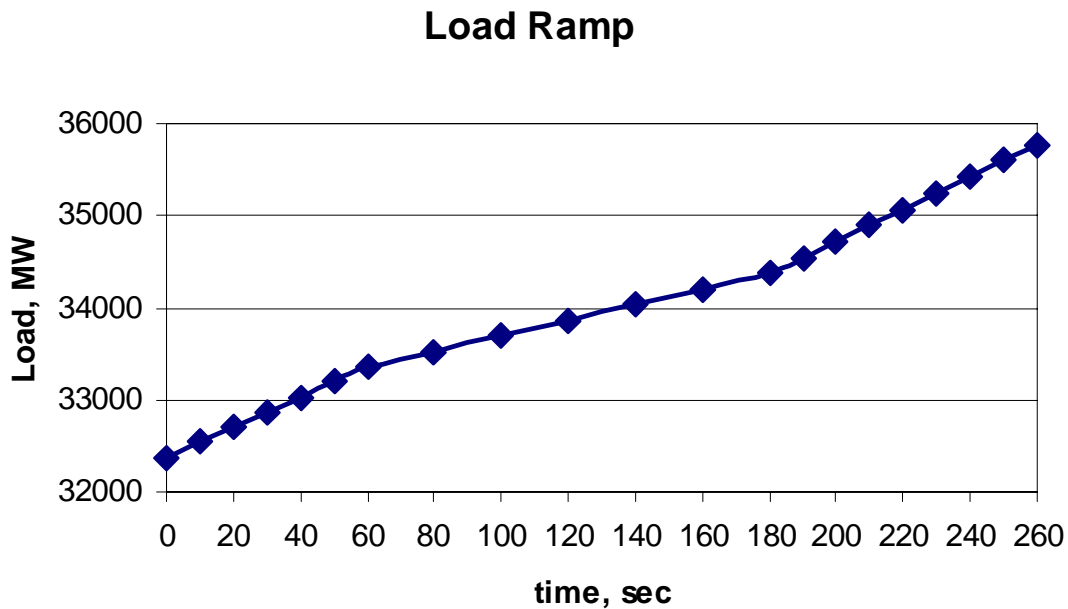
Load Ramp

Figure 5-16. Load Ramp Used to Test Proposed State Estimation Model

Figure 5-16 shows a load ramp used to test the accuracy of the developed interpolation of the unobserved buses in Utility System A. The total load ramp of 3400 MW occurs within 4.33 minutes at a rate of 784 MW per minute. This extremely fast load change can be expected in the morning. The individual substation loads were assumed to pick up the load according to their proportion to the total system load.

It is assumed that PMUs transfer information at the rate of 4 phasors per second to the data concentrator. For this 60 Hz system, this is equivalent to one phasor every 15 cycles.

To simulate the system response to this load ramp, a total of 20 load flow snapshots were made, each snapshot corresponds to a system-wide load increase of 0.5% over the previous case.

Table 5-3 shows the extent of observability on the communication-constrained network of Utility System A. The 3-staged phased installation of PMUs progressively increases observability but due to communication constraints, it is not possible to totally observe the system. In this network the minimum number of PMU placement to maximize observability could be achieved with 88 PMUs.

Table 5-3. Observability Status Associated with PMU Phased Installation at Utility
System B

Index	# of PMUS	# of Unobserved Buses	# of Unobserved Lines
PHASE 1	59	196	281
PHASE 2	69	158	228
PHASE 3	88	114	160

The proceeding discussion focuses on the results of simulation to test the interpolation coefficients update method. Several indices were used to assess the value of the proposed scheme in terms of errors on voltage magnitudes, voltage angles and line flows. It is assumed here that we have a very accurate classical state estimator, but rather slow because of the scan rate. It is further assumed that the voltages from the classical state estimator have virtually zero error. The more pragmatic case of errors in the measurements for both the phasors and the state from the classical state estimator will be dealt separately in the next section.

Performance Indices

The following indices were used to measure the performance of the proposed method. The errors in voltage magnitudes, angles and line flows occur at each instance of synchronized phasor measurement as the system responds to the load ramp. It is therefore logical to calculate the mean error over the time span of the ramp. Thus, we define the mean value of voltage magnitude error at bus i over a time period T as the difference between its estimated value and its true value as

$$\overline{V}_i^{ERR} = \overline{V}_i^{EST} - \overline{V}_i^{TRUE} \tag{5.61}$$

Similarly, the mean value of voltage angle error at bus i is

$$\overline{\theta}_i^{ERR} = \overline{\theta}_i^{EST} - \overline{\theta}_i^{TRUE} \tag{5.62}$$

and the mean value of the apparent flow error on line j is

$$\overline{f}_j^{ERR} = \overline{f}_j^{EST} - \overline{f}_j^{TRUE} \tag{5.63}$$

With the aforementioned definitions we now develop the following indices. For a system with *Nu* number of unobserved buses, the mean, standard deviation and maximum of average voltage magnitude error respectively are

$$\mu(Verror) = \frac{1}{Nu} \sum_{i=1}^{Nu} |\overline{V}_i^{err}| \tag{5.64}$$

$$\sigma(Verror) = \frac{1}{Nu} \sum_{i=1}^{Nu} |\overline{V}_i^{err}| \tag{5.65}$$

$$\sup(Verror) = \max_i |\overline{V}_i^{err}| \tag{5.66}$$

We extend the same indices definition for the bus voltage angle. Thus,

$$\mu(\theta error) = \frac{1}{Nu} \sum_{i=1}^{Nu} |\overline{\theta}_i^{err}| \tag{5.67}$$

$$\sigma(\theta error) = \frac{1}{Nu} \sum_{i=1}^{Nu} |\overline{\theta}_i^{err}| \tag{5.68}$$

$$\sup(\theta error) = \max_i |\overline{\theta}_i^{err}| \tag{5.69}$$

Finally, assuming we have N_L number of lines then we have the following error indices for the line power flows. In the calculation however we used only one power flow information per line.

$$\mu(ferror) = \frac{1}{N_L} \sum_{j=1}^{NL} |\overline{f}_j^{err}| \tag{5.70}$$

$$\sigma(ferror) = \frac{1}{N_L} \sum_{j=1}^{NL} |\overline{f}_j^{err}| \tag{5.71}$$

$$\sup(ferror) = \max_j |\overline{f}_j^{err}| \tag{5.72}$$

Simulation

In this first group of simulation we assume that the traditional state estimator outputs very accurate state estimate resulting in equally accurate injection estimates at

unobserved buses. We assume a traditional SE scan period of 10 seconds and that calculation of injections, sensitivities and LSE updating occur instantaneously.

Figure 5-17 shows the evolution of voltage magnitude of unobserved bus #243 in response to the load ramp. The solid curve shows its actual voltage magnitude. Note the significant improvement in estimation using LSE updating. The dotted curve represents the voltage estimate if constant LSE is used. In both estimation approaches, the error increases with time since the load ramp moves the operating point farther away from the base case (assumed here at time t=0). The same improvement is observed for the angle estimate shown in Figure 5-18. Figure 5-19 shows the average error in the voltage magnitude estimation for the system of unobserved buses while Figure 5-20 illustrates the error for the angle. Both figures confirm the benefits of updating the interpolation coefficients for the whole system of unobserved buses.

Although the benefits gained in updating seemed to be numerically small (maximum of than 0.0023 per unit for voltage and 0.15° for the angle) the errors on the line flows are significant. LSE updating lowered the error in apparent power flow estimate at a 161-kV line (between Bus 43 to Bus 425) from 27 MW to 7 MW (see Figure 5-21). It can be seen in Figure 5-22 and Figure 5-23 that LSE updating is equally effective in reducing the estimated real and reactive flow errors.

For the total system, the benefits of updating the interpolation coefficients become evident as seen in Figure 5-24. In this figure the sum of estimation flow errors on all unobserved lines was totaled. Figure 5-25 shows that for this 444-bus 574-line system, the total error in flow estimates on unobserved lines was reduced from 2250 MW to 250 MW maximum.
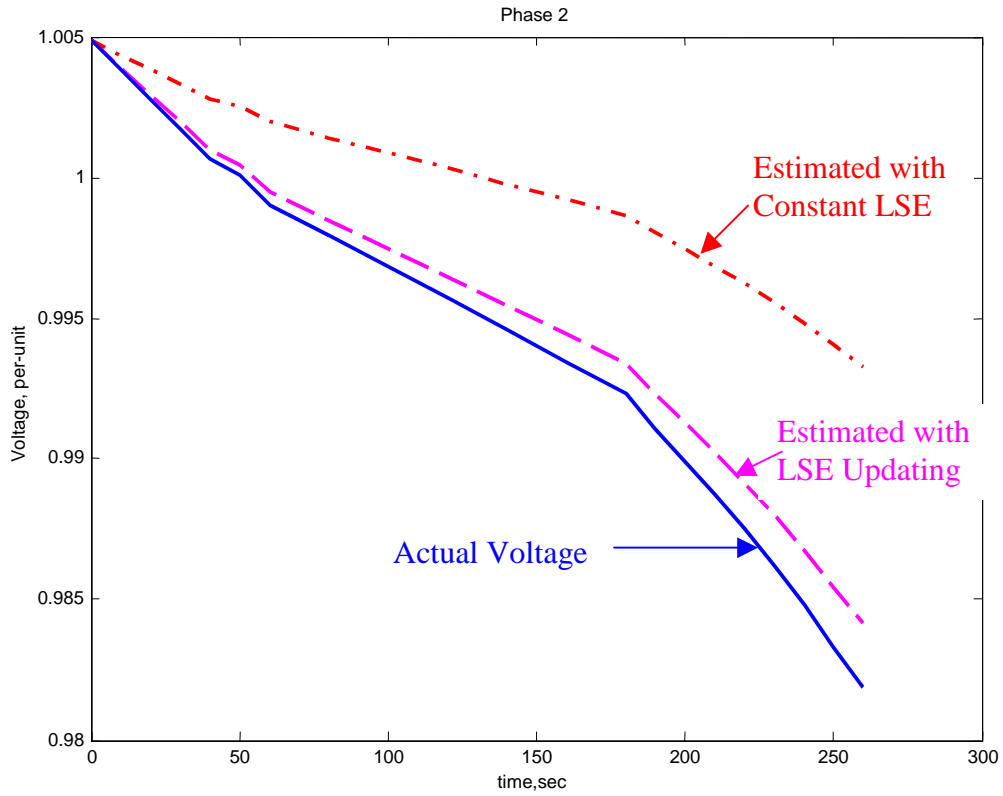
Figure 5-17. Evolution of Voltage Magnitude at Bus 243 with time: Constant vs. LSE Updating
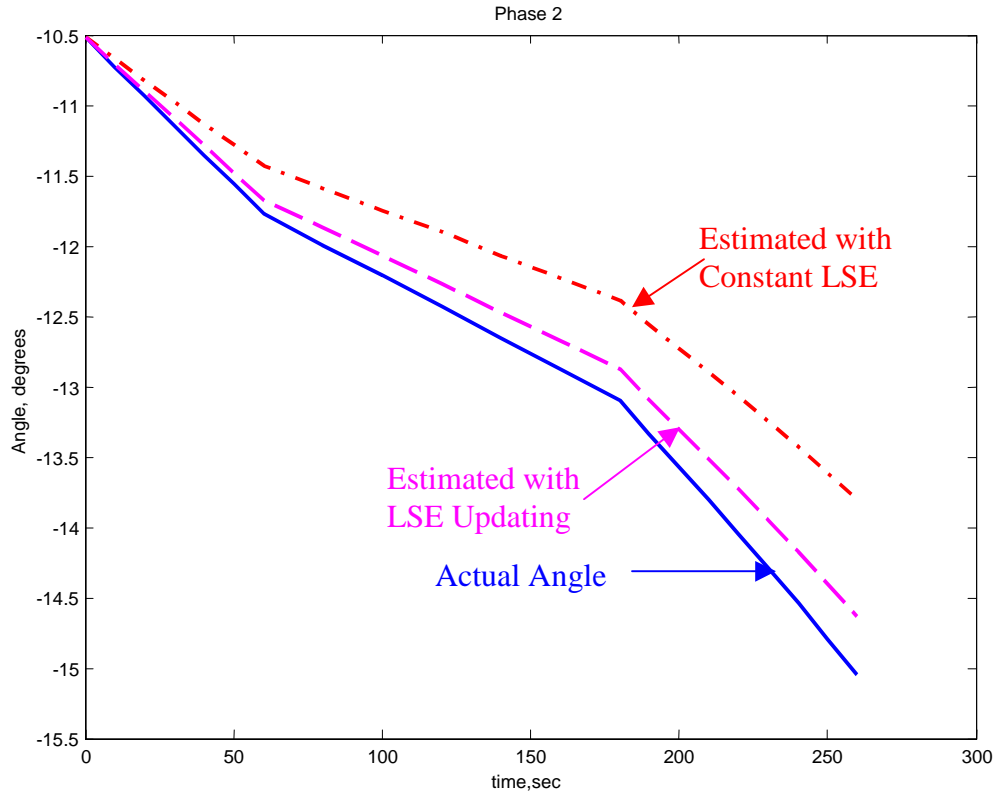
Figure 5-18. Evolution of Voltage Angle at Bus 243 with time: Constant vs. LSE
         Updating
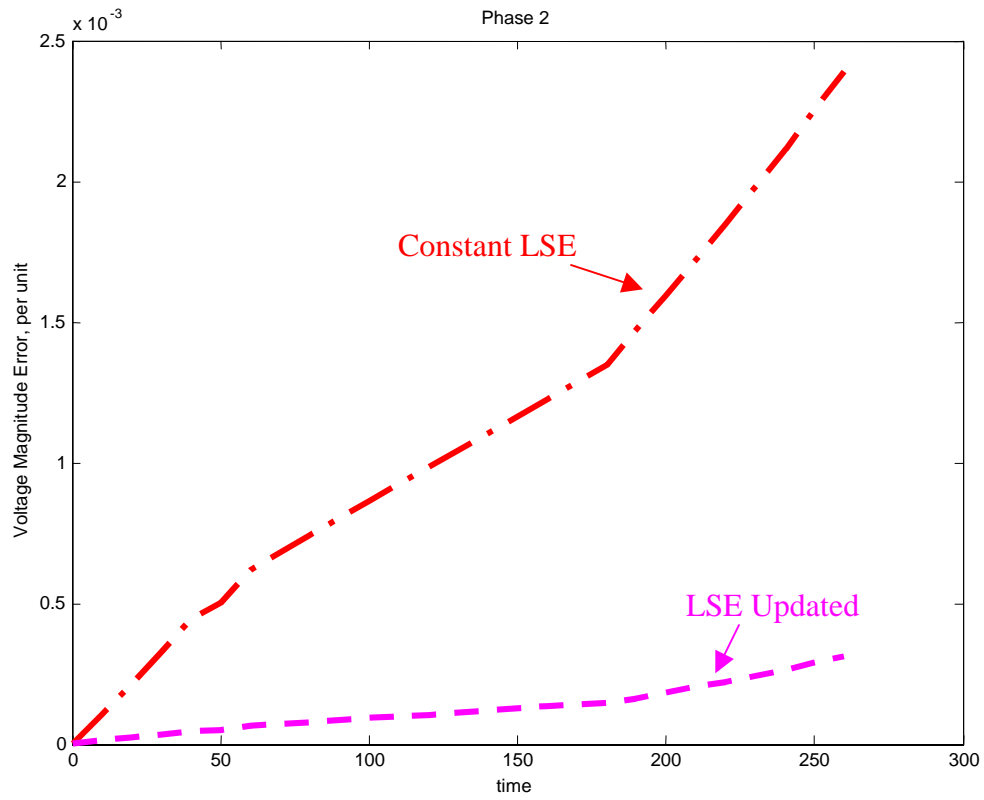
Figure 5-19. Evolution of Average System Voltage Magnitude Error: Constant vs. LSE
              Updating, Phase 2
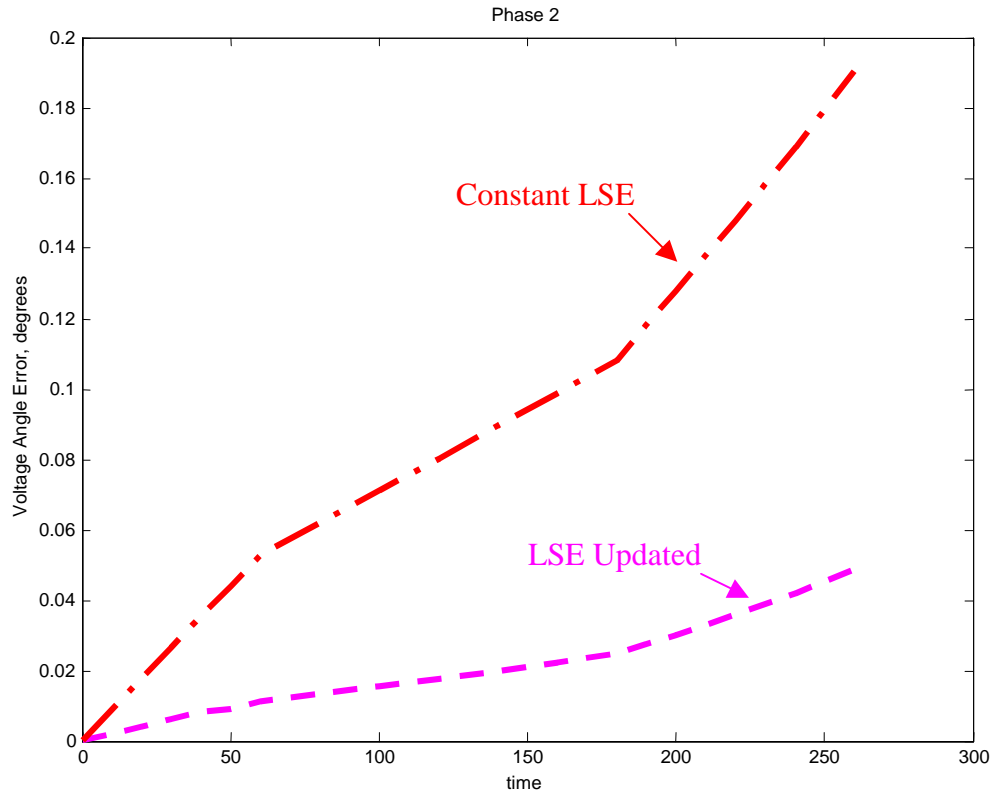
Figure 5-20. Evolution of Average System Voltage Angle Error: Constant vs. LSE
            Updating, Phase 2
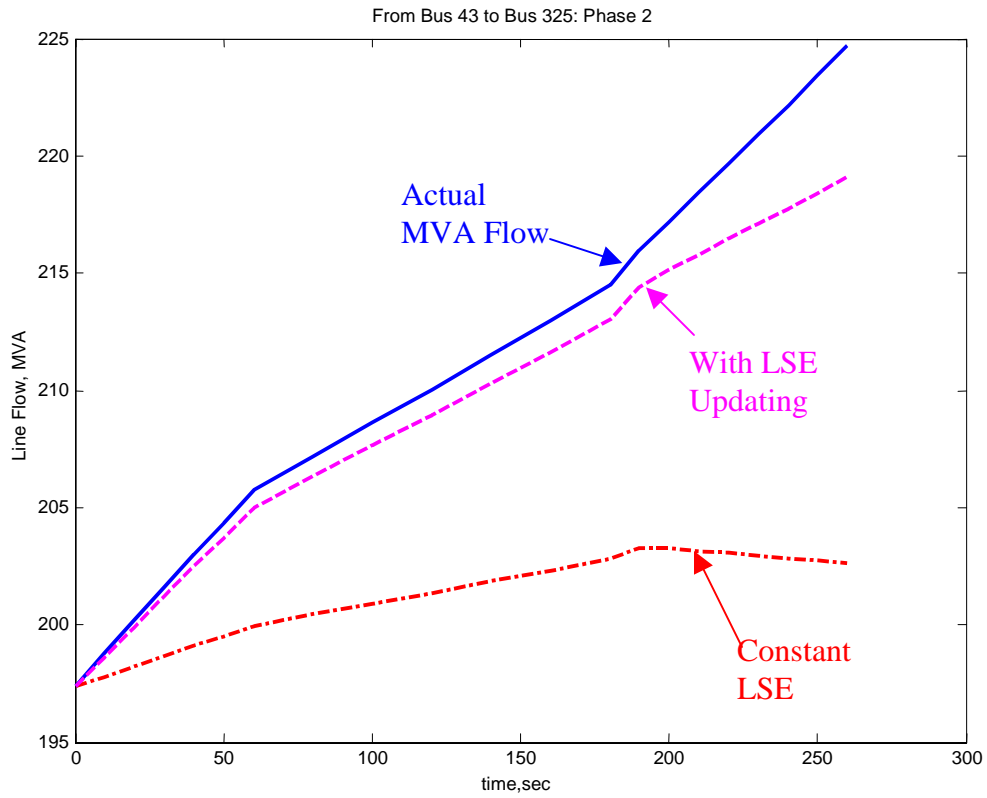
Figure 5-21. Evolution of MVA Line Flow from Bus 43 to Bus 325 (Constant vs. LSE
Updating: Phase 2
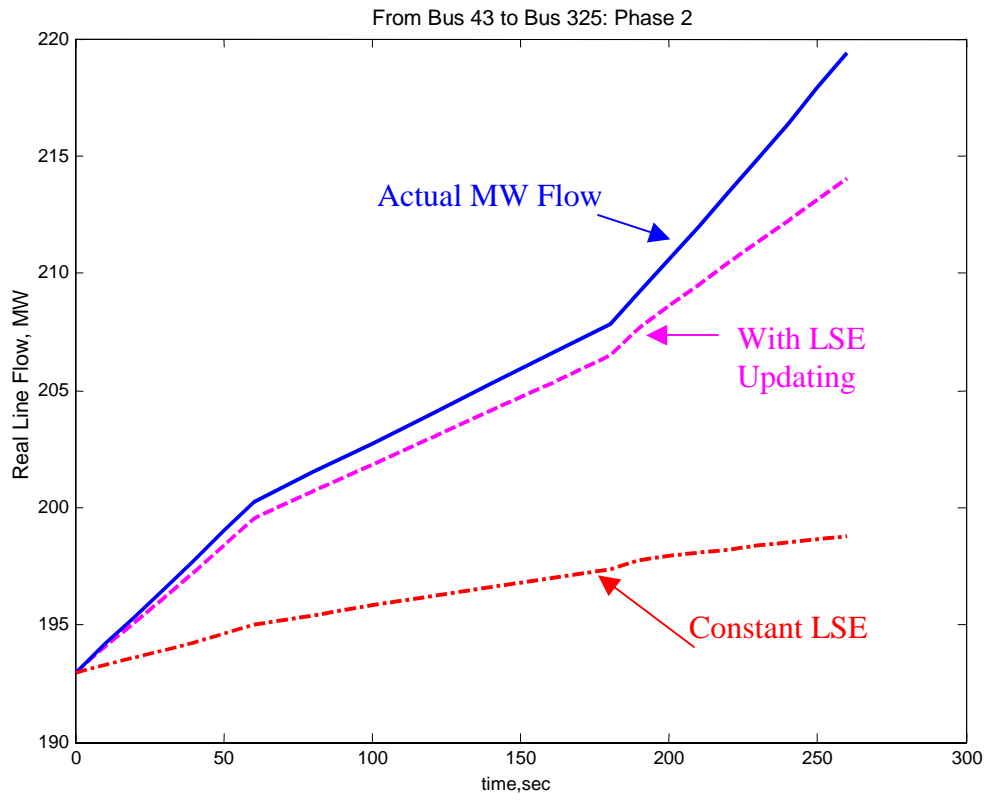
Figure 5-22. Evolution of Real Power Flow from Bus 43 to Bus 325 (Constant vs. LSE Updating: Phase 2
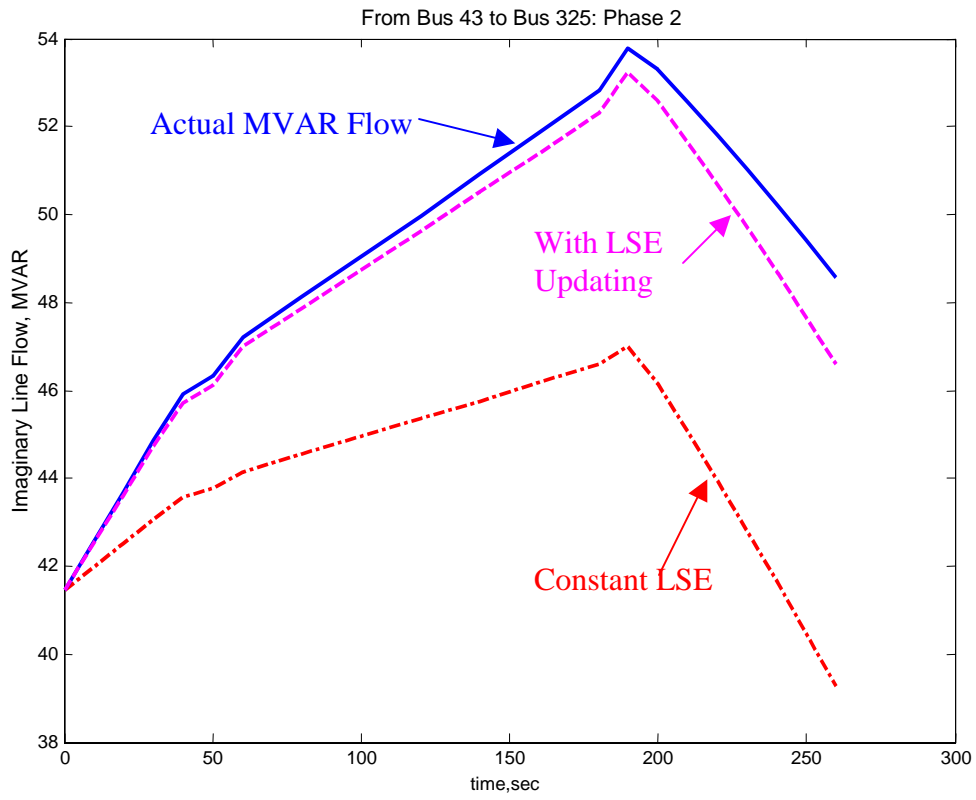
Figure 5-23. Evolution of Imaginary Power Flow from Bus 43 to Bus 325 (Constant vs. LSE Updating): Phase 2
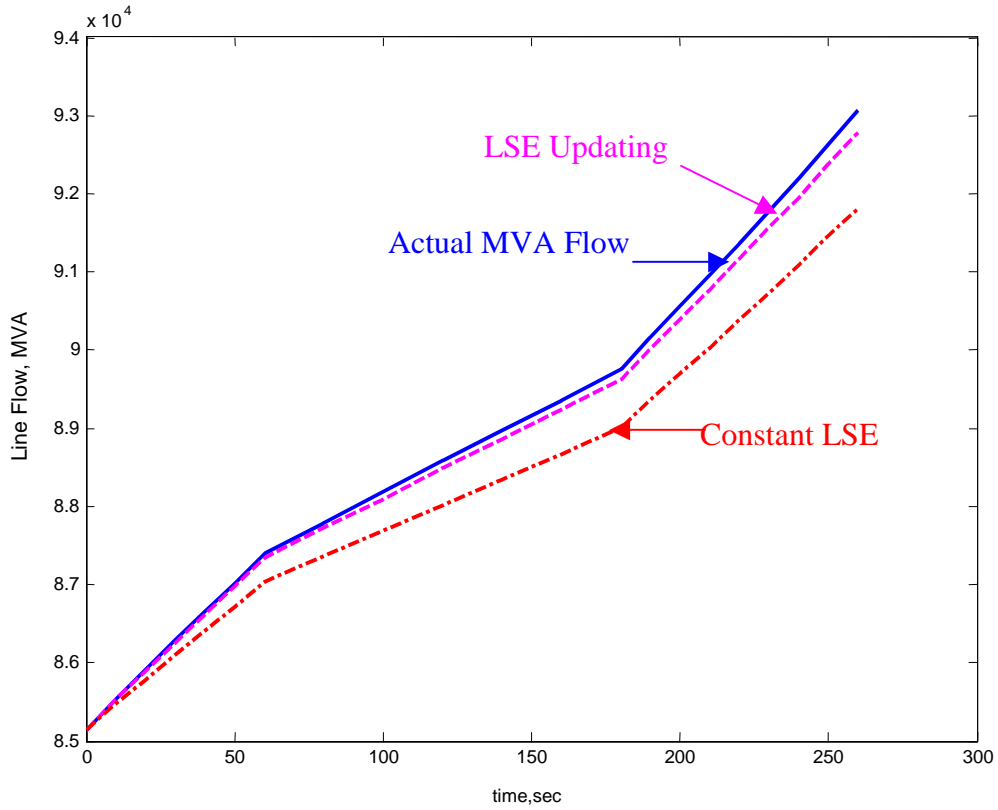
Figure 5-24. Evolution of Total MVA Flow on Unobserved Lines (Constant vs. LSE
          Updating): Phase 2
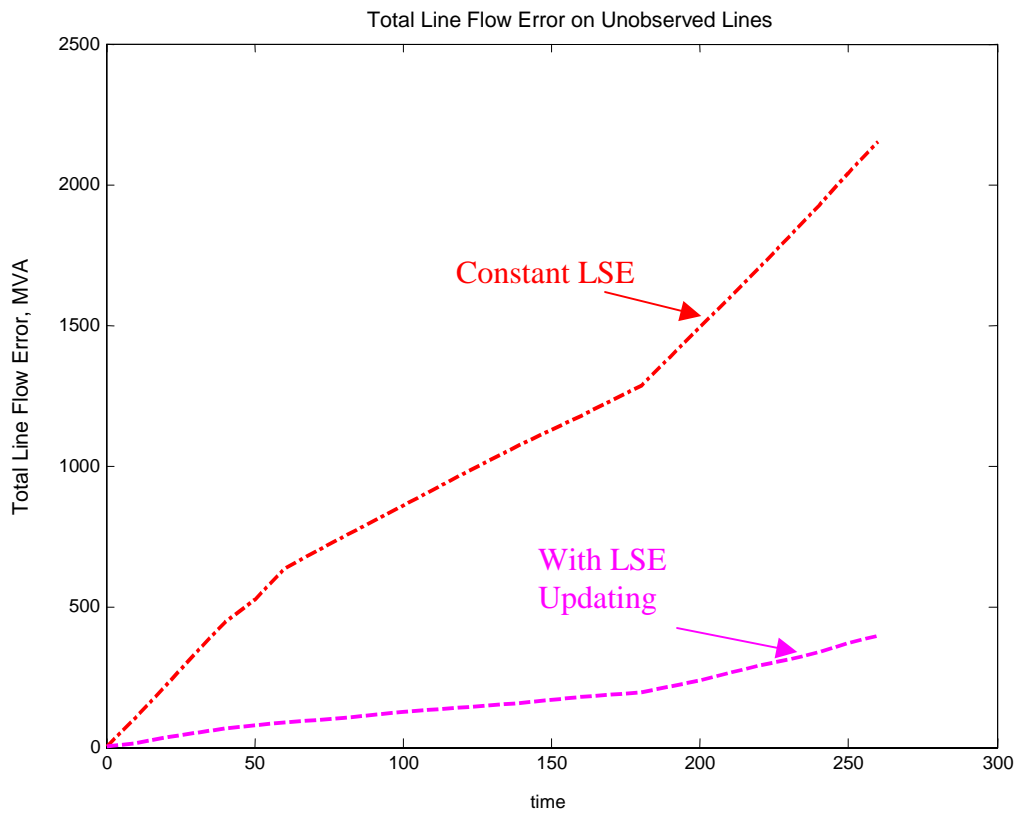
Figure 5-25. Total MVA Flow Error on Unobserved Lines (Constant vs. LSE Updating): Phase 2

