OPTIMAL MONITORING AND VISUALIZATION OF STEADY STATE POWER SYSTEM OPERATION

A Dissertation

by

BEI XU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2006

Major Subject: Electrical Engineering

OPTIMAL MONITORING AND VISUALIZATION OF STEADY STATE POWER SYSTEM OPERATION

A Dissertation

by

BEI XU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Ali Abur
Committee Members,	Mladen Kezunovic
	Anirudda Datta
	Ergun Akleman
Head of Department,	Costas N. Georghiades

August 2006

Major Subject: Electrical Engineering

ABSTRACT

Optimal Monitoring and Visualization of

Steady State Power System Operation. (August 2006)

Bei Xu, B.S.; M.S. Shanghai Jiaotong University

Chair of Advisory Committee: Dr. Ali Abur

Power system operation requires accurate monitoring of electrical quantities and a reliable database of the power system. As the power system operation becomes more competitive, the secure operation becomes highly important and the role of state estimation becomes more critical. Recently, due to the development of new technology in high power electronics, new control and monitoring devices are becoming more popular in power systems. It is therefore necessary to investigate their models and integrate them into the existing state estimation applications.

This dissertation is dedicated to exploiting the newly appeared controlling and monitoring devices, such as Flexible AC Transmission System (FACTS) devices and (Phasor Measurement Units) PMUs, and developing new algorithms to include them into power system analysis applications. Another goal is to develop a 3D visualization tool to help power system operators gain an in-depth image of the system operation state and to identify limit violations in a quick and intuitive manner.

An algorithm of state estimation of a power system with embedded FACTS devices is developed first. This estimator can be used to estimate the system state quantities and Unified Power Flow Controller (UPFC) controller parameters. Furthermore, it can also to be used to determine the required controller setting to maintain a desired power flow through a given line. In the second part of this dissertation, two methods to determine the optimal locations of PMUs are derived. One is numerical and the other one is topological. The numerical method is more effective when there are very few existing measurements while the topology-based method is more applicable for a system, which has lots of measurements forming several observable islands. To guard against unexpected failures of PMUs, the numerical method is extended to account for single PMU loss. In the last part of this dissertation, a 3D graphic user interface for power system analysis is developed. It supports two basic application functions, power flow analysis and state estimation. Different visualization techniques are used to represent different kinds of system information.

To My Parents: Guihai Xu, Yuhua Pang and My Husband: Yunli Xiong

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Dr. Ali Abur, for his guidance, patience and support throughout my Ph.D. studies. His encouragement and technical comments are essential to the completion of this dissertation.

I would also like to express my special thanks to Dr. Ergun Alkeman for his help and invaluable advice in developing the 3D visualization interface. I am also very grateful to my committee members, Dr. M. Kezunovic and Dr. A. Datta, for their help and academic advice during my graduate studies.

Thanks to Power System Engineering Research Center (PSERC), the National Science Foundation (NSF) and the Department of Electrical and Computer Engineering Department of Texas A&M University for their financial support.

I would like to thank my parents and my husband for their generous love and support. I also want to thank my dear friends Dr. Cansın Y. Evrenosoğlu for his generous help in my research and Dr. Aleksander K. Wójcik for his proof reading of the manuscript.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

BACKGROUND

A. Background

Power system state estimation has been playing an active and critical role in electric power system since 1969 when it was first introduced by Fred Schweppe [1]. By collecting analog measurements data and the status data of the circuit breakers through the Supervisory Control and Data Acquisition System (SCADA) and feeding them into state estimation function, state estimation can provide an estimate for all measured and unmeasured electrical quantities and network parameters of the entire power system, detect and filter out gross errors in the measurement set and detect the topology errors in the network configuration. It is one of the essential functions in Energy Management Systems (EMS) to provide an accurate monitoring of the operation quantities of power systems and a reliable database of the system upon which subsequent network functions such as optimal power flow, security analysis, etc. could be deployed.

As power system deregulation becomes widespread around the globe, power markets become competitive. As a result, to maintain the secure operation of power system becomes highly important and more difficult and the role of state estimation is becoming more critical. Meanwhile, new control and monitoring devices such as Flexible AC Transmission System (FACTS) and Phasor Measurement Units (PMU) are becoming more and more popular in the deregulated power system due to the development of new technology in high power electronics. Thus, to investigate the detailed mathematical models of these devices and integrate them into the existing

The journal model is *IEEE Transactions on Power Systems.*

state estimation applications will become necessary.