Tables 5-4 to 5-12 summarize the various performance indices for voltage magnitude, angle and flows for each phase of the PMU installation at Utility System B. In general, the indices show that improvements in estimation should be expected as we migrate towards more PMU placements. More importantly, these indices show that the proposed LSE update technique using sensitivities will result in very accurate estimation of voltages and flows on the unobserved buses and lines in the power system.

Table 5-4. Voltage Magnitude Error Indices (Constant vs. LSE Updating): Phased
             Installation #1

Index	Constant H	H-Update
$\mu(V_{error})$	0.0011906 pu	0.0001411 pu
$\sigma(V_{error})$	0.0014357 pu	0.0001993 pu
Max(V_{error})	0.0101960 pu	0.0013058 pu

Table 5-5. Voltage Angle Error Indices (Constant vs. LSE Updating): Phased Installation
             #1

Index	Constant H	H-Update
$\mu(\theta_{error})$	0.099570°	0.023823°
$\sigma(\theta_{error})$	0.107789°	0.032708°
Max(θ_{error})	0.758914°	0.207820°

Table 5- 6. Voltage Magnitude Error Indices (Constant vs. LSE Updating): Phased
             Installation #2

Index	Constant H	H-Update
$\mu(V_{error})$	0.0011355 pu	0.0001329 pu
$\sigma(V_{error})$	0.0015050 pu	0.0002110 pu
Max(V_{error})	0.0101960 pu	0.0013058 pu

Table 5-7. Voltage Angle Error Indices (Constant vs. LSE Updating): Phased Installation
             #2

Index	Constant H	H-Update
$\mu(\theta_{error})$	0.0916659°	0.021692°
$\sigma(\theta_{error})$	0.1131161°	0.033735°
Max(θ_{error})	0.758914°	0.207820°

Table 5-8. Voltage Magnitude Error Indices (Constant vs. LSE Updating): Phased
         Installation #3

Index	Constant H	H-Update
$\mu(V_{error})$	0.0011866 pu	0.0001573 pu
$\sigma(V_{error})$	0.0016897 pu	0.0002402 pu
Max(V_{error})	0.0101957 pu	0.0013058 pu

Table 5-9. Voltage Angle Error Indices (Constant vs. LSE Updating): Phased Installation
         #3

Index	Constant H	H-Update
$\mu(\theta_{error})$	0.1028266°	0.026406°
$\sigma(\theta_{error})$	0.127906°	0.038251°
max(θ_{error})	0.758914°	0.207820°

Table 5-10. Power Flow Error Indices (Constant vs. LSE Updating): Phased Installation
         #1

Index	Constant H	H-Update
$\mu(F_{error})$	2.8215 MVA	0.4446 MVA
$\sigma(F_{error})$	6.9383 MVA	1.0762 MVA
Max(F_{error})	82.7330 MVA	10.1495 MVA

Table 5-11. Power Flow Error Indices (Constant vs. LSE Updating): Phased Installation
         #2

Index	Constant H	H-Update
$\mu(F_{error})$	1.8657 MVA	0.3029 MVA
$\sigma(F_{error})$	3.8875 MVA	0.6323 MVA
Max(F_{error})	36.7962 MVA	5.8185 MVA

Table 5-12. Power Flow Error Indices (Constant vs. LSE Updating): Phased Installation
         #3

Index	Constant H	H-Update
$\mu(F_{error})$	1.0511 MVA	0.2200 MVA
$\sigma(F_{error})$	2.4941 MVA	0.5463 MVA
Max(F_{error})	29.4787 MVA	5.8185 MVA

Chapter 6. Conclusions

This thesis has presented a thorough analysis of two key issues associated with the utilization of synchronized phasor measurements in power system monitoring: placement and system applications. The main objectives of the research are as follows:

1. The development of a phasor measurement unit placement for incomplete observability using graph theoretic techniques;

2. The inclusion of communication constraints in the PMU placement technique using simulated annealing algorithm.

3. The development of a real time voltage security monitoring using synchronized phasor measurements;

4. The enhancement of the traditional weighted least square state estimator using phasor measurements; and

5. The development of a streaming-type state estimator that works on incompletely observable systems.

The concept behind PMU placement algorithm for incomplete observability by itself is a fundamental contribution of this thesis. The objective is to reduce the required number of PMUs by relaxing the requirement for full observability. The algorithm involves a series of "walks" along the branches of a spanning tree of the power system graph to locate the PMU buses. The search for candidate PMU buses is terminated when all possible paths in the spanning tree are taken. We introduce the novel concept of depth of unobservability, which is a measure of the distance of any unobserved bus from a known one. Varying the desired depth of unobservability controlled the required number of PMUs. A Monte Carlo sampling of a subset of spanning trees was done to ensure that trees of varying structure were considered in the search for optimal PMU placement. The system was tested on IEEE test systems and two real world utility study regions. Results show that PMU placement for incomplete observability yielded lower number of PMUs compared to that required for complete observability. Phased installation has its practical appeal and is concerned with the batched deployment of PMUs through time. We modeled the PMU phased installation problem as an optimization problem designed for

iterative refinement techniques. Basically, the solution starts with the placement for the lowest depth of unobservability. PMUs are then removed from this set if the next higher depth of unobservability is not violated. The process is repeated until no more PMUs can be removed after a certain number of iteration. The resulting new placement then corresponds to the placement of the preceding phase. The process therefore is backtracking in time, that is, the final phase installation strategy is solved first, followed by its antecedent phase, and so on. Our phased installation algorithm was designed to decrease the system depth of unobservability with each phase of the installation.

The problem of limited communication links will continue to exist even in modern power systems. The second objective of this thesis has been achieved by the formulation of a simulated annealing algorithm for the communication constrained placement problem. We developed the required cost functions, penalty function, state transition techniques and cooling schedule to model the problem within the general framework of simulated annealing.   Central to this formulation is the mapping of a candidate placement strategy with its distance vector, that is, the physical distance of unobserved buses from those observed. The cost function is designed so that unobserved buses with lower depth of unobservability (lower distance value) contributes less to the cost compared to those buses with higher depth of unobservability. A penalty function is included to penalize deviations from the desired depth of unobservability and especially placements on buses without communication facilities. The phased installation problem was also solved within the framework of simulated annealing. The overall process was still a backtracking through time. The state transition technique was modified such that transition moves are restricted to 'fusion' moves wherein a candidate PMU for removal can only transit to another PMU bus. Finally, we model a technique to find the minimum number and location of new communication facilities if the existing communication arrangement cannot permit complete observability. This was accomplished algorithmically by temporarily lifting the communication constraint and allowing SA to find the optimal locations of new PMUs only on the unobserved portion of the system. The SA model was tested on a study region on a real world power system. SA was able to successfully execute PMU placement that recognizes communication constraints. SA also was able to evaluate if the system given its available communication sites is

observable by PMUs, and subsequently suggest the minimum number and location of new communication sites for full observability.

The third objective was accomplished by using decision trees in conjunction with PMU measurements to reliably predict the voltage security status of a power system. Devoid of floating point operations, decision trees complement the fast measuring capability of PMUs. Of the several candidate measurement attributes considered, the bus voltage angle difference among PMUs proved to be the most critical in predicting voltage insecurity. Power systems are characterized by increasing angular differences associated with stress such as loading and line contingencies. Most PMUs will be installed in EHV buses for reasons of communication, monitoring disturbances on the EHV system, etc. Because of stiff voltage regulation in EHV systems, voltage magnitude becomes a poor measure of voltage insecurity. We run 500 load flow simulations to generate the required database for building and testing the decision trees. Each loadflow run is characterized by attributes on PMU measurements (voltage and current magnitudes and angles). The bus voltage angle difference among the PMUs was calculated and augmented as an attribute. The decision trees quickly identified bus voltage angle difference as the most critical attribute. Several classification trees were then built and a technique for locating the optimal placement of another new PMU to increase the reliability of predicting voltage insecurity was presented. The model was tested on a study region in an actual utility system that is prone to voltage collapse. Test cases were submitted to several decision trees to classify. It was immediately concluded that the existing PMU system has a high misclassification rate. Then, a decision tree was built with one new PMU significantly increase the reliability of prediction. A separate DT built using var attributes from generators inside the study region performed equally well in predicting voltage insecurity. The conclusion is that, with optimal placement the bus voltage angle is equally critical in predicting voltage insecurity as generator vars.

The fourth objective was accomplished by the development of a mathematical measurement model that combines the state estimate from the weighted least squares estimator with the direct state measurement from the PMUs. The model was linear and the solution is direct and non-iterative. A WLS state estimator algorithm was developed to generate the state estimate. Then, the references of the PMU measurements and the

WLS reference were aligned before solving the linear model. The model was tested on the New England 39 Bus Test System. A total of 100 Monte Carlo simulations corresponding to random noise on the PMU and RTU measurements were run for each test case; each test case comprising differing number of PMUs. The standard deviation of the errors (from their true values) of bus voltage magnitudes and angles as well as real and reactive line flows were calculated. Analysis showed that for a system with high degree of flow and voltage metering, more PMUs or greater PMU metering accuracy is required to significantly improve the state and flow estimate. However, if the system is only partially metered the addition of even a few PMUs can significantly increase the accuracy of the state and flow estimates. There is practical appeal to this model. Some utilities need to evaluate the gains in state estimation brought about by PMUs without going to the laborious process of reprogramming their state estimation package. This model that works only on the results of the WLS estimator and PMU measurements provides the solution to this problem.

The last objective was accomplished through the development of a hybrid estimator that utilizes the output of the traditional WLS SE to enhance the linear PMU-based SE. Because of limitations in the number of communication sites, power systems cannot be completely observed by PMUs in the interim. It was shown that using the bus admittance matrix that the voltages of unobserved buses could be interpolated from the observed buses. The technique consisted of converting the injections at the unobserved buses into apparent admittances and developing a matrix of interpolation coefficients or linear state estimators (LSE). Tests have shown that the accuracy of interpolation degrades as the system moves far from the base case operating point. A mathematical model for updating the interpolation coefficients based on their sensitivities to the unobserved bus injections was formulated. These sensitivities are functions of the bus admittance elements and the change in injections at the unobserved buses. To implement this model, we propose to utilize the state from the existing WLS estimator to calculate the injections at unobserved buses. This means that we can get a periodic update of the interpolation coefficients every measurement scan of the traditional WLS SE. The model was tested on Utility System B. Test results show that with the hybrid estimator, a significant improvement in the estimation of unobserved bus voltages as well as line flows on

unobserved lines was achieved. Comparing the estimates using the hybrid model with the constant LSE model arrived at this conclusion.   The significance of this accomplished objective is that now it is possible to execute streaming state estimation, that is, a continuous stream of synchronized power system state arriving at a very fast rate, on systems that are not completely observable by PMUs.

Future Research

The following research is recommended for future work.

1. Investigate the issue of graph partitioning and doing PMU placement on the partitions. For very large systems this could prove to be computationally efficient. The basic idea is to optimize placement on each partition but subject to coupling constraints between partitions.

2. Investigate the possibility of utilizing dynamic programming on the communication constrained PMU placement.

3. Research on the use of regression trees instead of classification trees on the voltage security monitoring system. As such, it is possible to predict the margin to voltage collapse or the margin to voltage insecurity. The nodes in a regression tree are closed form expressions of the attributes and some measure, such as security margins. Also further research is recommended on the use of surrogate attributes on decision trees that are linear combinations of several attributes. It was reported in some cases that these surrogate attributes could result in better DT classifiers.

4. Communication is a critical issue that needs to be resolved. Research must be conducted on the quality of communication required by streaming state estimation. Such requirements as bandwidth or expected communication delays and communication errors need to be determined and appropriately modeled with the hybrid state estimator.

References

[1]   V. Ajjarapu, C. Christy, "The Continuation Power Flow: A tool for Steady-State Voltage Stability Analysis", *IEEE Transactions on Power Systems,* Vol. 7, February 1992, pp. 416-423.

[2]   V. Ajjarapu, B. Lee, "Bibliography on Voltage Stability", *IEEE Transactions on Power Systems,* Vol. 13, No.1, February 1998, pp.115-125.

[3]   J. J. Alemong et al, "A Fast and Relible State Estimation Algorithm for AEP's New Control Center", *IEEE Transactions on PAS*, Vol. 101, No. 4, April 1982, pp. 933-944.

[4]   ANSI/IEEE Std. C57.13, "IEEE Standard Requirements for Instrument Transformers", 1993.

[5]   T. L. Baldwin, L. Mili, M. B. Boisen, R. Adapa, "Power System Observability with Minimal Phasor Measurement Placement", *IEEE Transactions on Power Systems,* Vol.8, No. 2, May 1993, pp. 701-715.

[6]   T. L. Baldwin, "Real-Time Phasor Measurements for Improved Monitoring and Control of Power System Stability, Ph.D. Thesis, *Virginia Polytechnic Institute and State University*, May 1993.

[7]   D. E. Barber, "Applications of Phasor Measurements to Real-Time Monitoring of a Power System", Master of Science Thesis, *Virginia Polytechnic Institute and State University*, March 1994.

[8]   M. Begovic, A.G. Phadke, "Control of Voltage Stability using Sensitivity Analysis", *IEEE Transactions on Power Systems*, Vol.7, Feb. 1992, pp.114-123.

[9]   L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, "*Classification and Regression Trees",* Wadsworth, Inc., Belmont, California, 1984.

[10]   D. J. Brueni, "Minimal PMU Placement for Graph Observability", Master of Science Thesis, *Virginia Polytechnic Institute and State University*, December 1993.

[11] R. O. Burnett, M. M. Butts, S. S. Sterlina, "Power system Applications for Phasor Measurement Units", *IEEE Computer Applications in Power*, Janury 1994, pp. 8-13.

[12] R.O. Burnett, M.M Butts, T.W. Cease, V. Centeno, G. Michel, R.J. Murphy, A.G. Phadke, "Synchronized Phasor Measurement of a Power System Event", *IEEE Transactions on Power Systems*, Vol. 9, No. 3, August 1994, pp. 1643-1649.

[13] C.A. Cañizares, "On Bifurcations, Voltage Collapse and Load Modeling", *IEEE Transactions on Power Systems,* Vol. 10, No.1, February 1995, pp. 512-520.

[14] C.A.Cañizares and F.L. Alvarado, "Point of Collapse and Continuation Methods for Large AC/DC Systems," *IEEE Transactions on Power Systems,* Vol.8, No. 1, February 1993, pp. 1-8.

[15] V. Centeno, J. De la Ree, A.G. Phadke, G. Michel, J. Murphy, R. Burnett, "Adaptive Out-of-Step Relaying Using Phasor Measurement Techniques," IEEE Computer Applications in Power Magazine, October 1993.

[16] M. Chau, R. Gutman, B.M. Pasternack, "Understanding Voltage Collapse in Bulk Power Transmission Systems," in [24], pp. 3-45 to 3-55.

[17] H.K. Clark, R.K. Gupta, C. Loutan, D.R. Sutphin, "Experience with Dynamic System Monitors to Enhance System Stability Analysis", *IEEE Transactions on Power Systems,* Vol. 7, No. 2, May 1992.

[18] K.A. Clements, G.R. Krumpholz, P.W. Davis, "Power System State Estimation with Measurement Deficiency: An Observability/Measurement Placement Algorithm", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-102, No.7, July 1983, pp. 2012-2020.

[19] T. Van Cutsem, L.Wehenkel, M.Pavella, B. Heilbronn, and M. Goubin. "*Decision Tree Approaches to Voltage Security Assessment*", IEE Proceedings-Part C, 140:189-198, 1993.

[20] T. Van Cutsem, C. Vournas, "*Voltage Stability of Electric Power Systems",* Kluwer Academic Publishers, 1998.

[21] P. Denys, C. Coundan, L. Hossenlopp, C. Holweck, " Measurement of Voltage Phase for the French Defense Plan Against Losses of Synchronism", *IEEE Transactions on Power Delivery*, Vol. 7, No.1, January 1992.

[22] J. F. Dopazo, S. T. Ehrmann, O. A. Klitin, A. M. Sasson, L.S. van Slyck, "Implementation of the AEP Real-Time Monitoring System", *IEEE Transactions on PAS*, Vol. 95, No. 5, Sept/Oct 1976, pp. 1618-1629

[23] G.C. Ejebe, G.D. Irisarri, S. Mokhtari, O. Obadina, P. Ristanovic, J. Tong, "Methods for Contingency Screening and Ranking for Voltage Stability Analysis of Power Systems", *IEEE Transactions on Power Systems*, vol. 11, February 1996, pp. 350-356.

[24] L.H. Fink, ed., Proceedings: Bulk Power System Voltage Phenomena – Voltage Stability and Security, EPRI EL-6183, January 1989.

[25] B.Gao, G.K. Morrison, P. Kundur, "Towards the Development of a System Approach for Voltage Stability Assessment of Large-Scale Power Systems", in *IEEE Summer Power Meeting,* Paper 95 SM 526-4 PWRS.

[26] B.Gao, G.K. Morrison, and P. Kundur "Voltage Stability Evaluation Using Modal Analysis", *IEEE Transactions on Power Systems*, vol.7, Nov. 1992, pp.1529-1542.

[27] GE Power Systems Engineering, "Assessment of Applications and Benefits of Phasor Measurement Technology in Power Systems", *EPRI Final Report*, April 1997.

[28] S. Greene, I. Dobson, F.L. Alvarado, "Sensitivity of Loading Margin to Voltage Collapse with Respect to Arbitrary Parameters", *IEEE PES Winter Power Meeting,* 96 WM 287-2-PWRS.

[29] S. Greene, I. Dobson, F.L. Alvarado, "Contingency Ranking for Voltage Collapse via Sensitivities from a Single Nose Curve", *IEEE Transactions on Power Systems,* vol. 14, No.1, February 1999, pp. 232-240.

[30] H. W. Hale, "A Logic for Identifying the Trees of a Graph", *Paper 61-54 presented at the AIEE Winter General Meeting*, New York, N.Y, January 1961, pp. 195-198.

[31]   E. Handschin et al, "Bad Data Analysis for Power System State Estimation", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-94. No. 2, March/April 1975, pp. 329-337.

[32]   IEEE Working Group H-7, "Synchronized Sampling and Phasor Measurements for Relaying and Control", *IEEE Transactions on Power Delivery*, Vol. 9, No.1, January 1994, pp. 442-452.

[33]   IEEE Working Group H-8, "IEEE Standard for Synchrophasors for Power Systems", *IEEE Transactions on Power Delivery*, Vol. 13, No. 1, January 1998.

[34]   IEEE/PES Voltage Stability Report at website *http://iliniza.uwaterloo.ca/~claudio/claudio.html*

[35]   J.S.R. Jang, C.T. Sun, E. Mizutani, *"Neuro-Fuzzy and Soft Computing",* Prentice Hall, 1997.

[36]   S. Kabra, A Technique for Estimating System-Wide Phasors in Real Time, MS Thesis, Virginia Tech, January 1997.

[37]   I. Kamwa, R. Grondin, Y. Hebert, "Wide-Area Measurement Based Stabilizing Control of Large Power Systems – A Decentralized/Hierarchical Approach", *IEEE Transactions on Power Systems,* Vol. 16, No.1, February 2001, pp.136-153.

[38]   K.J. Karimi, J.S. Thorp and A.G. Phadke, " Partitioned State Estimators and Bad Data Processing for Static State Estimators with Phasor Measurements", *Proceedings of the 1986 North American Power Symposium, Ithaca, New York*, October 1986, pp.131-140.

[39]   G.R. Krumpholz, K.A. Clements, P.W. Davis, "Power System Observability: A Practical Algorithm Using Network Topology", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-99, No.4, July/Aug 1980, pp. 1534-1542.

[40]   P. Kundur, *Power System Stability and Control*, Mc Graw-Hill, New York 1994.

[41]   P.J.M. van Laarhoven, E.J.L. Aarts, *"Simulated Annealing: Theory and Applications"*, D. Reidel Publishing Company, 1987.

[42]   P.Lof , T. Smed, G. Anderson, J. Hill, "Fast Calculation of Voltage Stability
        Index", *IEEE Transactions on Power Systems*, Vol. 7, Feb. 1992, pp. 54-64.

[43]   Hyde M. Merrill, Fred C. Schweppe, "Bad Data Suppression in Power System
        Static State Estimation, *IEEE Transactions in PAS*, Vol. PAS-90. No. 6, Nov/Dec
        1971.

[44]   L. Mili et al, "Bad Data Identification Methods in Power System State Estimation
        – A Comparative Study, "*IEEE Transactions on PAS,* Vol. 104, No. 11, November
        1985, pp. 3037-3049.

[45]   E. Minieka, "*Optimization Algorithms for Networks and Graphs*", Marcel Dekker,
        Inc., 1978.

[46]   A. Monticelli, *Generalized State Estimation,* Kluwer Academic Publishers, 1999.

[47]   R. F. Nuqui, A.G. Phadke, R.P. Schulz, N.B. Bhatt, "Fast On-Line Voltage
        Security Monitoring Using Synchronized Phasor Measurements and Decision
        Trees", Paper presented at the *IEEE PES 2001 Winter Power Meeting,* 01 WM
        287-2-PWRS.

[48]   Y. Ohura, M. Suzuki, K. Yanagihashi, M. Yamaura, K. Omata, T. Nakamura, S.
        Mitamura, H. Watanabe, "A Predictive Out-of-Step Protection System Based on
        Observation of Phase Angle Difference Between Substations", *IEEE Transactions
        on Power Delivery,* Vol. 5, No. 4, November 1990, pp. 1695-1704.

[49]   E.W. Palmer, G. Ledwich, "Optimal Placement of Angle Transducers in Power
        Systems", *IEEE Transactions on Power Systems,* Vol. 11, No.2, May 1996, pp
        788-793.

[50]   A.G. Phadke, "*Synchronized Phasor Measurements in Power Systems*," IEEE
        Computer Applications in Power, April 1993.

[51]   A.G. Phadke and J. Thorp*, Computer Relaying for Power Systems*, Research
        Studies Press Ltd., 1988.

[52]   A.G. Phadke, J.S. Thorp, K.J. Karimi, "State Estimation with Phasor Measurements", *IEEE Transaction on PWRS*, Vol. 1, No. 1, February 1986, pp 233-241.

[53]   A.G. Phadke, J.S. Thorp and K.J. Karimi, " Real Time Voltage Phasor Measurements for Static State Estimation", *IEEE Transactions on PAS*, Vol. 104, No. 11, November 1985, pp.3098-3107.

[54]   A. G. Phadke, J. S. Thorp, M. G. Adamiak, "A New Measurement Technique for Tracking Voltage Phasors, Local System Frequency, and Rate of Change of Frequency", *IEEE Transactions on PAS,* Vol. 102, No. 5, May 1983, pp. 1025-1038.

[55]   M.D. Potter *Sets: An Introduction*, Oxford University Press, 1990.

[56]   P.W. Sauer, M.A. Pai, *Power System Dynamics and Stability*, Prentice Hall, NJ, 1998.

[57]   R.P. Schulz, L.S. Van Slyck, S.H. Horowitz, "Applications of Phasor Measurements on Utility Systems", *Proceedings of the IEEE PICA Conference,* Paper 89-CH-2747-4, May 1989, pp. 49-55.

[58]   F.C. Schweppe, E. J. Handschin, "Static State Estimation in Electric Power Systems", *Proceedings of the IEEE*, Vol. 62, No. 7, July 1974.

[59]   F.C. Schweppe, J. Wildes, "Power System Static State Estimation Part I: Exact Model", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-89, No. 1, January 1970, pp. 120-125.

[60]   F.C. Schweppe, D. B. Rom, "Power System Static State Estimation Part I: Approximate Model", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-89, No. 1, January 1970, pp. 125-130.

[61]   F.C. Schweppe "Power System Static State Estimation Part I: Implementation", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-89, No. 1, January 1970, pp. 130-135.

[62] I.W. Slutsker, S. Mokhtari, L.A. Jaques, J.M.G. Provost, M.B. Perez, J.B. Sierra, F.G. Gonzalez, J.M.M. Figueroa, "Implementation of phasor measurements in state estimator at Sevillana de Electricidad", *Power Industry Computer Application Conference, 1995. Conference Proceedings, 1995*, pp. 392 –398.

[63] A.F. Snyder et al, "A Robust Damping Controller for Power Systems using Linear Matrix Inequalities", *Proceedings of the IEEE-PES Winter Meeting*, New York, New York, January 31 – February 4, 1999, pp. 519-524.

[64] A.F. Snyder et al, "Delayed-Input Wide-Area Stability Control with Synchronized Phasor Measurements and Linear Matrix Inequalities", *Proceedings of the IEEE-PES Summer Meeting*, Seattle, Washington, July 16-20, 2000, pp. 1009-1014.

[65] C. Taylor, *Power System Voltage Stability*, Mc-Graw Hill, 1994.

[66] L.A. Wehenkel, *Automatic Learning Techniques in Power Systems*, Kluwer Academic Publishers, 1997.

[67] A.J. Wood, B. Wollenberg, *Power Generation, Operation and Control*, John Wiley and Sons, Inc., 2nd edition, 1996.

[68] F. Zhuang, F. D. Galiana, "Unit Commitment by Simulated Annealing", *IEEE Transactions on Power Systems,* Vol. 5, No. 1, February 1990, pp. 311-318.

Appendix A

TEST SYSTEMS

This appendix contains the system description of the various test systems used in this thesis. Two types of system data will be presented. Test systems used solely for the PMU placement algorithms will be presented in terms of their line p-q list. Systems that were utilized for the state estimation models will be presented as solved load flow cases. The load flow line data will provide the topology needed for PMU placement for systems utilized both in the placement and state estimation models.

Utility System A is the study region behind the voltage security monitoring model presented in Chapter 4. The load flow model used to generate the measurement cases to build and test the decision trees however came from a much larger 6000+ interconnected network. Due to its size the load flow data is not listed here.

Utility System B is the subject of PMU placement algorithms, simulated annealing and the proposed hybrid state estimation. It is a 444-bus, 574-line subsystem that we extracted from a large 8000+ solved load flow case. Load flow bus and line data are reported in this appendix. Being a subsystem, the net injection at any bus generally does not match the bus load and generation data. The base case net real and reactive power injection at any bus is reported here.

The rest of the test systems are standard IEEE-14, IEEE-30, and IEEE-57 bus test system. They are reported as line p-q lists. The New England 39-bus test system is reported in a solved load flow format since it is the subject of the enhanced WLS state estimation algorithm presented in Chapter 5.

Table A-1. IEEE 14 Bus Test System P-Q List

ID	FROM BUS	TO BUS
1	1	2
2	1	5
3	2	3
4	2	4
5	2	5
6	3	4
7	4	5
8	4	7
9	4	9
10	5	6
11	6	11
12	6	12
13	6	13
14	7	8
15	7	9
16	9	10
17	9	14
18	10	11
19	12	13
20	13	14

Table A-2. IEEE 30 Bus Test System Line P-Q List

ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS
1	1	2	33	24	25
2	1	3	34	25	26
3	2	4	35	25	27
4	3	4	36	28	27
5	2	5	37	27	29
6	2	6	38	27	30
7	4	6	39	29	30
8	5	7	40	8	28
9	6	7	41	6	28
10	6	8			
11	6	9			
12	6	10			
13	9	11			
14	9	10			
15	4	12			
16	12	13			
17	12	14			
18	12	15			
19	12	16			
20	14	15			
21	16	17			
22	15	18			
23	18	19			
24	19	20			
25	10	20			
26	10	17			
27	10	21			
28	10	22			
29	21	22			
30	15	23			
31	22	24			
32	23	24			

Table A- 3. IEEE 57 Bus Test System Line P-Q List

ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS
1	1	2	36	24	25	71	11	43
2	2	3	37	24	26	72	44	45
3	3	4	38	26	27	73	40	56
4	4	5	39	27	28	74	56	41
5	4	6	40	28	29	75	56	42
6	6	7	41	7	29	76	39	57
7	6	8	42	25	30	77	57	56
8	8	9	43	30	31	78	38	49
9	9	10	44	31	32	79	38	48
10	9	11	45	32	33	80	9	55
11	9	12	46	34	32			
12	9	13	47	34	35			
13	13	14	48	35	36			
14	13	15	49	36	37			
15	1	15	50	37	38			
16	1	16	51	37	39			
17	1	17	52	36	40			
18	3	15	53	22	38			
19	4	18	54	11	41			
20	4	18	55	41	42			
21	5	6	56	41	43			
22	7	8	57	38	44			
23	10	12	58	15	45			
24	11	13	59	14	46			
25	12	13	60	46	47			
26	12	16	61	47	48			
27	12	17	62	48	49			
28	14	15	63	49	50			
29	18	19	64	50	51			
30	19	20	65	10	51			
31	21	20	66	13	49			
32	21	22	67	29	52			
33	22	23	68	52	53			
34	23	24	69	53	54			
35	24	25	70	54	55			

Table A- 4. Utility System A Line P-Q List

ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS
1	1	68	41	19	87	81	45	140
2	1	241	42	19	259	82	45	264
3	2	140	43	19	105	83	46	186
4	3	74	44	20	259	84	46	217
5	3	102	45	20	265	85	46	232
6	5	79	46	21	22	86	47	58
7	5	80	47	21	57	87	47	219
8	5	18	48	22	14	88	48	140
9	5	201	49	23	48	89	49	50
10	5	202	50	24	65	90	50	255
11	5	242	51	26	84	91	50	260
12	5	252	52	28	104	92	51	70
13	5	264	53	28	212	93	51	143
14	6	237	54	29	220	94	52	272
15	6	141	55	31	70	95	53	117
16	7	78	56	31	267	96	53	212
17	7	173	57	32	103	97	54	137
18	8	111	58	32	207	98	55	181
19	8	242	59	34	66	99	55	197
20	8	126	60	34	207	100	56	172
21	10	216	61	35	81	101	56	207
22	10	233	62	35	110	102	57	147
23	11	133	63	35	144	103	57	262
24	11	276	64	35	180	104	59	189
25	12	82	65	36	64	105	60	218
26	12	162	66	36	223	106	61	60
27	12	245	67	36	233	107	62	63
28	13	12	68	37	38	108	63	154
29	13	136	69	38	36	109	63	240
30	15	145	70	38	136	110	64	208
31	15	159	71	39	113	111	65	212
32	15	191	72	39	221	112	66	225
33	15	274	73	40	183	113	67	110
34	16	263	74	41	255	114	68	77
35	17	16	75	42	52	115	68	105
36	17	78	76	42	70	116	68	150
37	17	38	77	43	257	117	68	164
38	19	26	78	44	83	118	68	226
39	19	54	79	44	166	119	69	68
40	19	76	80	45	60	120	69	101

Table A-4. Continued

ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS
121	70	128	161	98	219	201	125	126
122	70	169	162	99	161	202	125	261
123	71	70	163	99	228	203	125	25
124	71	158	164	99	258	204	127	274
125	72	71	165	99	269	205	130	271
126	73	71	166	101	231	206	130	14
127	73	136	167	103	183	207	131	174
128	73	139	168	104	258	208	131	199
129	74	95	169	106	177	209	132	251
130	74	108	170	107	116	210	132	185
131	74	170	171	107	231	211	133	200
132	74	224	172	108	143	212	134	193
133	74	273	173	108	170	213	135	136
134	75	99	174	108	197	214	136	134
135	75	162	175	108	276	215	137	246
136	76	118	176	109	138	216	137	130
137	77	122	177	109	168	217	138	186
138	78	275	178	110	123	218	139	138
139	79	166	179	112	212	219	140	206
140	80	58	180	112	254	220	140	27
141	82	93	181	113	207	221	140	35
142	82	211	182	114	214	222	140	144
143	83	262	183	115	192	223	141	140
144	84	257	184	115	202	224	141	158
145	85	226	185	116	155	225	142	262
146	87	68	186	116	234	226	145	250
147	89	138	187	117	157	227	146	157
148	89	186	188	117	243	228	146	181
149	90	149	189	118	236	229	148	238
150	91	210	190	119	130	230	149	262
151	92	220	191	119	33	231	151	232
152	92	222	192	120	265	232	152	239
153	93	91	193	121	120	233	153	244
154	94	129	194	121	153	234	154	184
155	94	270	195	122	177	235	154	198
156	95	97	196	123	214	236	154	274
157	95	266	197	123	108	237	156	271
158	96	150	198	124	171	238	157	272
159	96	223	199	124	187	239	158	157
160	97	227	200	124	249	240	159	260

Table A-4. Continued

ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS	ID	FROM BUS	TO BUS
241	161	245	281	218	264	321	14	130
242	164	223	282	223	255	322	14	43
243	165	266	283	237	238	323	247	137
244	167	182	284	237	239	324	247	240
245	167	251	285	237	141	325	280	8
246	168	207	286	238	4	326	280	126
247	169	213	287	33	240			
248	171	226	288	246	240			
249	172	212	289	252	264			
250	174	253	290	253	248			
251	175	199	291	256	140			
252	175	274	292	256	250			
253	176	185	293	262	268			
254	177	205	294	263	262			
255	177	270	295	275	274			
256	179	178	296	24	157			
257	179	177	297	41	277			
258	180	253	298	29	277			
259	182	222	299	102	278			
260	184	261	300	269	278			
261	185	274	301	106	187			
262	186	194	302	188	124			
263	186	221	303	188	229			
264	187	229	304	200	134			
265	189	201	305	227	97			
266	191	234	306	210	12			
267	192	268	307	216	276			
268	193	273	308	224	74			
269	195	248	309	243	224			
270	196	232	310	269	278			
271	198	203	311	18	152			
272	201	230	312	29	277			
273	203	240	313	59	47			
274	204	239	314	59	58			
275	204	239	315	279	154			
276	205	229	316	279	127			
277	206	219	317	152	204			
278	209	228	318	152	18			
279	212	267	319	14	268			
280	213	232	320	14	16			

Table A- 5. Utility System B Load Flow Bus Data

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
1	1	0	0.02	-0.12	1.0401	14.581
2	1	0	-379.2	-20.53	1.0384	13.435
3	1	0	0.06	0.25	1.0366	14.741
4	1	0	402.03	43.27	1.043	19.979
5	1	0	134.06	-8.41	1.043	16.591
6	1	0	134.07	-7.17	1.043	16.583
7	1	0	134.03	43.54	1.043	15.871
8	1	0	268.13	54.73	1.043	16.171
9	1	0	274.36	0.13	1.043	20.036
13	1	0	-7.51	-2.8	1.0094	-0.8337
15	1	0	-0.01	0.06	1.04	13.571
16	1	0	-110.4	129.38	1.0361	10.01
17	1	0	0.01	-0.09	1.0544	8.821
18	1	0	-108.5	158.47	1.0492	4.2925
19	1	0	-0.03	0.14	1.0472	5.3333
20	1	0	-538.1	44.52	1.0458	1.5085
21	1	0	376.68	-132.8	1.039	2.7928
22	1	0	-318.9	133.76	1.0385	-1.5542
23	1	0	521.59	-192.8	1.0331	-0.0296
24	1	0	-616.3	82.88	1.0368	-3.8453
25	1	0	0	343.55	1.042	0.8765
26	1	0	-618.4	-53.96	1.0375	-5.2513
27	1	0	-277.4	59.82	1.0453	-4.3493
28	1	0	-87.2	48.19	1.0211	-3.7209
30	1	0	570.43	128.52	1.056	4.1431
33	1	0	0.06	0.27	1.0555	2.2438
34	1	0	-161.4	-41.41	1.0423	-1.0417
35	1	0	331.08	-86.03	1.0544	4.176
36	1	0	-299.5	12.43	1.046	0.4637
37	1	0	0.02	0.01	1.0387	11.121
38	1	0	129.96	14.36	1.056	7.3223
39	1	0	182.87	37.68	1.056	7.3223
40	1	0	2546.2	-139.3	1.0339	14.996
41	1	0	-26.85	-15.87	1.0273	2.9332
42	1	0	1020	-7.32	1.02	13.775
43	1	0	682.87	116.4	1.043	8.2713
44	1	0	-0.05	-0.08	1.0139	6.9973

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt	P Injection (MW)	Q Injection (MVAR)	Magnitude	Angle
45	1	0	-376.4	-13.32	1.0426	2.9047
48	1	0	0.08	0.12	1.0186	6.4298
49	1	0	-23.63	-2.83	1.0411	1.7865
50	1	0	-0.02	0.07	1.0286	4.7785
51	1	0	-351.6	-100.6	1.0437	1.3911
52	1	0	1926.5	-383.8	1.05	6.3542
53	1	0	-54	-28.97	1.0261	0.3298
54	1	0	146.82	27.37	1.0488	5.9568
55	1	0	-106.2	128.47	1.0409	2.2587
57	1	0	-132.3	-14.72	1.0313	-0.5407
58	1	0	153.6	-33.67	1.0558	4.417
59	1	0	-221.4	-31.1	1.0402	0.8807
60	1	0	0.02	-0.09	1.0522	4.8116
61	1	0	-248.8	153.09	1.0479	-0.9668
62	1	0	0	0	1.0639	5.845
63	1	0	0.01	0.06	1.0639	5.845
64	2	0	14	-0.03	1.037	1.5646
65	1	0	-0.01	-0.25	1.0622	6.3083
66	1	0	386.28	119.35	1.05	7.3664
67	1	0	690.32	216.11	1.05	8.3129
68	1	0	0.01	-0.05	1.0168	7.2968
69	1	0	-0.02	-0.04	1.0421	3.3547
70	1	0	-197.2	134.38	1.0363	-0.308
71	1	0	0.01	-0.08	1.0122	4.3773
72	1	0	-242	-24.4	1.0424	2.2711
79	1	0	-0.02	0.04	1.0585	7.4932
80	1	0	1594.2	116.25	1.043	10.762
81	1	0	255.93	-92.35	1.05	5.4425
82	1	1.595	698.28	-16739	1.025	6.3729
84	1	0	-58.81	7.73	1.0202	-1.0807
85	1	0	1156.4	109.39	1.0467	4.9146
87	1	0	99.67	-17.95	1.0183	0.0895
88	1	0	0.01	0.09	1.0487	5.1244
90	1	0	0.01	-0.16	1.03	3.1777
91	1	0	89.5	-95.86	1.0549	4.8787
93	1	0	-0.01	0.02	1.0268	1.8322
94	1	0	396.44	97.23	1.056	0.1453
96	1	0	-263.2	82.73	1.0203	-1.3908

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
97	1	0	-0.01	0.1	1.021	-0.7012
98	1	0	-18.8	-0.95	1.0419	-4.5926
99	1	0	-246	-20.3	1.0188	-1.3141
100	1	0	258.4	21.22	1.025	-7.3724
101	1	0	110.4	24.47	1.0212	-8.0213
102	1	0	-723	98.2	1.0295	-8.1478
103	1	0	-87.3	8.71	1.0238	-8.2774
106	1	0	283	80.48	1.0302	-9.3302
107	1	0	-134	-15.3	1.0292	-13.142
109	1	0	219.6	18.26	1.043	8.6725
124	1	0	-88.4	-35.8	1.0175	9.4209
125	1	0	-212	-52.9	1.0321	9.3622
127	1	0	136.1	-28	1.0327	9.2835
130	1	0	-157	-33.5	1.0207	5.5002
133	1	0	-53	-23.6	1.0022	-8.844
135	1	0	-128	-53.7	1.0337	2.8414
136	1	0	-86.9	-64.5	1.046	4.0854
138	1	0	-114	-19.2	1.0309	4.9192
144	1	0	-108	-20.4	1.0106	-1.9155
146	1	0	-22.8	2	1.0252	-1.1957
150	1	0	-46.8	-5.12	1.0163	-4.3918
152	1	0	-32.6	-12.6	1.0342	-0.1272
156	1	0	-16.6	4.72	1.0197	-5.2156
158	1	0	-46.5	15.95	1.0344	-3.606
160	1	0	-28.1	-12.2	1.021	3.9128
166	1	0	-93.2	19.37	1.0229	-4.07
168	1	0	-126	-6.74	1.0366	-0.826
169	1	0	194	57.98	1.0382	-0.4004
174	1	0	109	34.68	1.0392	-3.7781
175	1	0	-12.1	-25.8	1.0197	-6.9964
177	1	0	-280	-8.07	1.0198	-7.1754
188	1	0	-103	-27.5	1.0297	-4.8435
191	1	0	-535	-147	1.0247	-6.2103
198	1	0	-168	-30.8	1.0278	-6.0194
203	1	0	-85.9	-25.1	1.0251	-4.3102
208	1	0	157.1	43.49	1.033	-1.8066
209	1	0	-108	-14.5	1.0144	-3.2347
210	1	0	-70.7	-14.4	1.009	-4.7283

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
212	1	0	-113	19.26	1.0237	-5.0495
213	1	0	-8.59	78.98	1.0373	-7.3854
214	1	0	629.8	24.29	1.0244	10.224
216	1	0	-122	20.87	1.0166	0.7248
218	1	0	-92.6	31.46	1.0371	-3.8709
219	1	0	-37.5	37.95	1.0233	-3.2118
220	1	0	-50.9	-19	1.018	-1.8642
221	1	0	-26.4	-11.8	1.0092	-4.644
223	1	0	-65.1	-19.1	1.0152	-2.127
226	1	0	41.56	30.11	1.025	-2.2606
228	1	0	-237	-68.7	1.0147	-0.582
229	1	0	-42.4	-3.78	1.0225	-0.1108
230	1	0	-15.5	53.59	1.0318	1.3251
231	1	0	-23.2	-7.55	1.0238	2.5441
232	1	0	84.58	14.93	1.0346	5.3007
234	1	0	-131	-6.67	1.0471	1.7421
236	1	0	-245	-37.5	1.0295	-1.4974
237	1	0	-72.1	-22.6	1.0295	-1.0763
239	1	0	-45.9	-12.3	1.0157	-0.8492
240	1	0	-5.8	0.69	1.0296	-4.8489
241	2	0	14.93	-15.8	1.0118	-7.0861
242	1	0	-101	57.85	1.0204	-7.8173
243	1	0	-66.3	-10.2	1.0049	-10.519
248	1	0	-5.49	-1.77	1.0129	-4.2353
249	1	0	-16.9	-4.49	1.0082	-4.7974
257	1	0	-75.3	-32.5	1.0376	1.8969
259	1	0	-15.1	0.01	1.0291	1.9909
260	1	0	600.2	11.04	1.05	6.8898
261	1	0	-52	-20.5	1.045	6.0679
262	1	0	-92.4	-41.6	1.0434	5.7434
263	1	0	241.2	79.56	1.05	6.1669
264	1	0	-50.1	-17.7	1.0477	6.1283
265	1	0	-31.4	-10.4	1.0492	6.088
267	1	0	329.4	59.65	1.05	5.5741
268	1	0	-74.5	-24	1.0433	4.6557
269	1	0	-31.4	-10.4	1.0491	6.0847
270	1	0	-80.2	-27.3	1.0381	3.7301
271	1	0	-156	-18.5	1.0217	-1.0337

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (Per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
272	1	0	-30.3	-13.6	1.0312	-1.3893
273	1	0	-220	28.96	1.0362	-2.4808
274	1	0	-39.1	-10.8	1.0212	-0.4872
275	1	0	-111	-32.8	1.0381	1.9375
276	1	0	-105	-19.9	1.0321	-0.4431
279	1	0	30.23	-49.6	1.0272	-4.1246
280	1	0	-50.1	17.6	1.0353	-6.6545
281	1	0	-0.02	-0.02	1.0427	1.1395
282	1	0	-77.6	-16.2	1.01	-7.9834
283	1	0.021	-158	-188	1.0277	-8.5493
285	1	0	98.08	56.87	1.0345	-3.0023
286	1	0	-109	20.61	1.0194	-4.3981
290	1	0	-84.8	-11.5	1.0342	-0.984
291	1	0	-103	41.66	1.0455	-3.773
294	1	0	-234	-32.2	1.03	-3.9131
295	1	0	-199	-60.2	1.039	-1.4795
296	1	0	-20	-6.44	1.0264	2.4046
297	1	0	-49.7	-9.01	1.0304	2.177
299	1	0	18.4	-20.6	1.0216	-0.1794
300	1	0	0	0.01	1.0296	1.9449
301	1	0	-54.2	-20.4	1.0027	-4.8194
302	1	0	-43.4	-6.08	1.0396	2.6802
305	1	0	16.56	12.24	1.0388	2.6378
308	1	0	-25.1	-9.6	1.0084	-2.7402
310	1	0	92.29	-3.41	1.0444	2.3695
316	1	0	-80	-23.3	1.0284	-6.2927
318	1	0	-18.4	-1.43	1.0306	3.41
322	1	0	-123	-47.8	1.0262	1.5229
325	1	0	-235	-34.8	1.0174	3.7884
326	1	0	46.31	33.72	1.0333	8.8646
327	1	0	-16.3	-5.64	1.0488	5.7368
328	1	0	-22.8	-8.86	1.0188	5.6727
330	1	0	0	0	1.0084	1.1624
331	1	0	-220	-60	1.0138	-2.4828
332	1	0	75.02	55.62	1.0208	-2.7446
334	1	0	-102	-4.82	1.0094	-5.7212
336	1	0	-87.3	30.57	1.0215	-4.0442
339	1	0	-390	78.68	1.0404	1.3707