The concept of Flexible AC Transmission Systems was first introduced by Hingorani, N.G. in 1988 [2]. Since then, many types of FACTS devices have been developed. FACTS devices are able to change, in a fast and effective way, the network parameters to achieve a better system performance. They use circuit reactance, voltage magnitude and phase angle as control variables to provide flexible loading of lines by shifting the flows from heavy loaded lines to lighter loaded lines therefore increase the security and reliability of the power systems and enhance the Available Transfer Capability (ATC) of the power network without the construction of new transmission lines.

Synchronized Phasor Measurement Unit is a device, which uses synchronized signals from the global positioning system (GPS) satellites and is placed at selected substations to monitor the phasors of bus voltage as well as all the currents incident to the bus. PMU was first introduced in mid-1980s. It was first utilized in a substation for monitoring and analyzing power system dynamics. As they become available in large numbers, they can provide valuable information for EMS applications as well. PMUs are by far more accurate than the conventional measurements. By placing PMUs on certain strategic buses, a complete monitoring of the entire system can be accomplished, allowing the instant calculation of system states. Thus a faster and more reliable state estimation can be achieved.

In large scale power systems, system operators are overwhelmed by vast amount of data from EMS, which is usually displayed as numerical values on a one-line diagram. Recent advances in computer software and hardware makes it possible to develop highly detailed representation of power system in Graphical User Interface (GUI) for visualizing information about power system operation and deciding on the necessary corrective actions. The objective of power system visualization is to aid system operators to gain a better understanding of the system state at a glance.

B. Contributions of This Dissertation

The objectives of the research work are to exploit recently introduced control and monitoring devices and to develop a 3D visualization tool for improved monitoring of the power system operation. The contributions of this dissertation are briefly outlined below.

1. State estimation of system with FACTS devices

The detailed steady state model of Unified Power Flow Controller (UPFC) is used and integrated into the state estimation algorithm. The Interior Point (IP) method is used to solve the nonlinear optimization problem. The developed state estimation algorithm can work as a sate estimator as well as a tracing tool for the parameters of UPFC devices. Furthermore, it can detect multiple conforming bad data by using the inequality constraints of UPFC.

2. Optimal PMU placement

An integer programming based formulation and a topology-based algorithm for optimal PMU placement are developed. The integer programming based algorithm is a numerical method and allows systematic analysis of network observability for mixed measurement sets, which include conventional flow and injection measurements as well as PMUs. The other one is a topology analysis method, which makes the system observable by merging existing observable islands. Moreover, the numerical approach is extended to account for contingencies involving loss of single PMUs.

3. Visualization of power system

A graphical user interface for power system visualization is developed. This interface supports two basic application functions, power flow analysis and state estimation. Different kinds of visualization methods are used to provide images of system with easily identifiable characteristics related to violations of various operating limits. Convenient features, such as zooming in/out, panning and rotating are provided to aid the user for getting a better view of the power system.

C. Organization of Chapters

This dissertation is organized as below.

- Chapter II presents the formulation of state estimation of power system embedded with UPFCs.
- Chapter III describes the two algorithms for optimal PMU placements, a numerical method and a topology-based method.
- Chapter IV presents a user-friendly visualization tool, which supports two basic power system applications, power system analysis and state estimation.
- Chapter V is the summary of the research work in this dissertation.

CHAPTER II

STATE ESTIMATION OF POWER SYSTEM WITH FACTS DEVICES *

After the establishment of power markets with transmission open access, the significance and use of Flexible A.C. transmission systems (FACTS) devices for manipulating line power flows to relieve congestion and optimize the overall grid operation have increased. As a result, there is a need to integrate the FACTS device models into the existing power system applications. This chapter will present an algorithm for state estimation of network embedded with FACTS devices. Furthermore, it will be shown via case studies that the same estimation program can also be used to determine the controller setting for a desired operation condition.

A. Introduction

FACTS devices enable secure operation of power systems which have to be otherwise upgraded in order to relieve load on congested transmission lines, to increase the Available Transmission Capacity (ATC) of the power system and to optimize the system resources. As these devices are becoming more and more popular in the transmission systems, monitoring of the system state will require detailed models of these devices and their integration into the existing power system applications. One of these applications with a critical role in system monitoring is the state estimator.

There are several kinds of FACTS devices. Thyristor-switched series capacitors (TCSC) and thyristor switched phase shifting transformer (TCPST) can exert a voltage in series with the line and therefore can control the active power through a

* Reprinted with permission from "State estimation of systems with UPFCs using the interior point method" by Bei Xu and Ali Abur, 2004. *IEEE Transactions on Power Systems*, Vol. 19, 1635-1641. Copyright 2004 IEEE

transmission line [3]. On the other hand, the Unified Power Flow Controller (UPFC) has a series voltage source and a shunt voltage source, allowing independent control of the voltage magnitude, the real and reactive power flows along a given transmission line [4, 5]. In this chapter, only one device namely the UPFC is considered due to its complexity and versatility in controlling the power flows, even though the presented formulation is quite generic and can be applied to any type of FACTS device.

State estimation in power system can be formulated as a nonlinear weighted least square (WLS) problem. The UPFC model introduces operational and parameter constraints for the components. Thus, the state estimation problem will have to be formulated as an optimization problem with such constraints. It has a set of measurement equations: $z = h(x) + \varepsilon$; a set of equality constraints $g(x) = 0$, representing the zero injections of buses and the zero active power exchange between the power system and FACTS devices; a set of inequality constraints $f(x) \leq 0$, representing the Var limits on generators, transformer tap ratio limits and the power and voltage limit of FACTS devices. This problem is then solved by using the well documented and tested Interior Point (IP) method [6].

IP method has already been applied to the solution of a variety of power system problems [3, 7, 8, 9, 10]. In [9], it is used to solve the optimal reactive dispatch problem. IP method is used in optimal power flow formulation together with the multiple centrality corrections (MCC) technique in [8]. The problem of constrained LAV state estimation is formulated and solved via the IP method by using scaled penalty parameters [4]. Another application of IP method to the solution of constrained power system state estimation problem is given in [3] and [7]. In this formulation, Hachtel's method [11] is used due to its desirable numerical robustness. This chapter provides a more comprehensive description of the method including the discussion of the computational issues concerning the choice of the step length and use of scaling for improved matrix conditioning for large systems. By solving the problem we can not only estimate the state variables (bus voltages and phase angles) of power system but can also determine the controller settings of FACTS devices for a desired operating condition. Some practical solutions are proposed and simulation results for larger size IEEE test systems are provided to illustrate the performance of the proposed implementation.

This chapter is organized as below.

- Section A is introduction part.
- Section B introduces the steady-state model of the UPFC [5] with operating and parameters limits.
- Section C describes the method used in the proposed state estimator. First the commonly used Hatchtel's augmented matrix method [11] is used to implement a numerically robust and computationally efficient state estimator. Logarithmic barrier function method [7] is used and integrated into Hatchtel's matrix to treat the inequality constraints. Then the issues of observability analysis and bad data detections are discussed. The detailed equations for the measurements used in the algorithm are given in the section.
- Section D discusses about the computational issues such as the choice of initial points, adjusting step length and scaling the matrix to improve its conditioning.
- Simulation results for typical systems are shown in section E.
- The last section is the conclusion part.

B. Steady State Model of UPFC

The Unified Power Flow Controller (UPFC) can control the voltage magnitude, real and reactive power flows simultaneously. It is assumed that the system is operating