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
340	1	0	-20.2	-9.02	1.012	-6.4371
341	1	0	-68.9	-32.3	1.0069	-6.7291
344	1	0	1.39	40.28	1.0216	-5.6069
346	1	0	132	39.35	1.037	0.4089
347	1	0	-74.2	2.35	1.0423	-0.137
349	1	0	13	-27.1	1.0331	1.7253
351	1	0	-249	-84	1.036	1.2996
352	1	0	562.5	72.11	1.05	4.1619
354	1	0	-72	-20.2	1.0132	-2.6248
356	1	0	-10.2	-16.3	1.0216	-1.4925
357	1	0	-37.8	-13.8	1.0167	1.5778
358	1	0	-47.3	74.61	1.0303	-2.8808
359	1	0	-73.2	-30.2	1.0164	-2.7526
360	1	0	-307	-60.1	1.0183	-2.702
361	1	0	-182	41.54	1.0287	-2.0442
362	1	0	-49.9	-20.8	1.026	-0.5776
363	1	0	-47.1	-23.5	1.0213	-2.9601
365	1	0	-114	-99.2	1.0391	-0.549
374	1	0	-11.4	-4.03	1.0289	-7.2045
375	1	0	-49	-19.5	1.0322	-3.1973
378	1	0	-93.3	-46.2	1.0267	-1.9353
379	1	0	-68.3	-33.5	1.0312	-1.3585
380	1	0	-27.7	-9.41	1.0432	5.7904
381	1	0	-59.7	-30.7	1.0237	-1.3238
384	1	0	-20.7	-37.8	1.0225	-2.9859
385	1	0	-176	66.88	1.0133	-5.7821
386	1	0	-127	-45.3	1.0074	-6.2384
387	1	0	0.66	-8.1	1.0085	-4.2345
389	1	0	-74.9	-22.7	1.0404	0.9993
391	1	0	-67.8	-14.4	1.0122	-2.6037
392	1	0	74.65	-92	1.023	-0.0428
393	1	0	-95	-13.3	1.0351	-0.5343
397	1	0	-42.4	-19.8	1.0194	-2.0401
399	1	0	-21.7	5.87	1.0093	-4.7273
401	1	0	-65.8	28.92	1.0049	-7.4308
402	1	0	-25.2	-2.91	0.9997	-7.7196
408	1	0	122.4	4.8	1.0423	7.6728
410	1	0	-421	-118	1.0364	9.744

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
411	1	0	-372	-87.6	1.0141	5.81
412	1	0	-18.5	-2.8	1.0069	-5.6519
413	1	0	-41.6	-45.1	1.0024	2.2534
414	1	0	-140	-60.2	1.0055	2.7603
415	1	0	25.16	-35.8	0.9986	3.1102
420	1	0	-233	-70.4	0.9994	-0.5854
421	1	0	-0.09	-0.42	1.0432	4.6282
422	1	0	0.02	0.29	1.0419	4.2091
423	1	0	55.89	13.23	1.0259	1.3766
424	1	0	78.51	7.3	1.0156	1.1545
425	1	0	57.28	151.2	1.0173	-0.2525
427	1	0	-26.7	-8.07	1.0148	-0.8381
428	1	0	-256	9.65	1.0319	0.991
432	1	0	397.3	0.01	1.043	5.3802
433	1	0	345.1	-9.11	1.0499	6.7181
434	1	0	356.1	-16.6	1.043	4.8585
436	1	0	-195	-38.1	0.9986	3.1102
438	1	0	-120	-23.4	1.0387	3.7479
439	1	0	81.65	-22.3	1.0165	-2.7416
441	1	0	-31.3	-2.46	1.032	-0.7582
442	1	0	-59.7	-5.75	1.0425	0.3074
443	1	0	-24.8	-2.42	1.0395	1.649
444	1	0	55.67	-33.3	1.0278	1.2726
445	1	0	-27.1	-9.75	1.0324	-1.6869
446	1	0	-5.6	-1.42	1.0284	0.0922
448	1	0	137.4	51.63	1.0358	-4.1334
450	1	0	-59.9	-7.06	1.0263	-4.0259
452	1	0	-54.1	-0.74	0.9887	-11.807
453	1	0	-54.9	-17.9	1.0014	-8.2454
455	1	0	73.71	-12.5	1.0185	-5.7273
457	1	0	-163	-59.5	1.0363	-3.5859
458	1	0	-357	-132	1.0223	-6.2834
459	1	0	-156	-79.1	1.0276	-4.0597
460	1	0	212.6	-54.2	1.0096	-4.8606
462	1	0	-93.4	35.14	1.0142	-10.679
463	1	0	-65.5	-37.8	1.0221	-8.3778
465	1	0	-52.7	10.33	1.0207	-8.7373
466	1	0	39.91	-1.84	1.031	-6.4055

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
468	1	0	-208	-80.9	1.0006	-9.2591
469	1	0	-38.9	-0.57	1.0015	-9.6206
472	1	0	-119	-48.1	0.9861	-11.1
473	1	0	3.84	-2.34	1.0023	-7.6297
475	1	0	-57.1	-28.7	1.0305	-12.377
476	1	0	-91.4	4.11	0.986	-13.095
480	1	0	-44.8	-7.6	1.0189	-2.9207
481	1	0	-190	-33	1.0223	-14.555
485	1	0	-67.4	-9.1	1.0233	-14.594
486	1	0	-65.7	-18.3	1.0171	-6.188
487	1	0	-5.59	-1.78	1.0237	-15.684
488	1	0	-27.6	10.14	1.0245	-15.964
489	1	0	0	0	1.0243	-13.306
490	1	0	-83	-20.5	0.9734	-13.787
493	1	0	-156	-19	1.0362	14.562
495	2	0	-81	-48	1.0383	15.228
496	2	0	-59.5	37.68	1.0428	15.874
499	2	0	14.69	5.42	1.0373	14.872
500	2	0	-60	-35.2	1.0377	15.124
502	1	0	-13.1	-3.45	1.039	3.2437
510	1	0	-13.7	-1.63	1.0252	-1.9971
512	1	0	-11	-2.94	1.0293	-4.6698
522	1	0	-31.2	-12.1	1.0058	3.2949
524	1	0	-189	-43	1.026	-4.79
527	1	0	-108	-36.3	1.0326	-5.8454
532	1	0	-29.1	-8.45	1.0258	-6.1857
533	1	0	-54	59.8	1.0113	-1.2967
539	1	0	0	-0.11	1.056	4.1431
541	1	0	389.9	37.28	1.056	2.3323
542	1	0	459.9	153.4	1.0531	4.3973
547	1	0	382.2	80.34	1.05	2.9272
548	1	0	-87.4	5.33	1.0382	12.414
549	1	0	0	-0.02	1.0255	11.42
552	1	0	-27.1	-8.6	1.018	-3.2952
555	1	0	-17.1	-5.11	0.9959	-7.4772
562	1	0	-4.45	-9.44	1.0233	-1.0836
568	1	0	1280	7.76	1.05	7.9985
576	1	0	80.96	19.33	1.0327	9.4515

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
581	1	0	-0.02	0.19	1.0064	4.3308
582	1	0	510.3	160.8	1.056	10.028
640	1	0	-0.02	0.05	1.0601	17.358
641	1	0	414.6	-52.8	1.0598	17.657
642	1	0	120.1	-6.27	1.0601	17.444
643	1	0	307.8	-18.9	1.0603	17.69
1000	1	0	53.1	-2.06	1.025	1.8374
1001	1	0	113.7	-21.3	1.025	2.3779
1002	1	0	110	-35	1.025	2.724
1003	1	0	-82.4	-15.9	1.026	2.781
1006	1	0	198.9	21.1	1.031	3.354
1011	1	0	-32.2	-12.2	1.0152	-2.9543
1018	1	0	-18.6	-9.98	1.0338	0.508
1020	1	0	-8.07	-0.31	1.0388	1.9938
1021	1	0	-39.3	-12.3	1.0329	0.3048
1024	1	0	-62.7	-61.5	1.0293	-7.7165
1027	1	0	-0.01	-0.03	1.0538	6.9058
1029	1	0	-20.8	-8.02	1.0293	-1.1807
1033	1	0	-28.9	-2.44	1.0129	-3.023
1034	1	0	0.01	0.05	1.0173	-3.0652
1035	1	0	-23.5	-7.4	1.0256	-1.0405
1040	1	0	-65	-28.4	1.0176	10.189
1045	1	0	-58	-18	1.0203	-0.5265
1048	1	0	-64	-17.3	1.0458	7.5921
1053	1	0	-10.4	-3.92	1.0228	-15.063
1059	1	0	0	-0.01	1.0208	5.8116
1061	1	0	-0.01	-0.06	1.0024	-9.5686
1062	1	0	-23.4	-1.93	1.0055	-6.7345
1065	1	0	0	-0.01	1.0261	-8.4276
1067	1	0	-65	-5.65	1.0315	0.0183
1071	1	0	-18.7	-1.29	1.0193	7.3984
1072	1	0	-34.8	-4	1.0086	-2.6402
1075	1	0	-12.9	-3.85	1.0213	-2.7115
1076	1	0	-99.8	-38.2	0.998	0.719
1079	1	0	-44.6	-4.48	1.0134	0.6783
1081	1	0	-12.5	-4.32	1.0087	-4.7066
1086	1	0	-130	4.22	1.0016	0.4249
1094	1	0	-45.7	-14.5	1.0274	-4.3837

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
1095	1	0	0	-0.03	1.019	-4.971
1098	2	0	178.7	14.79	1.045	3.2157
1100	1	0	-17.5	-5.4	1.0377	-2.3527
1103	1	0	-51.8	38.32	1.0196	-3.099
1106	1	0	-18.5	-5.14	1.0338	1.0927
1108	1	0	-78.4	-43.9	1.0157	-3.4692
1109	1	0	-14.6	-2	1.0355	-2.5627
1111	1	0	-92.2	-0.12	1.0282	0.0141
1113	1	0	-75.1	-35.1	1.0012	-5.5281
1114	1	0	-11.1	-5.61	1.019	-3.001
1116	1	0	-122	-22.2	1.0174	8.493
1120	1	0	-19	-5.71	1.0136	-8.9455
1121	1	0	-49.2	-12.9	1.0382	-0.2927
1123	1	0	-22	-7.96	1.0176	-0.2486
1125	1	0	-125	-30.2	1.0361	-0.321
1127	1	0	0.4	2.2	1.0385	2.2916
1130	1	0	-4.6	-1.78	1.0246	0.4728
1132	1	0	-18.4	-8.87	1.0327	-3.1468
1135	1	0	-16.6	-5.01	1.0322	1.1464
1137	1	0	0	0.06	1.0383	-0.4011
1145	1	0	-55.1	-30.6	1.0344	-1.0285
1149	1	0	0	-0.03	1.0357	1.6046
1152	1	0	-51.9	-12.3	1.0276	-1.4987
1159	1	0	-9.21	-3.28	1.0282	-5.3113
1160	1	0	-14.9	-5.65	1.0327	-0.4891
1163	1	0	-3.5	-1.36	1.0339	9.6487
1166	1	0	-33	-13.1	1.0343	-4.1487
1168	1	0	-19.6	-10.8	1.0167	-2.7546
1172	1	0	-17.2	-2.35	1.026	4.2385
1176	1	0	-18.3	-3.99	1.0228	0.3913
1177	1	0	-14.9	-5.14	0.9983	0.1318
1180	1	0	-104	-26	1.0327	9.4515
1185	1	0	-46.5	-27.5	1.0369	2.1983
1186	1	0	-13	-4.53	1.0199	-1.1396
1191	1	0	-4.89	-0.26	1.0297	0.5127
1193	1	0	-7.78	-3.23	1.038	3.5309
1200	1	0	-7.11	-0.37	1.0423	5.954
1202	1	0	0	0.06	1.0409	2.4474

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
1203	1	0	-15.8	-4.15	1.0399	2.3119
1204	1	0	-16.9	-7	0.9999	-7.7112
1205	1	0	-13.2	-4.3	1.0124	-6.5343
1211	1	0	-15.6	-2.66	1.0174	-6.3743
1212	1	0	-29.3	-10.5	1.0291	1.9588
1214	1	0	-279	-49	1.0282	2.1779
1219	1	0	-42.9	-16.3	1.0069	-2.8341
1221	1	0	-14.6	-5.82	1.0157	-5.3627
1225	1	0	-28.4	-10.8	1.0142	-10.386
1227	1	0	-18.6	-10.8	1.0353	-6.6044
1232	1	0	-5.2	-0.9	1.0361	3.5766
1236	1	0	-0.02	-0.09	1.0159	-6.05
1241	1	0	-141	-9.94	1.0399	1.3608
1242	1	0	-25.7	-4.48	1.0392	1.7241
1245	1	0	-18.9	-9.42	1.023	-1.4997
1247	1	0	-7.7	-4.67	1.0398	-2.2158
1255	1	0	-24.3	-9.09	1.0149	-1.0757
1260	1	0	-44.8	-22.8	1.0179	-2.869
1262	1	0	-19.4	-4.77	1.0152	-0.3084
1265	1	0	-19.6	-4.73	1.0281	0.5248
1267	1	0	-23.6	-9.34	1.0125	-2.8983
1268	1	0	-18.8	-9.25	1.0248	1.7084
1270	1	0	0	0.03	1.0253	-1.5368
1271	1	0	-28.4	-6.75	1.0253	-1.6566
1280	1	0	-6	-1.39	1.0264	-5.8534
1282	1	0	-7.09	-1.98	1.015	-3.2007
1288	1	0	-37.6	-12.9	1.002	-7.8135
1289	1	0	-15.5	-5.57	1.0093	3.6658
1290	1	0	-9.89	-3.4	1.0386	-1.4957
1291	1	0	-7.01	-3.02	1.0345	0.8473
1292	1	0	-11.5	-3.83	1.0319	0.1772
1294	1	0	49.86	-6.21	0.9994	0.934
1295	1	0	-0.01	-0.09	1.0086	3.5867
1297	1	0	-27.6	-9.37	1.0327	-0.2877
1298	1	0	-58.9	-3.1	1.0317	2.6766
1301	1	0	-83.7	-33	1.0142	-3.3849
1303	1	0	-5.8	-0.44	1.0166	-3.1294
1304	1	0	-69.9	-31.9	1.0469	-1.0439

Table A-5. Continued

Bus Number	Type (1-PQ, 2-PV, 3-slack)	Bshunt (per unit)	P Injection (MW)	Q Injection (MVAR)	Magnitude (per unit)	Angle (degrees)
1308	1	0	-16.5	-4.95	1.0188	-9.1461
1309	1	0	-8.3	-4.18	1.0095	-8.495
1311	1	0	-26.3	-7.67	1.0253	3.0602
1320	1	0	0	-0.03	1.0404	1.8936
1324	1	0	-16.6	-5.73	1.0393	-2.1312
1327	1	0	-53	-8.71	1.0172	4.4639
1332	1	0	-9.5	-3.35	1.023	-4.0975
1334	1	0	-27.6	-8.11	1.0179	-2.8925
1338	1	0	-13.6	-4.36	1.0353	-2.9369
1342	1	0	31.22	-8.1	0.9964	-7.4103
1352	1	0	-0.41	-2.28	1.0386	2.291
1357	1	0	22.15	-1.39	1.0314	-0.4349
1362	1	0	-62.4	-26.2	1.0084	1.1624
1500	1	0	-219	9.43	1.0065	-0.0793
1502	1	0	0.01	0.03	0.9987	-1.0127
1503	1	0	-17.3	-4.8	1.0066	0.6381
1504	1	0	-0.01	-0.02	0.9826	0.7439
1505	1	0	-0.01	-0.02	0.9826	1.9809
1509	1	0	0.01	-0.02	0.9806	1.6092
1513	1	0	-77.1	11.25	1.0155	-3.209
1514	1	0	-100	-22.7	1.0072	-2.0745
5640	1	0	-71.7	-28.4	1.0235	-2.0359
5680	1	0	-73.1	-26.5	1.0164	-2.9049
9014	1	0	-36.3	-7.5	1.0171	8.0497
9032	1	0	115.6	43.05	0.9857	-12.264
9151	1	0	0.01	-0.01	1.0397	1.8199
9999	1	0	28.07	-6.15	1.0172	1.8816