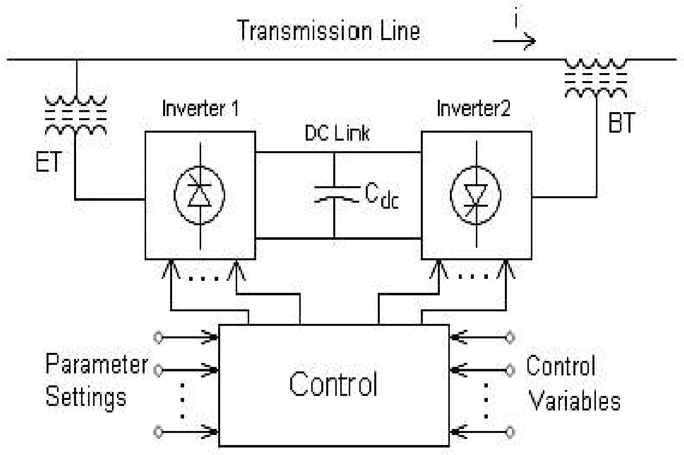


Fig. 1. Basic circuit arrangement of UPFC

under normal conditions and therefore only the steady state model of the UPFC is of interest. This model has been developed in [5]. The real physical model of UPFC consists of an excitation transformer, a boosting transformer and two switching converters as illustrated in Figure 1. The two inverters are operated from a common dc link provided by a dc storage capacitor. This arrangement functions as an ideal AC to AC power converter in which the real power can freely flow in either direction between the AC terminals of the two inverters and each inverter can independently generate (or absorb) reactive power at its own AC output terminal. The losses associated with the UPFC operation are typically neglected and under this assumption the UPFC will neither inject nor absorb any real power from the system while operating in the steady-state.

The steady state model can be built by using a voltage source and its source impedance inserted in series with the line and another voltage source and its source impedance connected in shunt at the bus where the excitation transformer is. The circuit model corresponding to this representation is shown in Figure 2.

The complex source voltage values are designated by $V_B\angle\theta_B$ and $V_E\angle\theta_E$ for the

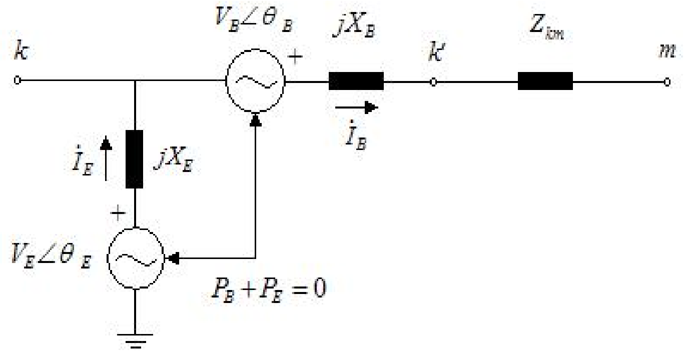


Fig. 2. Steady state model of UPFC

series and shunt sources respectively. Note that the given equality constraint $P_B + P_E = 0$ in Figure 2 implies that no real power is exchanged between the UPFC and the system. Here, P_B and P_E are the real power outputs of the two voltage sources. The operations of the two voltage sources are therefore mutually dependent. The parameters X_B, X_E and Z_{km} represent the source reactances for the series and shunt voltage sources and the transmission line impedance respectively.

The complex power flows that go through line k-m and m-k can be expressed as:

$$S_{km} = \hat{V}_k \cdot \hat{I}_{km}^* \tag{2.1}$$

$$S_{mk} = \hat{V}_m \cdot \hat{I}_{mk} \tag{2.2}$$

where,

$$\hat{I}_{km} = -\hat{I}_{mk} = \frac{\hat{V}_k + \hat{V}_B - \hat{V}_m}{Z_{km} + jX_B} \tag{2.3}$$

$\hat{I}_{km}$ is defined as the line current from bus k to bus m, while $\hat{I}_{mk}$ is the line current from bus m to bus k. $\hat{V}_k$ and $\hat{V}_m$ are the phasor voltages at bus k and bus m. they are defined as $V_k\angle\theta_k$ and $V_m\angle\theta_m$ respectively, where V_E, θ_E V_B, θ_B are the control variables of UPFC.

The complex power outputs of the series and shunt voltage sources can be expressed as:

$$S_E = \hat{V}_E \cdot \hat{I}_E \tag{2.4}$$

$$S_B = \hat{V}_B \cdot \hat{I}_B \tag{2.5}$$

where,

$$\hat{I}_B = \hat{I}_{km} = \frac{\hat{V}_k + \hat{V}_B - \hat{V}_m}{Z_{km} + jX_B} \tag{2.6}$$

$$\hat{I}_E = \frac{\hat{V}_E - \hat{V}_k}{jX_E} \tag{2.7}$$

There are equality and inequality constraints of UPFC, which can be formulated by Equations (2.8) to (2.12).

$$Real\ Power\ Constraints: \quad P_E + P_B = 0 \tag{2.8}$$

$$Shunt\ Power\ Constraints: \quad |S_E| = \sqrt{P_E^2 + Q_E^2} \leq S_{E,max} \tag{2.9}$$

$$Series\ Power\ Constraints: \quad |S_B| = \sqrt{P_B^2 + Q_B^2} \leq S_{B,max} \tag{2.10}$$

$$Shunt\ Voltage\ Constraint: \quad |V_E| \leq V_{E,max} \tag{2.11}$$

$$Series\ Voltage\ Constraint: \quad |V_B| \leq V_{B,max} \tag{2.12}$$

where, $S_{E,max}$ and $S_{B,max}$ are the power limits for the shunt and series voltage sources, and $V_{E,max}$ and $V_{B,max}$ are the limits on their voltage magnitudes.

C. Developed Algorithm

1. Hachtel's augmented matrix method

Power system state estimation problem can be formulated as a nonlinear least squares problem with a set of equality and inequality constraints [7] as shown in Equation

(2.13)

$$\begin{array}{lrcl} Minimize & \multicolumn{3}{l}{\frac{1}{2}r^T R^{-1} r} \\ & g(x) & = & 0 \\ Subject\ to & r - z + h(x) & = & 0 \\ & f(x) & \leq & 0 \end{array} \tag{2.13}$$

$z = h(x) + r$ represents the equations for measurements, where z is the $(m \times 1)$ measurement vector; $h(\cdot)$ is the $(m \times 1)$ vector of nonlinear functions, which relates measurements to state; x is the $(n \times 1)$ state vector; r is the $(m \times 1)$ measurement error vector.

$g(x) = 0$ represents the equality constraints, where $g(\cdot)$ is the $(r \times 1)$ vector of nonlinear functions. These equality constraints represent the zero injection buses and the zero active power exchange between the system and FACTS devices given by Equation (2.8).

$f(x) \leq 0$ represents the inequality constraints, where $f(\cdot)$ is the $(p \times 1)$ vector of nonlinear functions. These constraints represent the Var limits on generators, ratio limits of transformer tap and the power and voltage limits of the UPFC given by Equations (2.9) to (2.12)

The interior point method [6, 3, 7] can then be used to solve the nonlinear optimization problem. A non-negative slack variable vector $s \geq 0$ is added in order to convert the inequality constraints to equality constraints and the objective function is augmented by a logarithmic penalty function in order to ensure that s will remain non-negative. This leads to:

$$Minimize \qquad \frac{1}{2}r^T R^{-1} r - \mu \sum_{k=1}^{p} \ln s_k$$

$$Subject\ to \qquad \begin{aligned} f(x)+s &= 0 \\ g(x) &= 0 \\ r-z+h(x) &= 0 \\ s &\geq 0 \end{aligned} \tag{2.14}$$

The Lagrangian function for the transformed problem of 2.14 is given by:

$$L = \frac{1}{2}r^T R^{-1} r - \mu \sum_{k=1}^{p} \ln s_k - \lambda^T[f(x)+s] - \rho g(x) - \pi[r-z+h(x)] \tag{2.15}$$

Note that, the barrier parameter $\mu \geq 0$ is forced to decrease towards zero as the iterations progress.

The Kuhn-Karroush-Tucker (KKT) optimality conditions for this problem are

$$\begin{aligned} \nabla_s L = &\quad -\mu S^{-1}e - \lambda &= 0 \\ \nabla_\lambda L = &\quad -f(x) - s &= 0 \\ \nabla_\rho L = &\quad -g(x) &= 0 \\ \nabla_\pi L = &\quad -r + z - h(x) &= 0 \\ \nabla_r L = &\quad R^{-1}r - \pi &= 0 \\ \nabla_x L = &\quad -F^T\lambda - G^T\rho - H^T\pi &= 0 \\ &\quad \lambda &\leq 0 \end{aligned} \tag{2.16}$$

where, S is a diagonal matrix, whose K^{th} diagonal element is s_k; F, G and H are the gradients of $f(x)$, $g(x)$ and $h(x)$ respectively; R is a diagonal matrix whose diagonal elements are the measurement variances, $R_{ii} = \sigma_i^2$; e is a vector with all entries equal to 1.0.