Table A- 6. Utility System B Load Flow Line Data

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
1	2	0.0001	0.0057	0	1
1	4	0.0002	0.0151	0	1
1	15	0.0005	0.0064	0.5306	0
3	4	0	0.0001	0.0001	0
3	493	0.0002	0.003	0.0016	0
3	495	0.0007	0.0096	0.005	0
3	499	0.0006	0.0075	0.0039	0
3	500	0.0006	0.0075	0.0039	0
3	500	0.0006	0.0074	0.0039	0
3	548	0.0001	0.05	0.0001	0
3	548	0	0.0001	0.0001	0
3	549	0.0001	0.05	0.0001	0
3	549	0	0.0001	0.0001	0
4	9	0	0.0004	0.0002	0
5	496	0.0008	0.0101	0.0053	0
6	-496	0.0007	0.01	0.0052	0
7	495	0.0007	0.0093	0.0048	0
8	500	0.0006	0.0075	0.0039	0
9	-499	0.0006	0.0075	0.0039	0
13	533	0.0023	0.0195	0.0456	0
13	-1500	0.0031	0.0267	0.0624	0
15	16	0.0002	0.0151	0	1
15	40	0.0012	0.0177	1.4683	0
16	-124	0.0029	0.0394	0.0207	0
16	125	0.0007	0.0094	0.0051	0
16	-130	0.0071	0.0473	0.0726	0
16	496	0.0176	0.0928	0.0421	0
16	1163	0.0005	0.0064	0.0038	0
17	18	0.0003	0.0149	0	1
17	568	0.0009	0.0131	1.1029	0
17	582	0.0002	0.0033	0.2668	0
18	135	0.0067	0.0467	0.0272	0
18	135	0.0064	0.0443	0.0254	0
18	-136	0.0008	0.0106	0.0062	0
18	-136	0.0007	0.0089	0.005	0
18	-144	0.0134	0.1084	0.0669	0
18	-1106	0.0066	0.0405	0.0196	0
19	20	0.0003	0.0119	0	1
19	37	0.0008	0.012	0.9912	0
19	-542	0.0004	0.0061	0.5021	0
20	-152	0.0042	0.0423	0.0232	0
21	22	0.0002	0.008	0	1
21	25	0.0003	0.0048	0.3957	0
21	568	0.0005	0.0071	0.5938	0
22	-191	0.0044	0.0427	0.0241	0
22	191	0.0043	0.0422	0.0239	0
22	203	0.0029	0.0283	0.016	0
22	1338	0.0025	0.0245	0.0152	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
23	24	0.0002	0.0088	0	1
23	25	0.0004	0.0061	0.5044	0
24	174	0.0053	0.031	0.0156	0
24	198	0.0066	0.0385	0.0194	0
24	1227	0.0058	0.0568	0.0325	0
25	26	0.0004	0.0144	0	0.976
25	27	0.0002	0.0151	0	0.976
25	-541	0.0011	0.0097	1.5607	0
25	542	0.0006	0.0091	0.7531	0
26	-191	0.0024	0.0207	0.0155	0
26	-316	0.0018	0.0176	0.0135	0
27	-191	0.0024	0.0207	0.0155	0
27	-198	0.003	0.0296	0.0167	0
27	-527	0.0029	0.0271	0.0152	0
28	-336	0.007	0.0432	0.025	0
28	1034	0.0018	0.0109	0.0063	0
30	35	0.0003	0.0046	0.4208	0
30	234	0.0002	0.008	0	1
30	539	0	0.0001	0.0019	0
33	34	0.0002	0.0152	0	1
33	52	0.0017	0.0239	2.222	0
33	-541	0.0002	0.002	0.2782	0
34	219	0.0063	0.0449	0.0224	0
34	1247	0.0017	0.0139	0.0073	0
35	36	0.0002	0.0151	0	1
35	52	0.0019	0.0269	2.2434	0
36	234	0.0062	0.0634	0.0338	0
36	236	0.0065	0.0589	0.0305	0
36	1121	0.0046	0.0289	0.0167	0
36	1290	0.0052	0.0399	0.0183	0
37	40	0.0005	0.0076	0.6274	0
37	48	0.0009	0.013	1.0766	0
37	-582	0.0007	0.0097	0.8004	0
37	640	0.0002	0.0147	0	0.9651
38	39	0	0.0001	0	0
38	305	0.0289	0.1763	0.0821	0
38	640	0	0.0001	0	0
38	643	0.0001	0.0016	0.0012	0
38	1027	0.001	0.0105	0.006	0
39	302	0.0185	0.1127	0.0523	0
39	1200	0.0044	0.0271	0.0504	0
40	48	0.0007	0.0109	0.9077	0
41	-318	0.0023	0.0154	0.0076	0
41	1268	0.0108	0.0694	0.0322	0
42	-44	0.0008	0.012	0.9847	0
43	-325	0.0086	0.0882	0.0724	0
43	-325	0.0085	0.0881	0.0722	0
43	331	0.0078	0.08	0.066	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
43	9999	0.0048	0.0501	0.0418	0
44	-48	0.0006	0.009	0.767	0
44	71	0.0009	0.0124	1.1242	0
44	9151	0.0001	0.019	0	0.9563
45	297	0.0063	0.0404	0.0211	0
45	322	0.0025	0.0249	0.0133	0
9151	45	0.0001	-0.0039	0	1
48	49	0	0.0178	0	0.9625
48	50	0.0004	0.0064	0.5288	0
48	339	0.0002	0.0148	0	0.975
48	581	0.0004	0.0063	0.5188	0
49	360	0.0031	0.0377	0.025	0
49	361	0.003	0.0286	0.0269	0
50	51	0	0.0151	0	0.9625
50	-52	0.0014	0.0198	1.6428	0
50	69	0.001	0.0142	1.2137	0
51	310	0.0028	0.0187	0.009	0
51	389	0.0032	0.0209	0.0109	0
51	389	0.0032	0.0206	0.0108	0
51	1021	0.0056	0.0204	0.0089	0
52	54	0.0002	0.0024	0.2014	0
52	60	0.0006	0.0086	0.7039	0
52	88	0.0003	0.0038	0.3114	0
53	271	0.0063	0.0389	0.0225	0
53	1149	0.003	0.0207	0.0106	0
54	55	0.0002	0.0151	0	1
55	275	0.0007	0.0061	0.0031	0
55	-275	0.0009	0.0076	0.0042	0
55	-276	0.0021	0.029	0.0156	0
55	-1098	0.0016	0.0103	0.0058	0
55	1149	0.0017	0.0108	0.0057	0
55	1202	0.0011	0.0084	0.0068	0
55	1291	0.004	0.026	0.0136	0
55	1352	0.0007	0.0065	0.0037	0
57	-1067	0.0079	0.0521	0.0265	0
57	1297	0.0006	0.0053	0.0026	0
57	1357	0.0013	0.0088	0.0047	0
58	59	0.0005	0.02	0	1.0006
58	-63	0.001	0.0152	1.3095	0
59	1297	0.0022	0.019	0.0102	0
60	61	0.0002	0.0152	0	1
60	-65	0.0007	0.0105	0.8625	0
60	88	0.0004	0.0054	0.4488	0
61	-291	0.0046	0.0432	0.0233	0
61	294	0.004	0.0398	0.0209	0
61	295	0.0036	0.0225	0.0129	0
61	1304	0.0001	0.0014	0.0008	0
61	-1324	0.0065	0.0411	0.0199	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
62	63	0	0.0001	0	0
63	65	0.0003	0.0048	0.4055	0
64	66	0.0059	0.0537	0.0265	0
64	290	0.0024	0.0218	0.0109	0
65	67	0.0004	0.0152	0	1.025
65	79	0.0004	0.0061	0.5088	0
65	81	0.0008	0.0113	0.9274	0
66	-408	0.0066	0.0394	0.0194	0
66	410	0.0187	0.0937	0.0416	0
66	1045	0.0092	0.0578	0.0268	0
66	1048	0.0022	0.0141	0.007	0
68	67	0	0.0117	0	0.9692
68	67	0.0002	0.0143	0	0.9692
67	-80	0.0058	0.04	0.075	0
67	414	0.0137	0.0877	0.0456	0
67	414	0.0134	0.0876	0.0454	0
68	1289	0.0023	0.0215	0.0417	0
69	70	0.0002	0.015	0	1
69	-81	0.001	0.0143	1.1864	0
70	387	0.0104	0.0746	0.0344	0
70	392	0.0022	0.0153	0.0248	0
70	393	0.0014	0.0084	0.0158	0
70	-393	0.0013	0.0099	0.0144	0
70	441	0.0021	0.0103	0.0046	0
71	72	0.0002	0.0151	0	0.9628
71	90	0.0019	0.0279	2.3482	0
71	581	0.0002	0.0033	0.2688	0
72	352	0.0025	0.0173	0.0265	0
72	1020	0.002	0.0156	0.0087	0
72	1241	0.0009	0.0123	0.0071	0
72	-1320	0.0009	0.0123	0.0071	0
79	80	0.0002	0.0167	0	1
80	410	0.0007	0.0051	0.0093	0
80	410	0.0007	0.009	0.0052	0
80	410	0.0004	0.0048	0.0102	0
80	1048	0.0095	0.0655	0.0322	0
81	82	0.0002	0.0142	0	1.012
81	-85	0.0005	0.0078	0.6553	0
81	85	0.0006	0.0087	0.7999	0
82	411	0.0036	0.0487	0.0258	0
82	-425	0.0058	0.0572	0.0326	0
82	1076	0.0042	0.0559	0.0306	0
82	1295	0.0025	0.0319	0.0183	0
84	444	0.0268	0.1302	0.0598	0
84	1123	0.0112	0.0533	0.0255	0
85	-90	0.0006	0.0085	0.7818	0
85	93	0.0009	0.0123	1.0213	0
85	97	0.0014	0.0201	1.8544	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
87	-427	0.0046	0.0224	0.0102	0
87	1123	0.0025	0.0125	0.0055	0
87	1130	0.007	0.0339	0.0157	0
88	281	0.0002	0.0149	0	0.9938
90	91	0.0002	0.015	0	0.95
90	93	0.0002	0.0036	0.2963	0
91	-421	0.0025	0.0187	0.0262	0
91	433	0.0028	0.0203	0.0292	0
93	94	0.0003	0.0155	0	0.95
93	97	0.0003	0.0049	0.4	0
94	442	0.002	0.0154	0.0136	0
94	457	0.0023	0.0302	0.0173	0
94	457	0.0029	0.0369	0.0211	0
94	459	0.0028	0.0434	0.024	0
96	99	0.0014	0.0066	0.0032	0
96	99	0.0014	0.0071	0.003	0
96	99	0.0014	0.0071	0.003	0
96	-444	0.0045	0.0452	0.0218	0
96	1006	0.0179	0.0889	0.0366	0
96	1186	0.006	0.03	0.0132	0
97	98	0.0002	0.0148	0	0.9375
97	102	0.001	0.0143	1.2752	0
98	-101	0.0337	0.1633	0.0763	0
98	-457	0.0077	0.0376	0.0184	0
98	-457	0.0029	0.027	0.0282	0
98	458	0.0023	0.0162	0.0201	0
98	458	0.0022	0.0224	0.02	0
98	466	0.0125	0.0675	0.0312	0
98	468	0.0126	0.0731	0.1122	0
98	1159	0.0056	0.0331	0.016	0
99	1000	0.014	0.0549	0.025	0
99	1001	0.0193	0.0635	0.0302	0
99	1002	0.0199	0.076	0.0344	0
100	103	0.0025	0.0314	0.0192	0
100	103	0.0025	0.0315	0.0193	0
100	-466	0.0235	0.114	0.0532	0
100	1065	0.0051	0.0251	0.0135	0
100	1309	0.0047	0.0236	0.0425	0
101	103	0.0025	0.0312	0.0192	0
101	103	0.0025	0.0313	0.0193	0
101	1061	0.0052	0.025	0.0124	0
102	103	0.0002	0.0129	0	1
102	106	0.0008	0.011	0.9204	0
103	463	0.0007	0.0032	0.0017	0
106	107	0.0002	0.0149	0	0.9878
107	-475	0.0054	0.0359	0.0187	0
107	-481	0.0024	0.0237	0.0136	0
107	-481	0.0047	0.0326	0.0174	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
107	485	0.0039	0.0248	0.011	0
107	489	0.0049	0.0048	0.0261	0
109	232	0.0023	0.0291	0.0173	0
124	1040	0.0046	0.0232	0.0102	0
125	-576	0.0002	0.0021	0.0012	0
127	326	0.001	0.0104	0.0056	0
127	326	0.0018	0.0121	0.0115	0
130	-138	0.0079	0.0471	0.0917	0
133	-460	0.0093	0.048	0.0824	0
133	-462	0.009	0.0574	0.031	0
133	-469	0.0063	0.0353	0.0534	0
138	1200	0.0036	0.0218	0.0419	0
146	480	0.0048	0.0291	0.0141	0
146	1106	0.0055	0.0322	0.0189	0
150	480	0.0068	0.0414	0.0201	0
150	-1095	0.0098	0.0586	0.0284	0
152	1035	0.0116	0.0425	0.0192	0
156	-1095	0.0041	0.0254	0.0146	0
158	1094	0.0056	0.0332	0.016	0
158	-1338	0.0014	0.0135	0.0084	0
160	-296	0.0164	0.0789	0.0374	0
160	326	0.0303	0.1477	0.0696	0
166	-510	0.0101	0.0385	0.017	0
168	-169	0.001	0.0064	0.003	0
169	-260	0.0253	0.1347	0.0584	0
169	1137	0.003	0.0136	0.0073	0
169	1152	0.0067	0.0329	0.0153	0
169	1168	0.014	0.0456	0.0223	0
174	188	0.004	0.0203	0.0118	0
175	177	0.0029	0.0139	0.007	0
175	316	0.004	0.0397	0.0265	0
177	-191	0.0043	0.0223	0.0111	0
177	191	0.0043	0.0223	0.0111	0
177	203	0.0055	0.0537	0.031	0
191	198	0.0045	0.0259	0.0118	0
191	532	0.0028	0.016	0.008	0
198	-532	0.0017	0.01	0.005	0
208	248	0.0123	0.0562	0.0297	0
208	1075	0.0079	0.0253	0.011	0
209	-1011	0.0031	0.0101	0.0051	0
209	1075	0.0056	0.0176	0.0076	0
210	248	0.0027	0.012	0.0065	0
210	249	0.0048	0.0219	0.0116	0
212	512	0.0079	0.0531	0.026	0
212	1255	0.0102	0.0697	0.0339	0
213	-1024	0.0018	0.0116	0.0065	0
213	-1227	0.002	0.02	0.0116	0
214	1071	0.0014	0.0179	0.0114	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
214	1172	0.0093	0.0588	0.0333	0
214	9014	-0.0003	0.0249	0	1
216	1071	0.0036	0.0448	0.0286	0
216	1262	0.0019	0.0128	0.0064	0
218	1166	0.0014	0.0094	0.0046	0
218	1247	0.003	0.0204	0.0099	0
219	-221	0.011	0.0553	0.0288	0
219	1108	0.0038	0.0229	0.011	0
219	-1334	0.0038	0.026	0.0128	0
220	-226	0.0101	0.0454	0.0248	0
220	229	0.0081	0.0505	0.0292	0
221	1081	0.0011	0.005	0.0027	0
223	-259	0.01	0.0672	0.0337	0
223	1334	0.0041	0.0282	0.0137	0
226	1108	0.007	0.043	0.0208	0
228	-425	0.0008	0.0052	0.0028	0
228	-425	0.0006	0.0052	0.0027	0
229	230	0.0103	0.0639	0.0369	0
229	-231	0.0116	0.0754	0.0399	0
230	446	0.0069	0.0451	0.0236	0
230	1172	0.0051	0.0318	0.0184	0
230	1191	0.0044	0.0288	0.0151	0
231	1311	0.0018	0.0109	0.0062	0
232	239	0.0089	0.0601	0.0296	0
232	1311	0.0059	0.037	0.0211	0
234	236	0.0046	0.0467	0.0251	0
234	1125	0.0033	0.0327	0.019	0
234	1160	0.004	0.0277	0.0128	0
236	1029	0.0017	0.0119	0.0055	0
237	1029	0.0004	0.0028	0.0013	0
237	-1160	0.0011	0.0079	0.0037	0
239	1282	0.0044	0.0296	0.0146	0
240	1109	0.0081	0.0618	0.0284	0
240	1280	0.0044	0.0296	0.0145	0
241	242	0.0124	0.077	0.0443	0
242	1120	0.0036	0.0244	0.012	0
242	1211	0.0033	0.0217	0.0112	0
242	1280	0.0095	0.0636	0.0313	0
243	1120	0.0069	0.0431	0.0248	0
249	1081	0.0034	0.0153	0.0081	0
257	-275	0.0002	0.0011	0.0006	0
259	-260	0.0111	0.0725	0.0377	0
260	261	0.0063	0.0362	0.0178	0
260	264	0.0117	0.0677	0.0348	0
260	264	0.0105	0.0681	0.0355	0
260	327	0.0147	0.0838	0.0418	0
260	380	0.0121	0.0585	0.0282	0
260	510	0.0384	0.1473	0.065	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
260	1298	0.0128	0.0768	0.0367	0
261	263	0.0077	0.0385	0.017	0
262	-263	0.0024	0.011	0.0055	0
262	380	0.0019	0.0091	0.0044	0
263	264	0.0031	0.0165	0.0077	0
263	265	0.0011	0.0052	0.0026	0
263	-267	0.0096	0.0497	0.0204	0
263	269	0.0011	0.0054	0.0026	0
263	1202	0.0244	0.1207	0.0574	0
267	268	0.0036	0.0177	0.0078	0
267	270	0.0052	0.0257	0.0115	0
267	274	0.0206	0.0991	0.0458	0
267	327	0.0056	0.0292	0.012	0
268	-502	0.0278	0.0908	0.0377	0
270	1193	0.0011	0.0061	0.003	0
271	-274	0.0064	0.0312	0.0145	0
271	362	0.0015	0.0142	0.0085	0
272	273	0.0071	0.048	0.0267	0
272	-276	0.0029	0.0248	0.0147	0
273	-281	0.0017	0.0235	0.0122	0
273	1132	0.002	0.0115	0.006	0
274	-391	0.0174	0.086	0.0404	0
279	280	0.0097	0.184	0	1.004
279	280	0.0097	0.1827	0	1.004
279	291	0.0104	0.0503	0.0234	0
281	295	0.0045	0.0447	0.0245	0
281	362	0.003	0.03	0.0165	0
282	283	0.0143	0.0776	0.0336	0
282	1205	0.0061	0.0292	0.014	0
283	374	0.0018	0.0165	0.0137	0
285	290	0.0056	0.0274	0.0125	0
285	374	0.0044	0.0482	0.0904	0
285	375	0.002	0.0142	0.0071	0
285	1332	0.006	0.0534	0.0279	0
286	1045	0.0055	0.0353	0.0163	0
286	1221	0.003	0.0152	0.0067	0
286	1332	0.0022	0.0192	0.0102	0
294	1132	0.0026	0.0156	0.0084	0
295	1304	0.0036	0.0234	0.0123	0
296	-297	0.0056	0.0266	0.013	0
299	305	0.0108	0.0654	0.0304	0
299	1260	0.0078	0.0483	0.028	0
300	1079	0.0033	0.0155	0.0071	0
300	1212	0.0232	0.0497	0.0209	0
301	-387	0.0032	0.0202	0.0113	0
302	310	0.0212	0.0696	0.0314	0
302	1212	0.0241	0.0518	0.0217	0
305	502	0.0196	0.0708	0.0312	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
308	1219	0.0035	0.0173	0.0078	0
308	-1245	0.0136	0.0673	0.0299	0
318	-1027	0.0144	0.0876	0.0408	0
322	1268	0.0038	0.023	0.0107	0
325	-1327	0.0038	0.0272	0.0234	0
326	1059	0.0066	0.0472	0.0407	0
328	1059	0.0041	0.0126	0.0056	0
330	-1362	0	0.0001	0	0
331	332	0.0028	0.0246	0.013	0
331	1034	0.0014	0.0097	0.0051	0
331	-9999	0.0168	0.0833	0.0368	0
332	1114	0.0006	0.0053	0.0029	0
334	354	0.0135	0.0882	0.046	0
334	448	0.0192	0.0903	0.0442	0
334	-1113	0.008	0.0552	0.0256	0
336	439	0.0048	0.0299	0.017	0
339	381	0.0016	0.0199	0.0118	0
340	-1236	0.0055	0.0365	0.0245	0
341	-1236	0.0041	0.0195	0.0092	0
344	-547	0.0302	0.1788	0.0885	0
344	1236	0.0021	0.0099	0.0047	0
346	-397	0.0086	0.0342	0.0146	0
346	-1018	0.0044	0.0186	0.0083	0
347	-547	0.0134	0.078	0.0398	0
349	-1020	0.0057	0.022	0.0099	0
349	-1135	0.0066	0.0253	0.0116	0
351	547	0.0015	0.0148	0.0085	0
351	1320	0.0038	0.0347	0.0198	0
352	547	0	0.0001	0.0001	0
352	552	0.0137	0.0798	0.0404	0
352	1265	0.0063	0.0408	0.021	0
354	-1265	0.0059	0.0385	0.02	0
354	1267	0.0048	0.0297	0.0172	0
356	357	0.0072	0.0754	0.0608	0
356	358	0.0033	0.0396	0.0324	0
357	-9999	0.0005	0.005	0.0041	0
358	363	0.0025	0.0182	0.0295	0
359	360	0.0105	0.0239	0.0372	0
359	5640	0.0038	0.0207	0.0094	0
360	-363	0.0017	0.013	0.0189	0
360	381	0.001	0.0129	0.0078	0
360	-5680	0.0008	0.0053	0.0095	0
361	-378	0.0019	0.0113	0.0061	0
361	1301	0.0053	0.0312	0.0191	0
365	-379	0.0015	0.0091	0.0047	0
365	384	0.0031	0.0301	0.0174	0
581	365	0.0002	0.0151	0	0.9262
365	1145	0.0004	0.0048	0.0029	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
375	1100	0.0106	0.0752	0.0379	0
378	-379	0.0017	0.0103	0.0057	0
384	-385	0.0076	0.038	0.0167	0
385	386	0.0015	0.0088	0.0045	0
385	-552	0.0052	0.0303	0.0154	0
386	412	0.0065	0.0379	0.0194	0
387	412	0.0094	0.0548	0.0282	0
391	392	0.017	0.0839	0.0394	0
391	1033	0.0035	0.0225	0.0117	0
397	399	0.0137	0.0512	0.0228	0
399	427	0.0292	0.1422	0.0654	0
399	1062	0.0045	0.03	0.0147	0
401	1062	0.0019	0.0129	0.0063	0
401	1204	0.0032	0.0199	0.0113	0
402	1204	0.0003	0.0017	0.001	0
402	-1342	0.0054	0.0335	0.0192	0
408	414	0.0149	0.0847	0.0396	0
410	411	0.0023	0.0312	0.0162	0
410	411	0.0017	0.0235	0.0126	0
411	436	0.0026	0.0249	0.0142	0
413	-1295	0.0014	0.0185	0.0106	0
1509	413	0.0003	0.0129	0	0.9756
414	-415	0.0022	0.0213	0.0125	0
414	1086	0.0036	0.0314	0.0165	0
1505	414	0.0003	0.0131	0	0.9756
415	436	0	0.0001	0	0
420	424	0.0157	0.0789	0.0342	0
420	424	0.0141	0.0703	0.0306	0
420	425	0.0026	0.0271	0.0139	0
420	1076	0.002	0.0273	0.0146	0
420	1177	0.0068	0.0339	0.0151	0
421	422	0.0003	0.0028	0.0015	0
421	-434	0.0002	0.0017	0.0035	0
422	428	0.0028	0.0213	0.0166	0
423	424	0.0117	0.0593	0.025	0
423	1176	0.0078	0.0392	0.0172	0
428	1130	0.0075	0.0378	0.0179	0
432	438	0.0039	0.0232	0.011	0
432	438	0.0038	0.0218	0.0112	0
432	1232	0.005	0.0278	0.0136	0
433	1214	0.0037	0.0493	0.0271	0
434	-1214	0.0037	0.0493	0.0271	0
438	-443	0.0053	0.0248	0.0111	0
439	-1267	0.007	0.0467	0.0234	0
441	1245	0.0055	0.027	0.012	0
442	-443	0.0037	0.0189	0.0085	0
442	-444	0.0097	0.0472	0.0219	0
442	-445	0.0087	0.0433	0.0192	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
444	1232	0.0074	0.0366	0.0161	0
445	450	0.0144	0.0711	0.0319	0
446	-1111	0.0005	0.0032	0.0017	0
448	450	0.0316	0.1573	0.07	0
448	-486	0.0082	0.04	0.0186	0
452	-453	0.0233	0.1129	0.0522	0
453	455	0.0087	0.0421	0.0194	0
455	1159	0.0051	0.0249	0.0115	0
457	-459	0.003	0.0259	0.0143	0
457	524	0.002	0.0191	0.0103	0
458	460	0.0036	0.0357	0.0195	0
458	465	0.011	0.0676	0.0308	0
459	524	0.0021	0.0199	0.0108	0
462	1225	0.002	0.0124	0.0068	0
465	466	0.009	0.0578	0.0239	0
465	1308	0.0014	0.0087	0.0049	0
468	1309	0.0019	0.0165	0.0158	0
469	1061	0.0192	0.0853	0.0394	0
472	-1061	0.0037	0.0237	0.0124	0
472	-1065	0.0037	0.0237	0.0124	0
473	1288	0.017	0.0822	0.0387	0
475	1065	0.0194	0.0951	0.0431	0
475	9032	0.0006	0.009	0	1.05
476	9032	0.0003	0.0154	0	1
481	-485	0.0033	0.021	0.011	0
485	1053	0.004	0.019	0.009	0
486	1288	0.0187	0.0908	0.0427	0
487	488	0.0038	0.0171	0.0094	0
487	1053	0.0067	0.0322	0.015	0
489	490	0	0.0096	0	1.05
493	-496	0.0005	0.0575	0.0039	0
495	496	0.0001	0.0255	0	0
495	-500	0.0002	0.0409	0	0
499	500	0.0001	0.0255	0	0
512	1166	0.0059	0.0402	0.0197	0
522	1294	0.0013	0.0092	0.0042	0
522	-1295	0.0025	0.0175	0.008	0
533	1502	0.003	0.026	0.0566	0
549	1040	0.0038	0.0188	0.0083	0
555	-1342	0.0011	0.0071	0.0038	0
562	-1035	0.0039	0.0254	0.0134	0
576	-1180	0	0.0001	0.0001	0
640	641	0.0001	0.0014	0.001	0
640	-642	0.0001	0.0014	0.001	0
640	-643	0.0002	0.0021	0.0011	0
1000	1001	0.0044	0.0194	0.009	0
1001	-1002	0.0043	0.0167	0.0091	0
1002	1003	0.0017	0.0064	0.0118	0

Table A-6. Continued

FROM BUS	TO BUS	R (per-unit)	X (per-unit)	B (per-unit)	TAP
1003	1006	0.0023	0.011	0.005	0
1011	1168	0.0016	0.0046	0.0023	0
1018	-1135	0.012	0.0441	0.02	0
1021	-1072	0.0207	0.0836	0.036	0
1033	-1303	0.0064	0.0417	0.022	0
1035	-1270	0.0197	0.0579	0.0261	0
1059	1327	0.0036	0.0252	0.0218	0
1067	-1292	0.0005	0.0035	0.0018	0
1072	1219	0.0023	0.0114	0.0051	0
1079	-1298	0.0351	0.0838	0.0364	0
1094	1095	0.0031	0.0185	0.0089	0
1100	-1324	0.0018	0.011	0.005	0
1103	-1260	0.0011	0.007	0.0041	0
1103	1303	0.0061	0.0381	0.022	0
1109	1290	0.0031	0.024	0.011	0
1111	1191	0.003	0.0195	0.0102	0
1113	-1114	0.0086	0.0588	0.0271	0
9014	1116	0.0009	-0.0064	0	1
1121	1125	0.0038	0.024	0.0134	0
1127	1185	0.0009	0.0043	0.002	0
1127	1193	0.0086	0.043	0.0189	0
1127	1352	0	0.0002	0.0001	0
1137	1270	0.0104	0.04	0.018	0
1145	5640	0.0027	0.0144	0.0066	0
1149	1185	0.0001	0.0005	0.0002	0
1152	1271	0.0034	0.0225	0.0115	0
1163	1180	0.0003	0.0036	0.0022	0
1176	-1186	0.02	0.0999	0.0439	0
1177	-1294	0.0055	0.0274	0.0121	0
1202	1203	0.0026	0.0168	0.0089	0
1205	-1221	0.0042	0.0208	0.0092	0
1211	1282	0.0062	0.0421	0.0207	0
1225	1308	0.0051	0.0322	0.0178	0
1241	1242	0.0028	0.0129	0.0064	0
1242	-1320	0.0028	0.0129	0.0064	0
1255	-1262	0.0016	0.0109	0.0053	0
1270	1271	0.0024	0.015	0.0087	0
1289	1503	0.002	0.0187	0.0362	0
1291	1292	0.002	0.0131	0.0069	0
1362	-9999	0.0048	0.0226	0.0113	0
1500	-1503	0.0005	0.0047	0.0091	0
1502	-1504	0.0035	0.0307	0.0668	0
1502	1513	0.001	0.0509	0	0.9875
1504	1505	0.0029	0.0195	0.0407	0
1504	1509	0.0019	0.0162	0.0356	0
1504	1514	0.001	0.0506	0	0.9625

Table A-7. New England 39 Bus Test System Load Flow Bus Data

Bus ID	Name	kV base	Type	Pload	Qload	Pgen	Qgen	Qmax	Qmin	Vmag	Angle
1	'BUS_1 '	345	1							1.05409	-6.4508
2	'BUS_2 '	345	1							1.03208	-8.465
3	'BUS_3 '	345	1	322	2.4					1.0038	-10.5127
4	'BUS_4 '	345	1	500	184					0.96529	-10.405
5	'BUS_5 '	345	1							0.96348	-8.6068
6	'BUS_6 '	345	1							0.96456	-7.8545
7	'BUS_7 '	345	1	233.8	84					0.95663	-9.903
8	'BUS_8 '	345	1	522	176					0.95722	-10.2775
9	'BUS_9 '	345	1							1.01511	-7.2215
10	'BUS_10 '	345	1							0.97078	-5.4137
11	'BUS_11 '	345	1							0.9674	-6.2512
12	'BUS_12 '	230	1	7.5	88					0.9538	-6.3168
13	'BUS_13 '	345	1							0.96932	-6.2667
14	'BUS_14 '	345	1							0.97061	-8.2685
15	'BUS_15 '	345	1	320	153					0.9794	-8.9522
16	'BUS_16 '	345	1	329	32.3					0.99864	-7.5481
17	'BUS_17 '	345	1	158	30					1.00469	-8.9675
18	'BUS_18 '	345	1							1.00291	-10.0094
19	'BUS_19 '	345	1							0.9982	-1.8432
20	'BUS_20 '	345	1	628	103					0.99553	-2.8127
21	'BUS_21 '	345	1	274	115					1.00729	-4.1929
22	'BUS_22 '	345	1							1.0371	1.2293
23	'BUS_23 '	345	1	247.5	84.6					1.03334	1.5892
24	'BUS_24 '	345	1	308.6	-92.2					1.00582	-7.7924
25	'BUS_25 '	345	1	224	47.2					1.04585	-7.0351
26	'BUS_26 '	345	1	139	17					1.03639	-7.7083
27	'BUS_27 '	345	1	281	75.5					1.01588	-9.4842
28	'BUS_28 '	345	1	206	27.6					1.04173	-4.1475
29	'BUS_27 '	345	1	283.5	26.9					1.04402	-1.3616
30	'BUS_30 '	22	2	1104	250.5	1000	382.85	620	-485	1.03	-9.5049
31	'SWG_31 '	22	3	9.2	4.6	523.36	134.75	999	-999	0.982	0
32	'BUS_32 '	22	2			650	138.03	400	-315	0.983	2.4712
33	'BUS_33 '	22	2			632	37.73	400	-315	0.997	3.3515
34	'BUS_34 '	22	2			508	139.79	315	-250	1.012	2.3693
35	'BUS_35 '	22	2			650	301.35	400	-315	1.049	6.2534
36	'BUS_36 '	22	2			560	144.27	350	-250	1.063	9.522
37	'BUS_37 '	22	2			540	52.18	350	-250	1.028	-0.1927
38	'BUS_38 '	22	2			830	57.64	550	-400	1.026	5.7302
39	'BUS_39 '	22	2			250	63.89	155	-121	1.048	-5.1735

Table A-8. New England 39 Bus Test System Load Flow Line Data

From Bus	To Bus	ID	R Per-unit	X Per-unit	B Per-unit	Tap Per-unit
1	2	1	0.0035	0.0411	0.6987	0
1	39	1	0.001	0.025	0.75	0
2	3	1	0.0013	0.0151	0.2572	0
2	25	1	0.007	0.0086	0.146	0
3	4	1	0.0013	0.0213	0.2214	0
3	18	1	0.0011	0.0133	0.2138	0
4	5	1	0.0008	0.0128	0.1342	0
4	14	1	0.0008	0.0129	0.1382	0
5	6	1	0.0002	0.0026	0.0434	0
5	8	1	0.0008	0.0112	0.1476	0
6	7	1	0.0006	0.0092	0.113	0
6	11	1	0.0007	0.0082	0.1389	0
7	8	1	0.0004	0.0046	0.078	0
8	9	1	0.0023	0.0363	0.3804	0
9	39	1	0.0001	0.025	0.12	0
10	11	1	0.0004	0.0043	0.0729	0
10	13	1	0.0004	0.0043	0.0729	0
13	14	1	0.0009	0.0101	0.1723	0
14	15	1	0.0018	0.0217	0.366	0
15	16	1	0.0009	0.0094	0.171	0
16	17	1	0.0007	0.0089	0.1342	0
16	19	1	0.0016	0.0195	0.304	0
16	21	1	0.0008	0.0135	0.2548	0
16	24	1	0.0003	0.0059	0.068	0
17	18	1	0.0007	0.0082	0.1319	0
17	27	1	0.0013	0.0173	0.3216	0
21	22	1	0.0008	0.014	0.2565	0
22	23	1	0.0006	0.0096	0.1846	0
23	24	1	0.0022	0.07	0.361	0
25	26	1	0.0032	0.0323	0.513	0
26	27	1	0.0014	0.0147	0.2396	0
26	28	1	0.0043	0.0474	0.7802	0
26	29	1	0.0057	0.0625	1.029	0
28	29	1	0.0014	0.0151	0.249	0
19	20	1	0.0007	0.0138	0	0
6	31	1	0	0.025	0	1.007
10	32	1	0	0.02	0	1.007
19	33	1	0.0007	0.0142	0	1.007
20	34	1	0.0009	0.018	0	1.009
22	35	1	0	0.0143	0	1.025
23	36	1	0.0005	0.0272	0	1
25	37	1	0.0006	0.0232	0	1.025
2	30	1	0	0.0181	0	1.025
29	38	1	0.0008	0.0156	0	1.025
12	11	1	0.0016	0.0435	0	1.006
12	13	1	0.0016	0.0435	0	1.006