The nonlinear functions in 2.16 can be replaced by their first order approximations as given below:

$$\begin{aligned} f(x) &\approx f(x^k) + F\Delta x \\ g(x) &\approx g(x^k) + G\Delta x \\ h(x) &\approx h(x^k) + H\Delta x \\ S^{-1}e &\approx (S^k)^{-1}e - (S^k)^{-2}\Delta s \end{aligned} \tag{2.17}$$

Using the first equation in (2.16) and the fourth and fifth equation in (2.17):

$$S^{-1}e + \frac{1}{\mu}\lambda \approx (S^k)^{-1}e - (S^k)^{-2}\Delta s + \frac{1}{\mu}\lambda = 0 \tag{2.18}$$

The second equation in 2.16 yields:

$$f(x^k) = -s^k \tag{2.19}$$

$$F\Delta x = -\Delta s \tag{2.20}$$

Eliminating Δs and s^k from 2.18, 2.19 and 2.20,

$$\frac{(S^k)^2}{\mu}\lambda + F\Delta x = f(x^k) \tag{2.21}$$

Further eliminating r from the fourth and fifth equations in (2.16) and using the first order approximation of $h(x)$:

$$R\pi + H\Delta x = z - h(x^k) \tag{2.22}$$

Now, the above derived Equations (2.21), (2.22), the second equation in (2.17) and the sixth equation in (2.16) can be combined to obtain:

$$\begin{bmatrix} D & 0 & 0 & F \\ 0 & 0 & 0 & G \\ 0 & 0 & R & H \\ F^T & G^T & H^T & 0 \end{bmatrix} \cdot \begin{bmatrix} \lambda \\ \rho \\ \pi \\ \Delta x \end{bmatrix} = \begin{bmatrix} f(x^k) \\ -g(x^k) \\ z - h(x^k) \\ 0 \end{bmatrix} \tag{2.23}$$

where D is a diagonal matrix, whose k^{th} diagonal element is given by:

$$D_{kk} = \frac{S_k^2}{\mu} \tag{2.24}$$

The entire matrix on the left side will be denoted by K. The solution to the nonlinear optimization problem can be obtained by iteratively solving Equation (2.23) and updating the values of the variables at each iteration.

$$x^{k+1} = x^k + \alpha \cdot \Delta x^{k+1} \tag{2.25}$$

In Equation (2.25), the step length is adjusted via α, a factor used to keep the changes of phase angle to be small enough to maintain linearity of the first order equation and to keep x^{k+1} in the feasible region so that $f(x) \leq 0$, $\lambda \leq 0$. Further discussion of choosing α will be presented in section D.

Consider the case where a UPFC will be used to adjust the real power flow through a branch for a given system loading and generation dispatch condition. The same program, which solves 2.14, can be used to solve for the system states and the required UPFC parameters, by assigning all bus injections and the desired branch flow as measurements. Hence, the developed state estimator can also be used to obtain the required set of UPFC parameters to maintain a desired flow in the network. Note that in this case, the measurement set is chosen on purpose as a minimally observable set including all the injections and the desired line flows through the UPFC branches.

Hence, the solution will exactly satisfy the injections and the desired flows with zero residuals. This will guarantee that the solved UPFC setting will yield the desired flow exactly.

2. Observability analysis

Observability analysis can be carried out using the numerical method [12]. The measurement Jacobian matrix which excludes inequality constraints and given by:

$$J = \begin{bmatrix} G \\ H \end{bmatrix} \tag{2.26}$$

will be decomposed into its lower and upper rectangular factors by using the Peters-Wilkinson [13] method. In case of zero pivots, pseudo measurements will be added to make the system observable. The pseudo measurements will indicate deficiencies in the measurement system, both for the network states as well as for the UPFC parameters.

3. Bad data analysis

The largest normalized residual test is used for bad data analysis [14]. In order to avoid the possibility of singularity of the gain matrix, equality constraints are maintained as part of the measurement set despite the fact that their residuals are known to be zero ahead of time. Artificially small error variances are assigned to these equality constraints in forming the augmented diagonal measurement covariance matrix R_a that includes the variances of both the regular measurements as well as those of the equality constraints. Hence, the residual covariance matrix Ω is computed as shown below:

$$\Omega = R_a - J \cdot (J^T R_a^{-1} J)^{-1} \cdot J^T \tag{2.27}$$

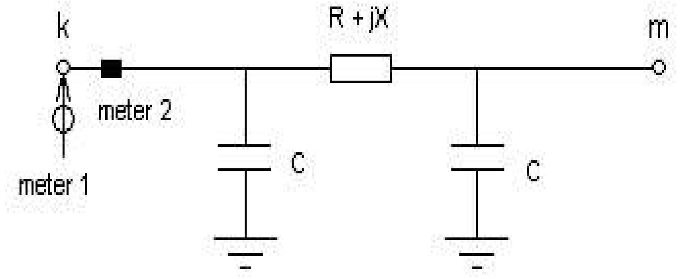


Fig. 3. Candidate measurements on line k-m without UPFC

where J is the Jacobian matrix given in 2.26.

The residuals are then normalized as :

$$r_i^N = \frac{|r_i|}{\sqrt{\Omega_{ii}}}, \quad i = 1, \cdots, m \tag{2.28}$$

and the measurement having the largest normalized residual which is also larger than the detection threshold (set to be 3.0 in the program), is identified as bad data. Identification and elimination of bad data and re-estimation of the system state is repeated, until no more bad data is detected.

4. Measurements equations

This section provides the detailed equations for the measurements incident to a given line, both with and without a UPFC device.

a. Lines without an installed UPFC

Consider the two possible measurements, meter 1 and meter 2 on line k-m as shown in Figure 3.

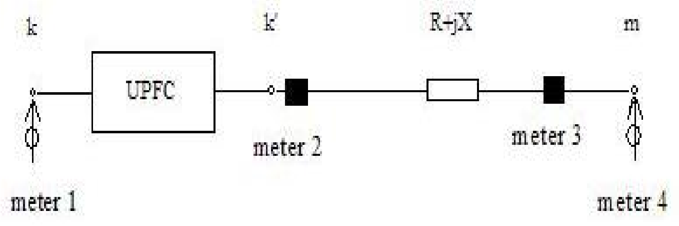


Fig. 4. UPFC and candidate measurements on line k-m

Equations for meter 1 are:

$$P_k = V_k^2 G_{kk} + V_k \sum_{j=1, j\neq k}^{n} V_j (G_{kj}\cos\theta_{kj} + B_{kj}\sin\theta_{kj}) \tag{2.29}$$

$$Q_k = -V_k^2 B_{kk} + V_k \sum_{j=1, j\neq k}^{n} V_j (G_{kj}\sin\theta_{kj} - B_{kj}\cos\theta_{kj}) \tag{2.30}$$

Equations for meter 2 are:

$$P_{km} = -V_k^2 G_{km} + V_k V_m (G_{km}\cos\theta_{km} + B_{km}\sin\theta_{km}) \tag{2.31}$$

$$Q_{km} = V_k^2 (B_{km} - C_{km}) + V_k V_m (G_{km}\sin\theta_{km} - B_{km}\cos\theta_{km}) \tag{2.32}$$

b. Lines controlled by a UPFC

Suppose a UPFC is installed on line k-m. Meters 1, 2, 3, 4 are the measurements that can be placed on line k-m as shown in Figure 4.

Equations for meter 1 are:

$$\begin{aligned} P_k = {} & V_k^2 (G_{kk} + G_{km}) + V_k \sum_{j=1, j\neq k,m}^{n} V_j (G_{kj}\cos\theta_{kj} + B_{kj}\sin\theta_{kj}) \\ & + V_k V_B B_B \sin\theta_{kB} - V_k V_{k'} B_B \sin\theta_{kk'} - V_k V_E B_E \sin\theta_{kE} \end{aligned} \tag{2.33}$$

$$\begin{aligned} Q_k = {} & -V_k^2 (B_{kk} + B_{km} + B_B + B_E) \\ & + V_k \sum_{j=1, j\neq k,m}^{n} V_j (G_{kj}\sin\theta_{kj} - B_{kj}\cos\theta_{kj}) \\ & - V_k V_B B_B \cos\theta_{kB} + V_k V_{k'} B_B \cos\theta_{kk'} + V_k V_E B_E \cos\theta_{kE} \end{aligned} \tag{2.34}$$