Appendix B

<u>Weighted Least Squares State Estimation Program</u>

This appendix contains the MATLAB program listing of the weighted least square estimator program as implemented in Chapter 5 of this thesis.

```
MVAbase = 100;
slack = 1111; %slack for 444 Bus Test Case
disp('Retrieving LOAD FLOW DATA ...');
MVA_base = 100;
bus = S.Bus.Number;
V(bus) = sparse(S.Bus.Vmag);
T(bus) = sparse(S.Bus.Angle);
disp('Retrieving LINE PARAMETERS ... ');
G = sparse(max(S.Bus.Number),max(S.Bus.Number));
B = G; Gs = G; Bs = G;
for i = 1:S.Branch.nl,
    p = abs(S.Branch.From(i)); % Always the tap side, t/l has tap=1
    q = abs(S.Branch.To(i));
    ypq = 1/S.Branch.Tap(i)*1/S.Branch.Z(i)*S.Branch.Status(i);
    yspq = (1-S.Branch.Tap(i))/S.Branch.Tap(i)^2*1/S.Branch.Z(i)*S.Branch.Status(i);
    ysqp = (S.Branch.Tap(i)-1)/S.Branch.Tap(i)*1/S.Branch.Z(i)*S.Branch.Status(i);
    G(p,q) = G(p,q) + real(ypq);
    B(p,q) = B(p,q) + imag(ypq);
    G(q,p) = G(p,q);
    B(q,p) = B(p,q);
    Bs(p,q) = Bs(p,q) + S.Branch.B(i)/2*S.Branch.Status(i) + imag(yspq);
    Gs(q,p) = Gs(p,q) + real(yspq);
    Bs(q,p) = Bs(q,p) + S.Branch.B(i)/2*S.Branch.Status(i) + imag(ysqp);
    Gs(q,p) = Gs(q,p) + real(ysqp);
end

disp('Generating Measurement Matrices & Indices to the Measurement Vector..');
 [mp,mq,mv] = M_PQV_S(B);
%RFN: DEBUG Temporarily remove the injection measurements
for i = 1:length(bus), mp(bus(i),bus(i))=0; end
for i = 1:length(bus), mq(bus(i),bus(i))=0; end
for i = 1:length(bus), mv(bus(i),bus(i))=0; end; mv(1,1)=1;
[PP,fip] = convert_M(mp,0);
[QQ,fiq] = convert_M(mq,fip);
[VV,nr] = convert_M(mv,fiq);
disp('Determining LINE FLOWS ...');
load S z;
[Pflow,Qflow] = flow_infS(z);
nb=size(bus,1);
disp('Calculating the FULL SCALE VALUEs of Meters ...');
[zdum,z1dum,injection] = flows_inject(S,MVAbase);
[s_lf,fs1,m1]=full_scalev4(S, injection, Pflow, Qflow, PP, QQ, VV, nb, MVA_base, V);
```

```
disp('Forming the Measurement Vector z...');
z = form_z(bus,PP,QQ,VV,Pflow,Qflow,injection,V,MVA_base);
disp('Determining the column indices of state variables in the Jacobian matrix H ..')
 [colt, colV] = statevar_indices(bus,slack);
disp('Adding noise to the measurements ...');
zm=noisy_meas(z,s_lf);%zm(1)=3; % add noise to the measurment Z
mm=[sqrt(zm(1:fip).^2+zm(fip+1:fiq).^2),sqrt(zm(1:fip).^2+zm(fip+1:fiq).^2) , zm(fiq+1:nr)];
%mm=noisy_meas(m,s_lf); %mm(1)=4;% add noise to the measurment Abs S
fs=sparse(noisy_meas(fs1,zeros(1,length(fs1)))); % resize the Full scale matrix
s=sparse((0.02*mm+0.0052*fs)/3);
Rs=sparse(diag(s.^2));
[R]=zero_flow(zm,Rs);
disp('Generating the initial value of measurement estimate h0...');
h0=get_h(S,V,T*pi/180,PP,QQ,VV);
X0=[full(T(find(colt))*pi/180) full(V(find(colV)))];
disp('... Now the Jacobian H0 of the measurement vector ...');
H0 = form_H3(PP,QQ,VV,S,T*pi/180,V,colt,colV);
condest(H0'*H0)
dx=2;
k=0
while (max(abs(dx))>5e-4)
        Xk1=X0(:)+(inv(H0'*inv(R)*H0)*H0'*inv(R)*(zm-h0)');
        dx=Xk1(:)-X0(:);
        XX=X0(:);
        mdx=max(abs(dx))
        X0=Xk1(:);
        Thn=ins_slack(X0(1:nb-1),slack,T,colt);
        E0(find(colV)) = sparse(X0(nb:2*nb-1));
        h0=get_h(S,E0,thn,PP,QQ,VV);
        H0 = form_H3(PP,QQ,VV,S,thn,E0,colt,colV);
        condest(H0'*H0)
        k=k+1
end
X2n=[nonzeros(thn*180/pi) nonzeros(E0')];

function [P,final_indx] = convert_M(P,start_indx)
% Function to convert a binary measurement matrix Mij to
% a measurement matrix with elements corresponding to the
% row indices of the corresponding measurements to h(x)
%P = sparse(P);
k = start_indx;
[i,j]= find(P);
[m,n]=size(P);
inj = find(i==j);
i = sparse(i); j = sparse(j); i(inj) = 0; j(inj) = 0;
ii=full(i);
jj=full(j);
kk = k+1:k+nnz(P)-length(inj);
if( length(kk) ~= 0 )
   P = sparse(jj,ii,kk,m,n);
end
```

```
if length(kk) == 0
   ks = start_indx;
else
   ks = kk(length(kk));
end
for h = 1:length(inj);
   P(inj(h),inj(h)) = ks + h;
end
final_indx = k + nnz(P);
return


function [z,z1,injection] = flows_inject(S,MVAbase)
% [z,z1,injection] = flows_inject(S,MVAbase)
% Calculates the 1) line flow indexed by p to q in z
%                   OR indexed by line number in z1
%                   (breakdown of parallel lines reported in the columns )
%                2) injection(bus #)- net injection at bus #.
%                    Shown as (+) if going into the bus OR (-) if out of the bus
%
z = sparse(max(S.Bus.Number),max(S.Bus.Number));
V = S.Bus.Vmag.*(cos(S.Bus.Angle*pi/180) + sqrt(-1)*sin(S.Bus.Angle*pi/180));
for i = 1:S.Branch.nl,
   if S.Branch.Status(i) ~= 0
      p = abs(S.Branch.From(i));
      q = abs(S.Branch.To(i));
      ip = find(S.Bus.Number==p);
      iq = find(S.Bus.Number==q);
      yl = 1/( S.Branch.R(i) + sqrt(-1)*S.Branch.X(i) );
      a = S.Branch.Tap(i);
      Y12 = yl/a;
      Y11 = sqrt(-1)*S.Branch.B(i)/2 + (a-1)/a*yl;
      Y22 = sqrt(-1)*S.Branch.B(i)/2 + (1-a)/a^2*yl; % The tapped side
      Ipq = (V(ip) - V(iq) ) * Y12;
      Ips = V(ip) * Y22;
      Iqs = V(iq) * Y11;
      z(p,q) = z(p,q) + V(ip)*conj(Ipq+Ips); % To account for parallel lines
      z(q,p) = z(q,p) + V(iq)*conj(-Ipq+Iqs);
      z1(i,1) = z(p,q);
      z1(i,2) = z(q,p);
      % Flows to shunt in the diagonal of z
      Ipgb = V(ip)*( S.Bus.G(ip) + sqrt(-1)*S.Bus.B(ip) );
      z(p,p) = V(ip)*conj(Ipgb);
      % G&B inputted in MW, MVAR(+,-), + for capacitor
      Iqgb = V(iq)*( S.Bus.G(iq) + sqrt(-1)*S.Bus.B(iq) );
      z(q,q) = V(iq)*conj(Iqgb);
   end
end
z = z*MVAbase;
z1 = z1*MVAbase;
```

```
injection = sparse(0);
for i = 1:S.Bus.n,
   b = S.Bus.Number(i);
   injection(b) = sum(z(b,:));
end

function [s,fsv,m]=full_scalev4(S, injection, Pflow, Qflow, PP, QQ, VV, nb, MVA_base, V )
busn = S.Bus.Number;
Vbase = sparse(0); Vbase(busn) = S.Bus.kVbase;
%pause
kV_base = unique(Vbase);
%pause
genn = S.Gen.Bus;
%pause
% Identify generator step-up transformers
step_up = sparse(1,S.Branch.nl);
S.Branch.From = abs(S.Branch.From);
S.Branch.To = abs(S.Branch.To);

for i = 1:S.Branch.nl,
   if ismember(S.Branch.From(i),genn) | ismember(S.Branch.To(i),genn)
     if ismember(S.Branch.From(i),genn) & Vbase(S.Branch.From(i)) ~= Vbase(S.Branch.To(i))
        step_up(i) = 1;
     end
     if ismember(S.Branch.To(i),genn) & Vbase(S.Branch.From(i)) ~= Vbase(S.Branch.To(i))
        step_up(i) = 1;
     end
   end
end
if exist('S_fs.mat') ~= 2
  disp('Full scale flow FILENAME not found!');
  disp('Will generate automatically...');
  fs = gen_fs(S,step_up,Pflow,Qflow,MVA_base);
  save S_fs.mat fs
else
  load S_fs.mat ;
end
Sflow = Pflow + sqrt(-1)*Qflow;
% Normalize the flows, Compute <m>
for l = 1:S.Branch.nl,
  %if( ~step_up(l) ) % Not a Gen xfmr
    p = abs(S.Branch.From(l)); q = abs(S.Branch.To(l));
    %ip = find(kV_base-Vbase(p)==0);
    %iq = find(kV_base-Vbase(q)==0);
    if( PP(p,q) ~= 0 )
      irow = PP(p,q);
      m(irow) = abs( Sflow(p,q)/MVA_base );
      fsv(irow) = fs(Vbase(p));
    end
      if( PP(q,p) ~= 0 )
      irow = PP(q,p);
```

```matlab
      m(irow) = abs( Sflow(q,p)/MVA_base );
      fsv(irow) = fs(Vbase(q));
    end
   if( QQ(p,q) ~= 0 )
      irow = QQ(p,q);
      m(irow) = abs( Sflow(p,q)/MVA_base );
      fsv(irow) = fs(Vbase(p));
    end
   if( QQ(q,p) ~= 0 )
      irow = QQ(q,p);
      m(irow) = abs( Sflow(q,p)/MVA_base );
      fsv(irow) = fs(Vbase(q));
    end
    %end
end
% Now on Step-Up Transformers
% Normalize and calculate <m>
% Now go over the injections
Sflow1=Pflow+sqrt(-1)*Qflow;
Snet=abs(sum(Sflow1.'))/MVA_base; % Pnet is the Gen - Load
for i = 1:length(busn)
  b = busn(i);
  div = fs(Vbase(b));
  if(PP(i,i) ~= 0 )
     m(PP(i,i)) = real(injection(b)) / MVA_base;
     fsv(PP(i,i)) = div;
  end
  if(QQ(i,i) ~= 0 )
     m(QQ(i,i)) = imag(injection(b)) / MVA_base;
     fsv(QQ(i,i)) = div;
  end
end
% Finally the voltages
for i=1:nb
  irow=VV(busn(i),busn(i));
  if(irow~=0)
     m(irow)=sparse(V(busn(i)));
     fsv(irow) = 1.2;
  end
end
% Obviously, full scale is 1.0 for all meas except voltage
%s=(0.02*m+0.0052*fs)/3;
for i = 1:length(m),
  s(i) = (0.02*m(i)+0.0052*fsv(i)) / 3;
end


function [z,zinfo] = form_z(bus,MP,MQ,MV,Pflow,Qflow,injection,V,MVA_base)
[i,j,irow] = find(MP);
for k = 1:length(irow),
   if i(k) ~= j(k)
```

```
        z(irow(k))=Pflow(i(k),j(k))/MVA_base;
        zinfo(irow(k)) = struct('class',{'CSE'},'type',{'Pflw'},'i',i(k),'j',j(k));
    else
        z(irow(k))=real(injection(i(k)))/MVA_base;
        zinfo(irow(k)) = struct('class',{'CSE'},'type',{'Pinj'},'i',i(k),'j',i(k));
    end
end
 [i,j,irow] = find(MQ);
for k = 1:length(irow),
    if i(k) ~= j(k)
        z(irow(k))=Qflow(i(k),j(k))/MVA_base;
        zinfo(irow(k)) = struct('class',{'CSE'},'type',{'Qflw'},'i',i(k),'j',j(k));
    else
        z(irow(k))=imag(injection(i(k)))/MVA_base;
        zinfo(irow(k)) = struct('class',{'CSE'},'type',{'Qinj'},'i',i(k),'j',i(k));
    end
end
for i =1:length(bus)
    irow = MV(bus(i),bus(i));
    if( irow ~= 0 )
        z(irow) = V(bus(i));
        zinfo(irow) = struct('class',{'CSE'},'type',{'Vmag'},'i',bus(i),'j',bus(i));
    end
end
return


function [zerr]=noisy_meas(z,sigma)
randn('state',sum(100*clock))
for i=1:length(z)
    if (abs(z(i))>0)
        if ~exist('rnP.mat')
            r = randn;
        else
            r = ran(i);
        end
        zerr(i)=z(i)+sigma(i)*r;
    end
end


function hs=get_h_S(S,V,th,PP,QQ,VV)
%function hs=get_h_S(S,V,th,PP,QQ,VV,Pflow,Qflow)
bus = S.Bus.Number;
E = sparse(V.*(cos(th)+sqrt(-1)*sin(th)));
P = sparse(max(S.Bus.Number),max(S.Bus.Number));
Q = P;
for k = 1:S.Branch.nl
    i = abs(S.Branch.From(k));
    j = abs(S.Branch.To(k));
    t = S.Branch.Tap(k);
```

```
    Y = 1/S.Branch.Z(k);
    Sij = E(i) * conj( E(i)*( sqrt(-1)*S.Branch.B(k)/2 + (1-t)/t^2*Y ) + (E(i)-E(j))*Y*1/t) *
          S.Branch.Status(k);
    Sji = E(j) * conj( E(j)*( sqrt(-1)*S.Branch.B(k)/2 + (t-1)/t*Y ) + (E(j)-E(i))*Y*1/t) *
          S.Branch.Status(k);
    P(i,j) = P(i,j) + real(Sij);
    P(j,i) = P(j,i) + real(Sji);
    Q(i,j) = Q(i,j) + imag(Sij);
    Q(j,i) = Q(j,i) + imag(Sji);
end

% To account for bus shunts
for k = 1:S.Bus.n
    i = S.Bus.Number(k);
    Sii = E(i)*conj( E(i)*( S.Bus.G(k) + sqrt(-1)*S.Bus.B(k) ) );
    P(i,i) = P(i,i) + real(Sii);
    Q(i,i) = Q(i,i) + imag(Sii);
end

Pnet=sum(P.'); % Pnet is the Gen - Load
Qnet=sum(Q.'); % Qnet is the Gen - Load +/- (Fixed shunt capacitor)

[i,j,irow] = find(PP);
for k = 1:length(irow),
    if i(k) ~= j(k)
        hs(irow(k)) = P(i(k),j(k));
    else
        hs(irow(k)) = Pnet(i(k));
    end
end

[i,j,irow] = find(QQ);
for k = 1:length(irow),
    if i(k) ~= j(k)
        hs(irow(k)) = Q(i(k),j(k));
    else
        hs(irow(k)) = Qnet(i(k));
    end
end


for k=1:S.Bus.n,
    i = bus(k);
    irow = VV(i,i);
    if(irow~=0)
        hs(irow)=sparse(V(i));
    end
end


function [H]= form_H3(PP,QQ,VV,S,th,E,colt,colV)
```

```
%function [H]= form_H3(mp,mq,mv,G,B,Gs,Bs,th,E,colt,colV,trn1,trfi)
% Assumes that the non-zero elements of P measurement matrix contains the
% indices of each measurement in the h(x) matrix
% Form the Jacobian of h(x)
% Structure of H
%Col.      1 2 3    n-1 n  n+1  2n-1
%State     t1 t2 t3 ... tn-1 E1 E2 ... En-1
%      k   +--+--+--+---+----+--+--+----+---+
%P12   1  | | | | | |   | | |  | |
%P13   2  | | | | | |   | | |  | |
% :    :
%P1n   n
%P21   n+1
% :
%P2n
% :
%Pnn
%P11
%P22
% :
%Pnn
%Q12
%Q13
% :
%Q1n
%Q21
% :
%Q2n
% :
%Qnn
%Q11
% :
%Qnn
%V1
%V2
% :
%Vn


H(1,1)=sparse(0);
dP_dt = sparse(1,max(S.Bus.Number));
dP_dE = dP_dt;
dQ_dt = dP_dt;
dQ_dE = dP_dt;
% Derivatives of Pij first
[J,I,irow]= find(PP'); % List and location of real power flow measurements
for k = 1:length(irow),
   i = I(k); j = J(k);
   br = find(abs(S.Branch.From)==i & abs(S.Branch.To)==j);
   if length(br) == 0
      br = find(abs(S.Branch.From)==j & abs(S.Branch.To)==i);
```

```
        end

    if i ~= j  % Derivative of Real Power Flow
        dPij_dti = 0; dPij_dtj = 0;
        dPij_dEi = 0; dPij_dEj = 0;
        for m = 1:length(br)
            Gij = real(1/S.Branch.Z(br(m))) * S.Branch.Status(br(m));
            Bij = imag(1/S.Branch.Z(br(m))) * S.Branch.Status(br(m));
            % WRT to thetas
            dPij_dti = dPij_dti -E(i)*E(j)*1/S.Branch.Tap(br(m)) * ( Bij*cos(th(i)-th(j)) -
             Gij*sin(th(i)-th(j)) );
            dPij_dtj = dPij_dtj -dPij_dti;
            dP_dt(i) = dP_dt(i) + dPij_dti; % Use to calculate derivative of Pinj wrt theta
            % WRT to Es
            dPij_dEi =  dPij_dEi + 2*Gij*E(i) - E(j)*1/S.Branch.Tap(br(m)) * ( Gij*cos(th(i)-th(j)) +
             Bij*sin(th(i)-th(j)) );
            dPij_dEj =  dPij_dEj  -E(i)*1/S.Branch.Tap(br(m)) * ( Gij*cos(th(i)-th(j)) + Bij*sin(th(i)-
             th(j)) );
            dP_dE(i) = dP_dE(i) + dPij_dti; % Use to calculate derivative of Pinj wrt E

        end
        % Place into H matrix
        if( colt(i) ~= 0 )
            H(irow(k),colt(i)) = dPij_dti;
        end
        if( colt(j) ~= 0 )
            H(irow(k),colt(j)) = dPij_dtj;
        end
        H(irow(k),colV(i)) = dPij_dEi;
        H(irow(k),colV(j)) = dPij_dEj;
    else %Derivative of Real Power Injection Pi
        H(irow(k),colV(i)) = sum(dP_dE(i)); % wrt to Vi
        if colt(i) ~= 0
            H(irow(k),colt(i)) = sum(dP_dt(i)); % wrt to Ti
        end
        jj = find(PP(i,:)); % Buses with links to i
        H(irow(k),colV(jj)) = H(PP(i,jj),colV(jj));
        for n = 1:length(jj),
            if colt(jj(n))
                prev_row = PP(i,jj(n));
                H(irow(k),colt(jj(n))) = H(prev_row,colt(jj(n)));
            end
        end
    end

end

% THEN Derivatives of Qij
[J,I,irow]= find(QQ'); % List and location of real power flow measurements
for k = 1:length(irow),
    i = I(k); j = J(k);
```

```
      br = find(abs(S.Branch.From)==i & abs(S.Branch.To)==j);
      if length(br) == 0
         br = find(abs(S.Branch.From)==j & abs(S.Branch.To)==i);
      end


      if i ~= j  % Derivative of Imaginary Power Flow
         dQij_dti = 0; dQij_dtj = 0;
         dQij_dEi = 0; dQij_dEj = 0;
         for m = 1:length(br)
            Gij = real(1/S.Branch.Z(br(m))) * S.Branch.Status(br(m));
            Bij = imag(1/S.Branch.Z(br(m))) * S.Branch.Status(br(m));
            % WRT to thetas
            dQij_dti = dQij_dti -E(i)*E(j)*1/S.Branch.Tap(br(m)) * ( Gij*cos(th(i)-th(j)) +
             Bij*sin(th(i)-th(j)) );
            dQij_dtj = dQij_dtj -dQij_dti;
            dQ_dt(i) = dQ_dt(i) + dQij_dti; % Use to calculate derivative of Qinj wrt theta
            % WRT to Es
            dQij_dEi =  dQij_dEi -2*(S.Branch.B(br(m))/2+Bij)*E(i) + E(j)*1/S.Branch.Tap(br(m)) *
             ( Bij*cos(th(i)-th(j)) - Gij*sin(th(i)-th(j)) );
            dQij_dEj =  dQij_dEj +E(i)*1/S.Branch.Tap(br(m)) * ( Bij*cos(th(i)-th(j)) - Gij*sin(th(i)-
             th(j)) );
            dQ_dE(i) = dQ_dE(i) + dQij_dti; % Use to calculate derivative of Qinj wrt E
         end
         % Place into H matrix
         if( colt(i) ~= 0 )
            H(irow(k),colt(i)) = dQij_dti;
         end
         if( colt(j) ~= 0 )
            H(irow(k),colt(j)) = dQij_dtj;
         end
         H(irow(k),colV(i)) = dQij_dEi;
         H(irow(k),colV(j)) = dQij_dEj;
      else %Derivative of Imaginary Power Injection Qi
         H(irow(k),colV(i)) = sum(dQ_dE(i)); % wrt to Vi
         if colt(i) ~= 0
            H(irow(k),colt(i)) = sum(dQ_dt(i)); % wrt to Ti
         end
         jj = find(QQ(i,:)); % Buses with links to i
         H(irow(k),colV(jj)) = H(QQ(i,jj),colV(jj));
         for n = 1:length(jj),
            if colt(jj(n))
               prev_row = QQ(i,jj(n));
               H(irow(k),colt(jj(n))) = H(prev_row,colt(jj(n)));
            end
         end
      end
   end
end

% Finally, simple identity matrix for measured Es
[I,J,irow] = find(VV);
if length(irow) ~= 0
```

```
   for k = 1:length(irow),
      i = I(k);
      H(VV(i,i),colV(i)) = 1;
   end
end
return

function [thn]=ins_slack(X01nbm1,slack,T,colt)
ref=T(slack)*pi/180;
thn(find(colt)) = sparse(X01nbm1);
thn(slack) = ref;
return
```

Appendix C
Building a Decision Tree

This appendix discusses two key areas in decision tree building: node splitting and tree-pruning. Much of the theory behind these methods is discussed in detail in [9][35].

A. Node Splitting

Define N as the total number of measurement cases, and its subset $N(t)$ as the number of measurement cases going into node t. Also, define $N_1(t)$ and $N_2(t)$ as the number of voltage secure and voltage insecure cases going into the same node. The impurity of a node t, $I(t)$, is expressed in terms of the entropy function ϕ shown in (A.1).

$$I(t) = \Phi(p_1, p_2) = -\sum_{j=1}^{2} p_j \ln p_j \qquad (A.1)$$

$p_1 = N_1(t)/N(t)$ is the proportion of all cases in node t having a voltage secure status, while $p_2 = N_2(t)/N(t)$ is the proportion of voltage insecure cases. ϕ is minimum when the node t is class pure, that is, either $p_1 = 1$ or $p_2 = 1$. The overall impurity of a tree T is the sum of impurities of its terminal nodes t, that is,

$$I(T) = \sum_{t \in \tilde{T}} I(t) \qquad (A.2)$$

When node t is split, a proportion $p_L = N_L(t)/N(t)$ of node t cases goes to the left child node t_L, while a proportion $p_R = N_R(t)/N(t)$ cases goes to the right child node t_R. If we define the resulting impurity of both child nodes as $I(t_L)$ and $I(t_R)$ respectively, then the change in impurity due to the split s, $\Delta I(s,t)$ is given by (A.3).

$$\Delta I(s,t) = I(t) - p_L I(t_L) - p_R I(t_R) \qquad (A.3)$$

Figure A-1 graphically illustrates how a split s(a) on attribute a will distribute the measurement cases into the left and right nodes, each node assuming an impurity and from which the change in impurity due to split is calculated as shown.

I(t)   t:N(t)

p1(t)=N1(t)/N(t)   N1(t)   N2(t)   p2(t)=N2(t)/N(t)

p1(t)   p2(t)

Is x(a) ≤s(a)?

pL=N(tL)/N(t)   pR=N(tR)/N(t)

I(tL)   tL:N(tL)   I(tR)   tR:N(tR)

N1(tL)   N2(tL)   N1(tR)   N2(tR)

ΔI(s,t) = I(t) - pL*I(tL) - pR*I(tR)
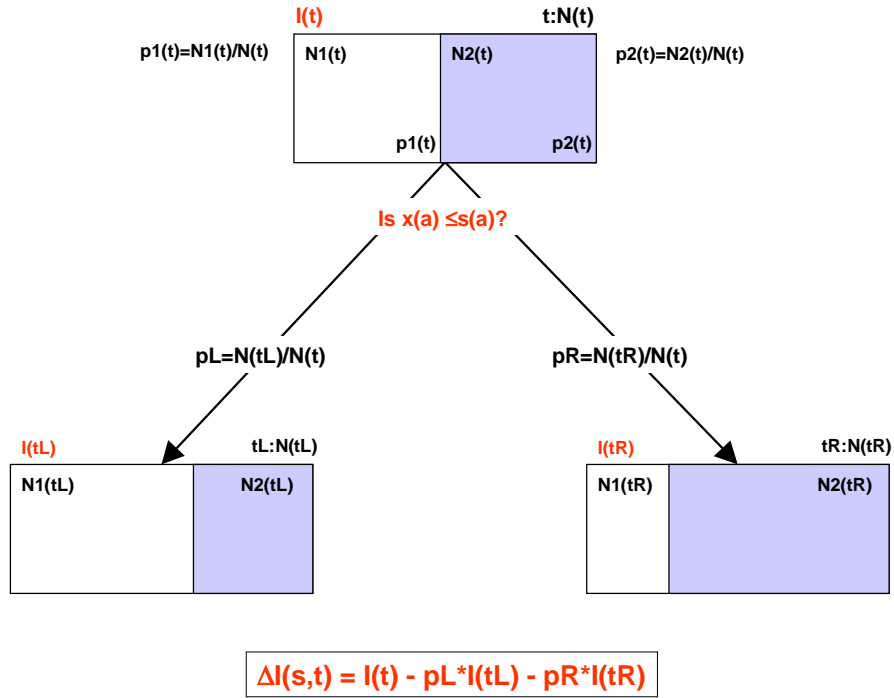
Figure A-1. Node Splitting: Calculating the Change in Impurity Due to Split s(a)

Thus the objective at each node is to find the optimal s* that will result in greatest reduction in impurity (A.4). This is repeated for all attributes; the optimal split of that attribute resulting in the maximum value of $\Delta I(s^*,t)$ is chosen for this node.

$$\Delta I(s^*,t) = \max_{s \in S} \Delta I(s,t) \qquad (A.4)$$

B. Pruning

The objective of pruning is to extract a smaller subtree T with a better reliability than the parent tree T_{max}. Pruning is accomplished by generating a set of subtrees and then identifying that tree from this set that will best perform in classifying the test data.

Define *m(t)=1-max(p1,p2)* as the probability of misclassification at node t. Also, define *U(t)=m(t)*N(t)/N*. The misclassification rate or "cost" of any tree T, U(T), is the sum of each of the cost of its terminal nodes $|\check{T}|$ (A.5).

$$U(T) = \sum_{t \in \bar{T}} U(t) \qquad (A.5)$$

Define the cost complexity measure of any tree T as a linear combination of its cost and its complexity (the number of terminal nodes $|\breve{T}|$) (A.6).

$$U_\alpha(T) = U(T) + \alpha|\breve{T}| \qquad (A.6)$$

For any value of α, a minimizing subtree $T(\alpha)$ can be found using (A.7).

$$U_\alpha(T(\alpha)) = \min_{T \subset T_{max}} U_\alpha(T) \qquad (A.7)$$

To find the first minimizing subtree T_1, prune those terminal nodes that satisfy (A.8), if any. These will be those terminal child nodes with combined cost equal to the cost of their parent node.

$$U(t) = U(t_L) + U(t_R) \qquad (A.8)$$

The next minimizing subtree is found by applying (A9) to each internal node in T_1 and choosing that internal node t which yields the smallest α_t. (T_t is the subtree pruned from T_1).

$$\alpha_t = \frac{U(t) - U(T_t)}{|\breve{T}_t| - 1} \qquad (A.9)$$

All descendant nodes of t are pruned resulting in the subtree $T\text{-}T_t$. This process is repeated on $T\text{-}T_t$ subsequently yielding a smaller subtree until finally the root node is reached.

At this point a sequence of minimizing subtrees were generated. A test data set is submitted for classification to each subtree. The one yielding the lowest misclassification rate is chosen as the optimal-sized tree.

Figure A-2 (A) illustrates a tree T with root node t_0 and a subtree T_{t1} enclosed in dashed line. Figure A-2 (B) shows the tree $T\text{-}T_{t1}$ after pruning the subtree T_{t1} into a terminal node t_1.
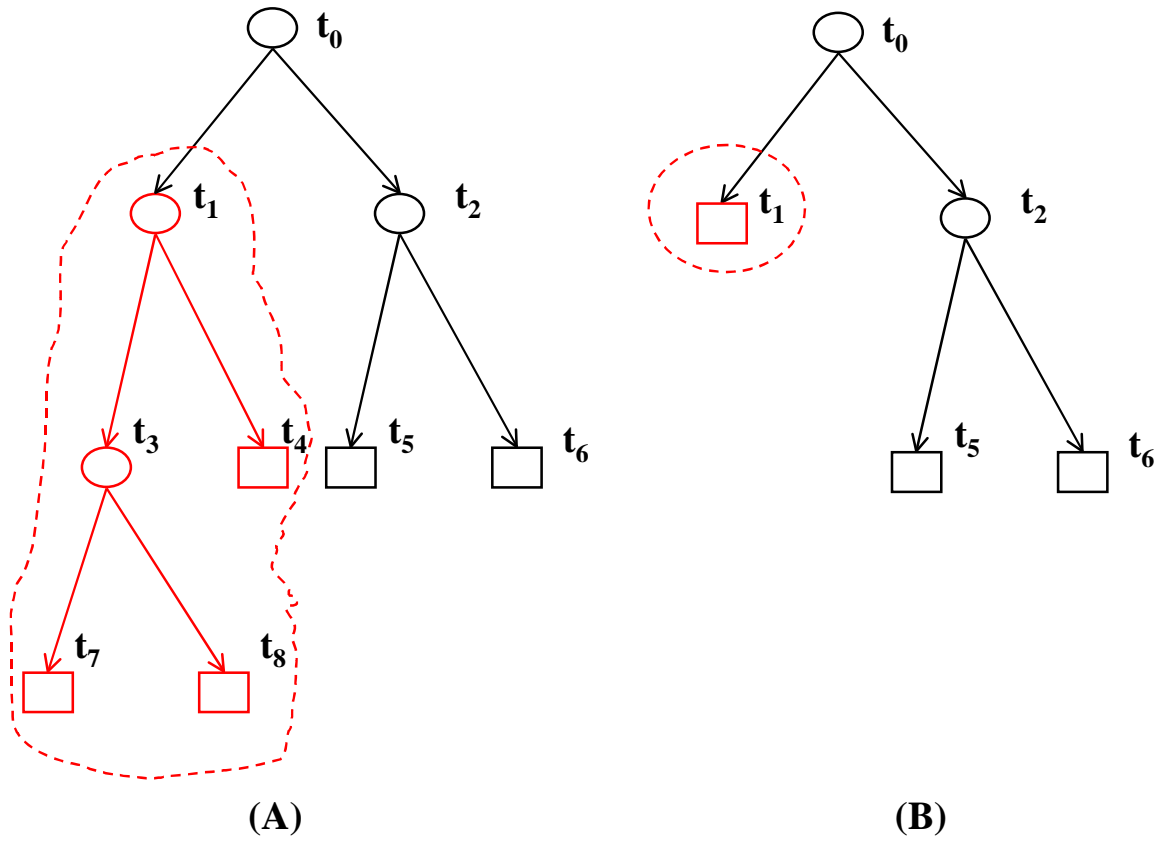
Figure A-2. Tree Pruning: (A) Tree T with Subtree Tt1 shown; (B) Pruned tree T-Tt1 with subtree Tt1 pruned into a terminal node t1

Appendix D
C Code Listing for Building a Classification Type Decision Tree

This appendix contains the program listing in C language of the classification-type decision tree algorithm discussed in Appendix C and implemented in Chapter 4 of this thesis. The code implements the following:

      1. tree growing technique via node splitting

      2. finding the right size tree via pruning

```c
// ==========================================================
// Program MAIN
#include <stdio.h>
#include <stdlib.h>
#define MAX_NODE 500
#define CRIT 0.1  /* Any node with impurity > than 0.1 will not be split */
float main ()
{
        float fmin = 0;
        int i,ii, j;
        int M,N, Mt, Nt;
        int maxn;
        double **Lt, **Ltest;
        int *Vt, *Vtest;
        int nT = 0;
        int nnodes;
        float fp;
        int c0, c1;
        double criterion = CRIT;
        int **lset;
        float cost;
        int **desc, *nd;
        int **set_term;
        int nstrees;
        int *terminal;
        int nterm_nodes;
        int node_id[MAX_NODE]; /* Identification of node t*/
        int parent[MAX_NODE]; /* Parent node of node t */
        int **child; /* child nodes of node t child[t][1..2] */
        // Optimal split VALUE of attribute a* at node t indexed by node number t
        double split_value[MAX_NODE];
        // Optimal split atrribute a* at node t indexed by node number t
        int split_var[MAX_NODE];
        // Change in impurity at node t indexed by by node number t
        double change_Imp[MAX_NODE];
        // Impurity at node t indexed by by node number t
        double Imp_node[MAX_NODE];
```

```
// Terminal Node Tag: 1- terminal; 0 - not a terminal node */
int termnode_tag[MAX_NODE];
// No of descendant terminal nodes of indexed by node t
int nterm[MAX_NODE];
// Number of node t's learning states indexed by node number of t */
int NN[MAX_NODE];
// Number of node t's secure states indexed by node number of t */
int N1[MAX_NODE];
// Number of node t's insecure states indexed by node number of t */
int N2[MAX_NODE];
// Number of node t's learning data indexed by node number of t */
int nlset[MAX_NODE];
// Misclassfication cost of node t indexed by node number of t */
float U[MAX_NODE];
// Subtree ID by shrinked node ID
int subtrees[MAX_NODE];
// Misclassfication cost of subtree t indexed by node number of shrinked node t float
UT[MAX_NODE]; /
// Number of node t's secure states indexed by node number of t
int N1test[MAX_NODE];
// Number of node t's insecure states indexed by node number of t */
int N2test[MAX_NODE];
float MR[MAX_NODE];
int FA[MAX_NODE];
FILE *stream, *stream1, *stream2, *stream3, *stream5, *stream6;
char *filnam, *filnam1, *filnam5, *filnam6;


// Function Declarations
int tree4aep(double **Lt, int *Vt, int M, int N, int maxn,  int *N1, int *N2,
                int *node_id, int *parent, int **child,  double *split_value,
                int *split_var, double *Imp_node, double *change_Imp,
                int *termnode_tag, int **lset, int *nlset, double criterion );
void set_termnodes( int *termnode_tag, int *node_id, int nnodes, int *parent,
                    int **set_term, int *nterm );
float misclassrate( int *node_id, int nnodes, int *V, int N,
                    int *nlset, int **lset, int *termnode_tag, float *U);
void descendant(int *termnode_tag, int *node_id, int nnodes, int *parent,
                    int **child, int **desc, int *nd);
int prune( FILE *stream, int *N1, int *N2, int **child, int *split_var,
            double *split_value, float *U, int **set_term, int *nterm,
            int *node_id, int nnodes, int *parent, int **desc, int *nd,
            int *termnode_tag, int *subtrees, float *UT );
void populate( int *terminal, int nterm_nodes, int *node_id, int nnodes,
                int *parent, int **child, double *split_value, int *split_var,
                double **L, int *Vclass, int ntset, int *N1, int *N2, int *N );

void right_size( FILE *stream, int *node_id, int nnodes,
                    int *N1, int *N2, float *MR, int *FA );


maxn = MAX_NODE;
```

```c
printf("Enter Measurement Training Space filename:");
scanf("%50s",filnam);
if( (stream = fopen(filnam,"r")) == NULL )
{
        printf("File not found\n");
        return fmin;
}
// Determine the size of measurement space
fscanf(stream, "%d %d", &M, &N);
// printf("Measurement space is %4d x %4d\n", M, N );
// Allocate memory to L
Lt = (double **)malloc(M * sizeof(double *) );
if( Lt == NULL) printf("Memory Allocation Problem for Lt\n");
for(i=0;i<M;i++)
{       Lt[i] = (double *)malloc(N * sizeof(double));
        if( Lt == NULL) printf("Memory Allocation Problem for Lt\n");
}
for(i=0;i<M;i++)
{       for(j=0;j<N;j++)
        {       fscanf(stream,"%f",&fp);
                Lt[i][j] = fp;
                /*printf("%f\n",Lt[i][j]); */
        }
}
// Allocate memory to Vt
Vt = (int *)malloc(N * sizeof(int) );
printf("Enter Training Class Vector filename: ");
scanf("%50s",filnam1);
if( (stream1 = fopen(filnam1,"r")) == NULL )
{       printf("File not found\n");
        return fmin;
}
for(j=0;j<N;j++)
{       fscanf(stream1,"%d",Vt+j);
}
fclose(stream);
fclose(stream1);
// Allocate memory to child
child = (int **)malloc(MAX_NODE * sizeof(int *) );
if( child == NULL) printf("Memory Allocation Problem for child\n");
for(i=0;i<MAX_NODE;i++)
{       child[i] = (int *)malloc(2 * sizeof(int));
        if( child == NULL) printf("Memory Allocation Problem for child\n");
}
// Allocate memory to lset
lset = (int **)malloc(MAX_NODE * sizeof(int *) );
if( lset == NULL) printf("Memory Allocation Problem for lset\n");
for(i=0;i<MAX_NODE;i++)
{       lset[i] = (int *)malloc(MAX_NODE * sizeof(int));
        if( lset == NULL) printf("Memory Allocation Problem for lset\n");
}
```

```
nnodes = tree4aep(Lt, Vt, M, N, maxn, N1, N2, node_id, parent, child,
                           split_value,  split_var, Imp_node, change_Imp,
                           termnode_tag, lset, nlset, criterion );
// Gather the terminal nodes
for(ii=0; ii<nnodes; ii++)
{       i = node_id[ii];
        if( termnode_tag[i] == 1 )
        {       child[i][0] = 0;
                child[i][1] = 0;
                nT++;
        }
}
stream3 = fopen("cart.out","w");
fprintf(stream3,"CODED DECISION TREE STRUCTURE\n");
for(i=0;i<nnodes;i++)
{       c0 = child[i][0];
        c1 = child[i][1];
        fprintf(stream3,"%4d %4d %4d %4d %10.4f %10.8f %4d %4d\n",
        node_id[i], c0, c1, split_var[i], split_value[i], Imp_node[i], N1[i], N2[i]);
}
// Allocate memory to set_term
set_term = (int **)malloc(MAX_NODE * sizeof(int *) );
if( set_term == NULL) printf("Memory Allocation Problem for set_term\n");
for(i=0;i<MAX_NODE;i++)
{       set_term[i] = (int *)malloc(MAX_NODE * sizeof(int));
        if( set_term == NULL)
                printf("Memory Allocation Problem for set_term\n");
}
// Find the set of terminal nodes w/c are descendants of each internal node t
set_termnodes( termnode_tag, node_id, nnodes, parent,  set_term, nterm );
//  Calculate the probability of misclassfication of each node t --> U[t]
    and the overall [cost] of this tree
cost = misclassrate( node_id, nnodes, Vt, N, nlset, lset, termnode_tag, U);
// Allocate memory to desc & nd
nd = (int *)malloc(MAX_NODE * sizeof(int) );
desc = (int **)malloc(MAX_NODE * sizeof(int *) );
if( desc == NULL) printf("Memory Allocation Problem for desc\n");
for(i=0;i<MAX_NODE;i++)
{       desc[i] = (int *)malloc(MAX_NODE * sizeof(int));
        if( desc == NULL) printf("Memory Allocation Problem for desc\n");
}
//  Find the set of nodes DESCENDANT to all internal nodes, desc[][] nd[]
descendant(termnode_tag, node_id, nnodes, parent, child, desc, nd);
// Allocate memory to terminal
terminal = (int *)malloc(nnodes * sizeof(int) );
nterm_nodes = 0;
for(i=0; i<nnodes; i++)
{       if( termnode_tag[i] == 1 )
        {       nterm_nodes++;
                terminal[nterm_nodes-1] = node_id[i];
                nd[i] = 0;
```

```
                        desc[i][0] = 0;
            }
      }
// Determine the set of MINIMIZING SUBTREES
stream2 = fopen("subtrees.prn","w");
printf("FINDING MINIMIZING SUBTREES & STORING IN FILE...\n");
nstrees = prune( stream2, N1, N2, child, split_var, split_value,
                        U, set_term, nterm, node_id, nnodes, parent, desc, nd,
                        termnode_tag, subtrees, UT );
fclose(stream2);
fprintf(stream3, "\nTHE MINIMIZING SUBSTREES\n");
for(i=0;i<nstrees;i++)
{       fprintf(stream3, "Subtree: %4d: Misclass Cost:%10.4f\n", subtrees[i],
                  UT[subtrees[i]]);
}

printf("Enter Measurement Test Space filename: ");
scanf("%50s",filnam5); */
if( (stream5 = fopen(filnam5,"r")) == NULL )
{       printf("File not found\n");
        return fmin;
}
fscanf(stream5, "%d %d", &Mt, &Nt);
// Allocate memory to L
Ltest = (double **)malloc(Mt * sizeof(double *) );
if( Ltest == NULL) printf("Memory Allocation Problem for Ltest\n");
for(i=0;i<Mt;i++)
{       Ltest[i] = (double *)malloc(Nt * sizeof(double));
        if( Ltest == NULL) printf("Memory Allocation Problem for Ltest\n");
}
for(i=0;i<Mt;i++)
{       for(j=0;j<Nt;j++)
        {       fscanf(stream5,"%f",&fp);
                Ltest[i][j] = fp;
        }
}
scanf("%50s",filnam6);
if( (stream6 = fopen(filnam6,"r")) == NULL )
{       printf("File not found\n");
        return fmin;
}
// Allocate memory to Vtest
Vtest = (int *)malloc(Nt * sizeof(int) );
for(j=0;j<Nt;j++)
        fscanf(stream6,"%d",Vtest+j);
printf("POPULATING THE MAX TREE WITH THE TEST DATA SET ...\n");
populate( terminal, nterm_nodes, node_id, nnodes, parent, child,
            split_value, split_var, Ltest, Vtest, Nt, N1test, N2test, NN );
// MISCLASSIFICATION RATE of each MINIMIZING SUBTREE indexed by
// SUBTREE SHRINKED NODE ID */
printf("FINDING THE RIGHT-SIZE TREE ...\n");
```

```
        if( (stream2 = fopen("subtrees.prn","r")) == NULL )
        {       printf("File not found\n");
                return fmin;
        }
        right_size( stream2, node_id, nnodes, N1test, N2test, MR, FA );
        fclose(stream2);
        fprintf(stream3,"the MISCLASSIFICATION RATE of each MINIMIZING SUBTREE
        indexed by SUBTREE SHRINKED NODE ID \n");
        for(i=0;i<nstrees;i++)
        {       j = subtrees[i];
                fprintf(stream3,"Subtree: %4d:  Misclass Rate: %10.4f:  False Alarms: %4d \n",
                j, MR[j], FA[j]);
        }
        fclose(stream3);
        fmin = MR[subtrees[0]];
        for(i=1;i<nstrees;i++)
        {       j = subtrees[i];
                if( MR[j] <= fmin )
                        fmin = MR[j];
        }
        printf("Minimum Misclassification of %6.4f\n",fmin);
        stream3 = fopen("results.prn","a");
        fprintf(stream3,"\n%f\n",fmin);
        fclose(stream3);
        return fmin;
}


// =================================================================
// TREE BUILDING ALGORITHM
/* TREE4AEP.C - Returns the number of nodes nnodes */
#include <stdio.h>
#include <stdlib.h>
int tree4aep(double **Lt, int *Vt, int M, int N, int maxn, int *N1,          int *N2,
                int *node_id, int *parent, int **child,  double *split_value, int *split_var,
                double *Imp_node,  double *change_Imp, int *termnode_tag,
                int **lset, int *nlset, double criterion )
{
        // At this point, the function tree4aep has accepted Lt and Vt
        int i,j, node_2_split=0, c0, c1, N12[4];
        int nnodes = 1; /* Initialize the number of nodes */
        int max_nodeid = 0, *learning_set, *lset_left, *lset_rght;
        double sv=0, CI=0, Imp_left=0, Imp_rght=0;

        // FUNCTION DECLARATIONS
        double phi( int n1, int n2);
        int whichnode(int *termnode_tag, int nterm, double *Imp_node, double criterion);
        int SPLIT_NODE( double **Lt, int *Vt, int N, int nattrib, int *learning_set,
                        double *sv, double *CI, double *Imp_left, double *Imp_rght,
                        nt *N12, int *lset_left, int *lset_rght);

        // To hold the learning set indexed by column number
```

```
learning_set = (int *)malloc(N * sizeof(int));
lset_left = (int *)malloc(N * sizeof(int));
lset_rght = (int *)malloc(N * sizeof(int));
for(i=0;i<maxn;i++)
{       for(j=0;j<N;j++)
        lset[i][j] = 0;
}
// By default, termnode_tag[] is initially set to [1] - all nodes are terminal nodes
for(j=0;j<maxn;j++)
        termnode_tag[j] = 1;

printf("A TREE-GROWING FUNCTION: \n");
printf("Given the inputs:\n");
printf("1.  Measurement Space of MxN dimensional Sample Data\n ");
printf("2.  N dimensional class vector\n");
printf("OUTPUTS:\n");
printf("1. Tree Structure\n ");
/* R.F. Nuqui 02-10-00
% Decision Trees
% For reference see:
% Book on "Neuro-Fuzzy and Soft Computing (chapter 14)
% Book on Classification and Regression Trees  */

/* Data Preprocessing */
/* % Learning set numbered sequentially */
for(i=0;i<N;i++)
        learning_set[i] = i;
/* Grow the tree */
node_id[0] = 0; /* <== This is the ROOT NODE */
parent[0] = -1; /* The ROOT node has no parent */
/* Segregate the voltage security classes at the root node */
N1[0] = 0; /* The number of voltage SECURE cases at root node */
N2[0] = 0; /* The number of voltage INSECURE cases at root node */
for(i=0;i<N;i++)
{       if( Vt[i] == 1 )
                N1[0] = N1[0] + 1;
        else
                N2[0] = N2[0] + 1;
}
Imp_node[0] = phi(N1[0],N2[0]);
for(j=0;j<N;j++)
        lset[0][j] = j; /* The learning set of root node [1] contains
                                the complete set of training data */
nlset[0] = N; /* This is the number of learning set at root node [1] */
termnode_tag[0] = 1; /* Initially, the terminal node is the root node */
while( node_2_split >= 0 )
{       node_2_split = whichnode(termnode_tag, nnodes, Imp_node, criterion);
        if( node_2_split == -1 )
        {       printf("Termination Criterion is Satisfied\n ");
                break;
        }
```

```c
                    termnode_tag[node_2_split] = 0; /* Not Anymore a Terminal Node */

                    printf("Now at Node %4d\n", node_2_split);
                    printf("Child Node of %4d\n", parent[node_2_split]);
                    for(j=0;j<nlset[node_2_split];j++)
                            learning_set[j] = lset[node_2_split][j];
                    split_var[node_2_split] = SPLIT_NODE( Lt, Vt, nlset[node_2_split], M,
                    learning_set, &sv, &CI, &Imp_left, &Imp_rght, N12, lset_left, lset_rght);
                    split_value[node_2_split] = sv;
                    change_Imp[node_2_split] = CI;
                    node_id[nnodes] = max_nodeid + 1;
                    child[node_2_split][0] = node_id[nnodes];
                    node_id[nnodes+1] = max_nodeid + 2;
                    child[node_2_split][1] = node_id[nnodes+1];
                    nnodes = nnodes + 2;
                    max_nodeid = nnodes-1;
                    c0 = child[node_2_split][0];
                    c1 = child[node_2_split][1];
                    N1[c0] = N12[0];
                    N2[c0] = N12[1];
                    N1[c1] = N12[2];
                    N2[c1] = N12[3];
                    parent[c0] = node_2_split;
                    parent[c1] = node_2_split;
                    nlset[c0] = N1[c0]+N2[c0];
                    nlset[c1] = N1[c1]+N2[c1];
                    for(i=0;i<nlset[c0]; i++)
                            lset[c0][i] = lset_left[i];
                    for(i=0;i<nlset[c1]; i++)
                            lset[c1][i] = lset_rght[i];
                    Imp_node[c0] = Imp_left;
                    Imp_node[c1] = Imp_rght;
            }
        return nnodes;
}


// ===============================================================
/* PHI.C - Computes the impurity of measurement space with
            parameters: N1- Class 0 (Insecure); N2- Class 1 (Secure) */
#include <math.h>
double phi( int n1, int n2)
{
        double x;
        double N1, N2, N;
        N1 = n1;
        N2 = n2;
        N = n1 + n2;
        N = N1+N2;
        x = -N1/N*log(N1/N)-N2/N*log(N2/N);
        return x;
}
```

```
// ================================================================
// SPLIT NODE.C: Function to determine
                         1. The optimal split variable <RETURN VALUE>
                          2. The optimal split value
# include <stdio.h>
# include <stdlib.h>
# include <math.h>
int SPLIT_NODE( double **L, int *V, int nlset, int nattrib, int *lset_node,
                         double *sv, double *CI, double *Imp_left, double *Imp_rght,
                              int *N12, int *lset_left, int *lset_rght)
{
        int i, j, k, l, ll, d1, d2, nsec=0, ninsec=0, nleft = 0, nrght = 0,
        int nsecL = 0, ninsecL = 0, nsecR = 0, ninsecR = 0, nsplit;
        int N1l=0, N2l=0, N1r=0, N2r=0, Nl, Nr, ls;
        double delEt, delEtmax, x, y, z, Etl, Etr, Impurity;
        int den, *ii, *iii;
        double *split, *osplit, *Lti, *ImpL, *ImpR, *imprty, sa;

        double phi(int nsec, int ninsec);
        void DsortI( double *A, int n, int *indx );
        split = (double *)malloc( (nlset-1) * sizeof(double) ); // To hold the split values
        osplit = (double *)malloc( nattrib * sizeof(double) );
        imprty = (double *)malloc( nattrib * sizeof(double) );
        ImpL = (double *)malloc( nattrib * sizeof(double) );
        ImpR = (double *)malloc( nattrib * sizeof(double) );
        ii = (int *)malloc( nlset * sizeof(int) );
        iii = (int *)malloc( nattrib * sizeof(int) );
        Lti = (double *)malloc( nlset * sizeof(double) );

        for(j=0;j<nlset;j++)
        {       i = lset_node[j];
                if(V[i]==1 )
                         nsec++;
                if(V[i] == 0 )
                         ninsec++;
        }
        /* The impurity of this data set is */
        Impurity = phi(nsec, ninsec);

        nsplit = nlset - 1; /* The # of split values is  */

        for(k=0;k<nattrib;k++)
        {       i = k;

                /*Generate lists of split values for attribute i */
                for(j=0;j<nlset;j++)
                         Lti[j] = L[i][lset_node[j]];
                DsortI(Lti,nlset,ii);

                //Take the average of 2 consecutive values
```

```
for(j=0;j<nsplit;j++)
{       d1 = ii[j];
        d2 = ii[j+1];
split[j] = ( Lti[d1] + Lti[d2] ) / 2;
}

delEtmax = 0;
for(j=0;j<nsplit;j++)
{       nleft = 0;
        nrght = 0;
        nsecL = 0;
        nsecR = 0;
        ninsecL = 0;
        ninsecR = 0;
        for(ll=0;ll<nlset;ll++)
        {       l = lset_node[ll];
                if( L[i][l] <= split[j] )
                {       nleft++ /* No of cases branching to left */
                        if( V[l] == 1 )
                                nsecL++; /* No. of cases on left as secure */
                        else
                            ninsecL++; /* No. of cases on left as insecure */
                }
                else
                {       nrght ++; /* No of cases branching to right */
                        if( V[l] == 1 )
                            nsecR++; /* No. of cases on right as secure */
                        else
                            ninsecR++; /* No. of cases on right as insecure */
                }
        }

        if( nleft > 0 )
        {       den = nleft;
                // Impurity of the left child node
                if( nsecL ==0 || ninsecL == 0 )
                        Etl = 0;
                else
                {       x = nsecL;
                        y = ninsecL;
                        z = x + y;
                        Etl = -x/z * log(x/z) - y/z * log(y/z);

                }
        }
        else
                Etl = 1;

        if( nrght > 0 )
        {       den = nrght;
                // Impurity of the right child node
```

```
                              if( nsecR==0 || ninsecR == 0 )
                                      Etr = 0;
                              else
                              {       x = nsecR;
                                      y = ninsecR;
                                      z = x + y;
                                      Etr = -x/z * log(x/z) - y/z * log(y/z);
                              }
                      }
                      else
                              Etr = 1;

                  z = nleft + nrght;
                  // Change in impurity due to split
                  delEt = Impurity - nleft/z * Etl - nrght/z * Etr;

                  if( delEt > delEtmax )
                  {       delEtmax = delEt;
                          osplit[i] = split[j];
                          imprty[i] = delEt;
                          ImpL[i] = Etl;
                          ImpR[i] = Etr;
                  }
          }

          // The optimal split value for this atribute 'i' is
          printf("Attribute Number: %4d  Delta Impurity: %f\n",i,imprty[i]);
}
DsortI( imprty, nattrib, iii );
k = iii[nattrib-1];
sa = k;
*sv = osplit[k];
*Imp_left = ImpL[k];
*Imp_rght = ImpR[k];
*CI = imprty[k];

for(i=0;i<nlset;i++)
{       lset_left[i] = 0;
        lset_rght[i] = 0;
}

Nl = -1;
Nr = -1;
for(i=0;i<nlset;i++)
{       ls = lset_node[i];
        if( L[sa][ls] <= *sv )
        {       Nl++;
                lset_left[Nl] = ls;
                if( V[ls] == 1 )
                        N1l = N1l + 1;
                else
```

```
                                          N2l = N2l + 1;
                    }
                    else
                    {        Nr++;
                             lset_rght[Nr] = ls;
                             if( V[ls] == 1 )
                                     N1r = N1r + 1;
                             else
                                     N2r = N2r + 1;
                    }
          }
          N12[0] = N1l;
          N12[1] = N2l;
          N12[2] = N1r;
          N12[3] = N2r;

          return sa;
}


// =============================================================
int whichnode(int *termnode_tag, int nnodes, double *Imp_node, double criterion)
{
          int I,choice = -1;

          for(i=0;i<nnodes;i++)
          {        if( termnode_tag[i] == 1 )
                   {
                             if( Imp_node[i] >= criterion )
                                      choice = i;
                   }
          }
          return choice;
}


// =============================================================
// POPULATE.C - Populate the Nodes of a Decision Tree Given the test data set

#include <stdio.h>
#include <stdlib.h>

void populate( int *terminal, int nterm_nodes, int *node_id, int nnodes,
                        int *parent, int **child, double *split_value, int *split_var,
                        double **L, int *Vclass, int ntset, int *N1, int *N2, int *N )
{
          int i,ii,j, cn, in, lj, *tag, **test_set;
          int ismember( int *A, int a, int x );
          int setdiff( int *A, int a, int *B, int b );
          test_set = (int **)malloc(nnodes * sizeof(int *) );
          if( test_set == NULL) printf("Memory Allocation Problem for test_set\n");
          for(i=0;i<nnodes;i++)
          {        test_set[i] = (int *)malloc(ntset * sizeof(int));
```

```
            if( test_set == NULL)
                    printf("Memory Allocation Problem for test_set\n");
    }
    for(ii=0;ii<nnodes;ii++)
    {       i = node_id[ii];
            for(j=0;j<ntset;j++)
                    test_set[i][j] = 0;
    }
    tag = (int *)malloc(nnodes * sizeof(int) );
    for(i=0;i<nnodes;i++)
            tag[node_id[i]] = 0;
    for(i=0;i<nnodes;i++)
    {
            N[node_id[i]] = 0;
            N1[node_id[i]] = 0;
            N2[node_id[i]] = 0;
    }
    cn = node_id[0];
    for(i=0;i<ntset;i++)
            test_set[cn][i] = i;

    N[cn] = ntset;
    N1[cn] = 0;
    for(i=0;i<ntset;i++)
    {       if( Vclass[i] == 1 )
            N1[cn]++;
    }
    N2[cn] = N[cn] - N1[cn];
    while(cn != parent[node_id[0]] )
    {       if( tag[child[cn][0]] == 1 && tag[child[cn][1]] == 1 )
            {       tag[cn] = 1;
                    cn = parent[cn];
            }
            else
            {       for(i=0;i<2;i++)
                    {       in = child[cn][i];
                            if( tag[in] == 0 )
                            {       if( i == 0 )
                                    {       for(j=0;j<N[cn];j++)
                                            {       lj = test_set[cn][j];
                                                    if( L[split_var[cn]][lj] <=
                                                            split_value[cn] )
                                                    {       N[in]++;
                                                            test_set[in][N[in]] = lj;
                                                            if( Vclass[lj] == 1 )
                                                                    N1[in]++;
                                                            else
                                                                    N2[in]++;
                                                    }
                                            }
                                    }
```

```
                                  else
                                  {       for(j=0;j<N[cn];j++)
                                          {       lj = test_set[cn][j];
                                                  if( L[split_var[cn]][lj] >
                                                          split_value[cn] )
                                                  {       N[in]++;
                                                          test_set[in][N[in]] = lj;
                                                          if( Vclass[lj] == 1 )
                                                                  N1[in]++;
                                                          else
                                                                  N2[in]++;
                                                  }
                                          }
                                  }
                                  if( ismember(terminal,nterm_nodes,in) == 1 )
                                          tag[in] = 1;
                                  else
                                  {       cn= in;
                                          break;
                                  }
                          }
                  }
          }
  }

// ==============================================================
#include <stdio.h>
#include <stdlib.h>

// PRUNE.C - To FIND: 1. The set of minimizing SUBTREES [subtrees]
                              subtrees[] contains the set of SHRINKED nodes'
                  2. The misclassification cost of each subtree indexed by the shrinked nodes'
int prune( FILE *stream, int *N1, int *N2, int **child, int *split_var,
                double *split_value, float *U, int **set_term, int *nterm,
                int *node_id, int nnodes, int *parent, int **desc, int *nd,
                int *termnode_tag, int *subtrees, float *UT )
{
          int i, j, ii, jj, s, n = -1, nint_nodes, *int_nodes, *nodes, *terminal;
          float alpmin, *alp;
          int nstrees = 0,nn, *term_tag, nterm_nodes, *nt, **set_t;

          void alpha( int *int_nodes, int nint_nodes, float *U,
                          int **set_term, int *nterm, float *alp);
          int setdiff( int *A, int a, int *B, int b );
          void set_termnodes( int *termnode_tag, int *node_id, int nnodes, int *parent,
                                  int **set_term, int *nterm );
          int ismember( int *A, int a, int x );
          void store( FILE *stream, int s, float F, int *N1, int *N2,
                          int *subtreenodes, int nnodes, int *sterminal,
                          int nsterm, int **child, int *split_var, double *split_value);
          nn = nnodes;
```

```
printf("FIND:1. The set of minimizing SUBTREES [subtrees]\n");
printf(" subtrees[] contains the set of SHRINKED nodes\n");
printf(" 2. The misclassification cost of each subtree indexed by the shrinked nodes\n");

// Allocate memory to nodes[] - proxy for node_id[]
nodes = (int *)malloc(nnodes*sizeof(int));
for(i=0;i<nnodes;i++)
        nodes[i] = node_id[i];

/* Allocate memory to terminal[] */
terminal = (int *)malloc(nnodes*sizeof(int));

/* Allocate memory to term_tag[] - proxy to termnode_tag[] */
term_tag = (int *)malloc(nnodes * sizeof(int));

/* Allocate memory to nt[] - proxy for nterm[] */
nt = (int *)malloc(nnodes * sizeof(int));
for(i=0;i<nnodes;i++)
        nt[i] = nterm[i];

/* Allocate memory to set_t[][] - proxy for set_term[][] */
set_t = (int **)malloc(nnodes * sizeof(int *));
if( set_t == NULL) printf("Memory Allocation Problem for set_t\n");
for(i=0;i<nnodes;i++)
{
        set_t[i] = (int *)malloc(2 * sizeof(int));
        if( set_t == NULL) printf("Memory Allocation Problem for set_t\n");
}
for(i=0;i<nnodes;i++)
{
        for(j=0;j<nterm[i];j++)
                set_t[i][j] = set_term[i][j];
}
nint_nodes = 0;
nterm_nodes = 0;
for(ii=0;ii<nnodes;ii++)
{
        i = node_id[ii];
        if( termnode_tag[i] == 0 )
                nint_nodes++;
        else
        {
                nterm_nodes++;
                terminal[nterm_nodes-1] = i;
        }
}
/* Allocate memory to int_nodes[] */
int_nodes = (int *)malloc(nint_nodes * sizeof(int));
nint_nodes = 0;
for(ii=0;ii<nnodes;ii++)
```

```
{
        i = nodes[ii];
        if( termnode_tag[i] == 0 )
        {
                nint_nodes++;
                int_nodes[nint_nodes-1] = i;
        }
}

/* Allocate memory to alp[] */
alp = (float *)malloc(nint_nodes * sizeof(float));
for(i=0;i<nint_nodes;i++)
        alp[i] = 9999;

printf("STORING TREE INFORMATION .....\n");
fprintf(stream,"subtree ID, T\n");
fprintf(stream,"Misclassification Cost R(T)\n");
fprintf(stream,"Node Child1 Child2 Attrib Split Vc_train\n");
fprintf(stream,"0 - Terminator\n");

s = 1000;
while( s != nodes[0] )
{
        alpha( int_nodes, nint_nodes, U, set_term, nterm, alp);
        alpmin = alp[0];
        ii = 0;
        for(j=1;j<nint_nodes;j++)
        {
                i = int_nodes[j];
                if( alp[i] < alpmin )
                {
                        ii = j;
                        alpmin = alp[i];
                }
        }
        n++;
        subtrees[n] = int_nodes[ii];
        s = subtrees[n];
        nstrees++;

        /* Prune the tree at s */
        nterm_nodes = setdiff( terminal, nterm_nodes, set_t[s], nt[s] );

        /* Include node s in the set of terminal nodes */
        nterm_nodes++;
        terminal[nterm_nodes-1] = s;

        /* Calculate the cost of this subtree */
        UT[s] = 0;
        for(jj=0;jj<nterm_nodes;jj++)
        {
```

```
                    i = terminal[jj];
                    UT[s] = UT[s] + U[i];
            }

            /* Update the list of nodes */
            nn = setdiff(nodes, nn, desc[s], nd[s] );
            for(i=0;i<nn;i++)
            {
                    ii = nodes[i];
                    if( ismember( terminal, nterm_nodes, ii ) == 1 )
                            term_tag[ii] = 1;
                    else
                            term_tag[ii] = 0;
            }

            /* Store tree information in file */
            store( stream, s, UT[s], N1, N2, nodes, nn, terminal, nterm_nodes,
                        child, split_var, split_value);

            /* Rebuild the internal nodes set */
            for(i=0;i<nn;i++)
            {
                    int_nodes[i] = nodes[i];
            }
            nint_nodes = setdiff( int_nodes, nn, terminal, nterm_nodes );

            /* Update set_t */
            set_termnodes( term_tag, nodes, nn, parent, set_t, nt );

    }
    return nstrees;
}

// ==============================================================
#include <stdio.h>
#include <stdlib.h>

/* Function to find the RIGHT-SIZE TREE given the SUBTREES file */
void right_size( FILE *stream, int *node_id, int nnodes,
                                int *N1, int *N2, float *MR, int *FA )
{
    int i,j, ii, NN, subtree=0, node=0, c1=0, c2=0, attrib=0, Vclnode =0;

    float split_value=0;
    char line[80];
    for(i=0;i<nnodes;i++)
    {
            ii = node_id[i];
            MR[ii] = 0;
            FA[ii] = 0;
    }
```

```
        NN = N1[0] + N2[0];
        for(i=0;i<4;i++)
                fgets(line,80,stream);
        j = 0;
        node = -1;
        while( fgets(line,80,stream) != NULL )
        {
                if( node == -1 )
                {
                        sscanf(line,"%d",&subtree); /* The subtree ID record */
                        j++;
                        fgets(line,80,stream); /* Skip the cost record of this tree */
                        fgets(line,80,stream);
                }
                sscanf(line,"%d",&node);
                if( node != -1 )
                {       scanf(line,"%d %d %d %d %f %d", &node, &c1, &c2, &attrib,
                                &split_value, &Vclnode);
                        if( c1 == 0 && c2 == 0 )
                        {       if(     clnode == 1 )
                                        R[subtree] = MR[subtree] + N2[node];
                                else
                                {
                                        MR[subtree] = MR[subtree] + N1[node];
                                        FA[subtree] = FA[subtree] + N1[node];
                                }
                        }
                }
                else
                        MR[subtree] = MR[subtree] / NN;
        }
}

// ==============================================================
// SET_TERMNODES.C - To determine the set and number of terminal nodes attached
                                to each internal node:
                                1. # of terminal nodes nterm[node]
                                2. the set of terminal nodes set_term[node][nterm[node]]
#include <stdio.h>
void set_termnodes( int *termnode_tag, int *node_id, int nnodes, int *parent,
                                int **set_term, int *nterm )
{
        int i, ii, m;
        for(ii=0;ii<nnodes;ii++)
        {       i = node_id[ii];
                nterm[i] = 0;
                set_term[i][0] = 0;
        }

        for(ii=0;ii<nnodes;ii++)
        {       i = node_id[ii];
```

```
                    if( termnode_tag[i] == 1 )  // Loop for all terminal nodes
                    {       set_term[0][nterm[0]-1] = i;
                            m = parent[i];
                            while( m >= 0 )
                            {       nterm[m] = nterm[m]++;
                                    set_term[m][nterm[m]-1] = i;
                                    m = parent[m];
                            }
                    }
            }
}


// ================================================================
// DESCENDANT.C - To determine the set of nodes descendant to all
                                internal nodes
void descendant(int *termnode_tag, int *node_id, int nnodes, int *parent,
                                int **child, int **desc, int *nd)
{
        int i, j, root_node=0;
        int *tag, cn, in, ij;

        tag = (int *)malloc(nnodes * sizeof(int));
        nd[root_node] = nnodes - 1;
        for(j=0;j<nd[root_node];j++)
                desc[root_node][j] = j + 1;


        for(j=0;j<nnodes;j++)
        {       if( termnode_tag[j] == 1 )
                {       nd[j] = 0;
                        desc[j][0] = 0;
                        desc[j][1] = 0;
                }
        }
        for(j=0;j<nnodes;j++)
        {       ij = node_id[j];
                cn = ij;
                if( ij != root_node && termnode_tag[ij] == 0 )
                {       for(i=0;i<nnodes;i++)
                                tag[i] = 0;
                        nd[ij] = 0;
                        while( cn != parent[ij] )
                        {       if( tag[child[cn][0]] == 1 && tag[child[cn][1]] == 1 )
                                {       tag[cn] = 1;
                                        if( cn != ij )
                                        {       nd[ij]++;
                                                desc[ij][nd[ij]-1] = cn;
                                        }
                                        cn = parent[cn];
                                }
                                else
                                {       for(i=0;i<2;i++)
```

```
                                        {            in = child[cn][i];
                                                     if( tag[in] == 0 )
                                                     {       if( termnode_tag[in] == 1 )
                                                             {       tag[in] = 1;
                                                                     nd[ij]++;
                                                             desc[ij][nd[ij]-1] = in;
                                                             }
                                                             else
                                                             {       cn = in;
                                                                     break;
                                                             }
                                                     }
                                             }
                                     }
                             }
                     }
             }
}


// ================================================================
// ALPHA.C - To find the ALPHAs of each internal nodes;
#include <stdio.h>
void alpha( int *internal_nodes, int nint_nodes, float *U,
                    int **set_term, int *nterm, float *alp)
{
        int i, j, in, nt, jt;
        float tsum;

        for(i=0;i<nint_nodes;i++)
        {       in = internal_nodes[i];
                tsum = 0;
                nt = 0;
                for(j=0;j<nterm[in];j++)
                {       jt = set_term[in][j];
                        if( jt != 0 )
                        {       tsum = tsum + U[jt];
                                nt++;
                        }
                }
                alp[in] = ( U[in] - tsum ) / (nt - 1);
        }
}


// ================================================================
// DSORTI.C: To determine the index of sorted double-type elements
#include <stdlib.h>
void DsortI( double *A, int n, int *indx )
{
        int i,j, k, minI;
        double min, *a;
```

```c
        a = (double *)malloc(n * sizeof(double) );
        for(i=0;i<n;i++)
        {       indx[i] = i;
                a[i] = A[i];
        }

        for(i=0;i<n-1;i++)
        {       k = i;
                min = a[k];
                minI = indx[k];
                for(j=i+1;j<n;j++)
                {       if( a[j] < min )
                        {       k = j;
                                min = a[k];
                                // Exchange a[k] with a[i]
                                a[k] = a[i];
                                a[i] = min;

                                minI = indx[k];
                                // Exchange a[k] with a[i]
                                indx[k] = indx[i];
                                indx[i] = minI;
                        }
                }
        }
}

// ============================================================
#include <stdio.h>
int ismember( int *A, int a, int x )
{
        int i,ans=0;

        for(i=0;i<a;i++)
        {       if( A[i] == x )
                {       ans = 1;
                        break;
                }
        }
        return ans;
}

// ============================================================
#include <stdio.h>
#include <stdlib.h>

int setdiff( int *A, int a, int *B, int b )
{
        int i, n=0, *Adummy;
        int ismember( int *A, int a, int x );
```

```
Adummy = (int *)malloc(a * sizeof(int) );
for(i=0;i<a;i++)
        Adummy[i] = A[i];


for(i=0;i<a;i++)
{       if( ismember( B, b, Adummy[i] ) == 0 )
        {       n++;
                A[n-1] = Adummy[i];
        }
}
return n;
}
```

VITA
Reynaldo Francisco Nuqui

Personal

Place of Birth: San Simon, Pampanga, Philippines

Education

Ph.D. in Electrical Engineering, July 15, 2001
   Virginia Polytechnic Institute and State University, Blacksburg, VA
   Course Work: Power System Operation and Control, Power System Dynamics
      and Control, Power System Protection (lecture and laboratory),
      Microcomputer Applications in Power Systems, Alternate Energy
      Systems, Multivariable Control, Linear Systems Theory, Neural
      and Fuzzy Systems, Numerical Analysis, Power System Security
      in a Deregulated Environment.

M.S. in Electrical Engineering, May 7, 1987
   University of the Philippines, Quezon City, Philippines
   Thesis Title: Optimal Generator Maintenance Outage Scheduling Using Integer
      Programming
   Course Work: Power Systems, Reliability, Fault Analysis, Transient Stability,
      Optimization

B.S. in Electrical Engineering, November 15, 1981
   University of the Philippines, Quezon City, Philippines

Professional Experience

August 15, 2000-June 30, 2001
   R&D Consultant, ABB Power T&D Company, Raleigh, NC
   Areas of Work: phasor measurement units, voltage stability analysis, phasor
      measurement unit placement, high impedance fault detection,
      portfolio optimization.

August 16, 1998- August 15, 2000
   Graduate Research Assistant, Virginia Tech, Blacksburg, VA
   Areas of Work: phasor measurement unit placement for incomplete observability,
      state estimation with phasor measurements, voltage stability
      monitoring with phasor measurements.

August 1, 1983- August 1, 1996
   Principal Engineer, National Power Corporation, Philippines
   Areas of Work: dynamic stability analysis and simulation, security analysis, event
      reconstruction, scheduling and dispatch, high voltage direct
      current transmission, fault analysis, system studies, load
      forecasting, production costing, and tariffs.

Professional Experience (continued)
1991-1993
>   Senior Lecturer, Manila Electric Company, Philippines
>   Teaching Area: Power System Reliability Evaluation

June 7, 1987 – March 1, 1993
>   Assistant Professor, Mapua Institute of Technology, Philippines
>   Teaching Areas: energy systems, control systems

Projects undertaken
1999-2001: PMU Initiative at the Tennessee Valley Authority

1994-1995: Development of a Philippine Grid Code (with NGC, UK representatives).

1993-1994: Contracting and Economic Dispatching of Independent Power Producers
>   (with Power Technologies Inc., USA)

1991-1993: Leyte-Luzon HVDC Interconnection (with SWEDPOWER consultants)

Research Interests
>   Phasor measurement units, state estimation, voltage stability, and dynamic
>   stability

Professional Societies
IEEE Student Member, 1999
>   Power Engineering Society
>   Computer Society

Publications
1. R. F. Nuqui, A.G. Phadke, R.P. Schulz and N Bhatt "*Fast On-Line Voltage Security Monitoring Using Synchronized Phasor Measurements and Decision Trees*", presented at the IEEE Winter Power Meeting 2001, Columbus, Ohio.
2. R.R. Austria, R. Nadira, W.R. Puntel , P.S. Hurley , R.C. Bacani, R.F. Nuqui, M.R. Pangilinan, "*Wheeling Tariffs for Transmission Services*", Presented at the 11th Conference of the Electric Power Supply Industry (11 CEPSI), Kuala Lumpur, Malaysia, October 21-25, 1996.
3. R. F. Nuqui, Discussion of "*Current Issues in Operational Planning*", IEEE Transactions on Power Systems, Vol. 7, No.3, August 1992.
4. R. F. Nuqui, C. C. Claudio, Discussion of "*Operational Aspects of Generation Cycling*", IEEE Transactions on Power Systems, August 1991.
5. R. F. Nuqui, "*Impact of Fast-Track Generation Projects on the Operational Security of the Luzon Grid Power System*", presented at the IIEE Annual Meeting, June 1991, Manila, Philippines.