Equations for meter 2 are:

$$\begin{aligned} P_{km} &= V_{k'}V_kB_B\sin\theta_{k'k} + V_{k'}V_BB_B\sin\theta_{k'B} && (2.35)\\ Q_{km} &= -V_{k'}^2B_B - V_{k'}V_kB_B\cos\theta_{k'B} && (2.36)\end{aligned}$$

and,

$$\begin{aligned} P_{km} &= -V_{k'}^2G_{km} + V_{k'}V_m(G_{km}\cos\theta_{k'm} + B_{km}\sin\theta_{k'm}) && (2.37)\\ Q_{km} &= V_{k'}^2(B_{km} - C_{km0}) + V_{k'}V_m(G_{km}\sin\theta_{k'm} - B_{km}\cos\theta_{k'm}) && (2.38)\end{aligned}$$

Equations for meter 3 are:

$$\begin{aligned} P_{mk} &= -V_m^2G'_{km} + V_mV_k(G_{kj}\cos\theta_{mk} + B_{kj}\sin\theta_{mk})\\ &\quad +V_mV_B(G'_{km}\cos\theta_{mB} + B'_{km}\sin\theta_{mB}) && (2.39)\\ Q_{mk} &= V_m^2(B'_{km} + C_{km0}) + V_mV_k(G_{kj}\sin\theta_{mk} - B_{kj}\cos\theta_{mk})\\ &\quad +V_mV_B(G'_{km}\sin\theta_{mB} - B'_{km}\cos\theta_{mB}) && (2.40)\end{aligned}$$

Equations for meter 4 are:

$$\begin{aligned} P_m &= V_m^2(G_{mm} + G_{mk} - G'_{km}) + V_m\sum_{j=1,j\neq k,m}^{n} V_j(G_{mj}\cos\theta_{mj} + B_{mj}\sin\theta_{mj})\\ &\quad +V_mV_k(G'_{km}\cos\theta_{mk} + B'_{km}\sin\theta_{mk})\\ &\quad +V_mV_B(G'_{km}\cos\theta_{mB} + B'_{km}\sin\theta_{mB}) && (2.41)\\ Q_m &= -V_m^2(B_{mm} + B_{mk} - B'_{km}) + V_m\sum_{j=1,j\neq k,m}^{n} V_j(G_{mj}\sin\theta_{mj} - B_{mj}\cos\theta_{mj})\\ &\quad +V_mV_k(G'_{km}\sin\theta_{mk} - B'_{km}\cos\theta_{mk})\\ &\quad +V_mV_B(G'_{km}\sin\theta_{mB} - B'_{km}\cos\theta_{mB}) && (2.42)\end{aligned}$$

5. Algorithm

The procedures of the proposed algorithm used in this program of state estimation is listed below. It is based upon the previously presented analysis and the reader is referred to the previous sections for the notation used in the following description of algorithm steps.

Step 1: Read network data and measurements;

Step 2: Initialize: $x^k,\ k = 0$;

Step 3: Form K^k matrix;

Step 4: Calculate the equality and inequality constraints, measurements mismatches, and form the right hand side b vector;

$$b^k = \begin{bmatrix} -f(x^k) \\ -g(x^k) \\ z - h(x^k) \\ 0 \end{bmatrix} \tag{2.43}$$

Step 5: Solve the equation:

$$K^k \cdot \begin{bmatrix} \lambda \\ \rho \\ \pi \\ \Delta x \end{bmatrix} = b^k \tag{2.44}$$

get Δx^k.

Step 6: Update x: $x^{k+1} = x^k + \Delta x^k$;

Step 7: Terminate execution if $\Delta x^k - \Delta x^{k-1} \leq \epsilon$, and go to Step 3; Else, $k = k + 1$ and go to Step 8

Step 8: Stop and print out results.

D. Computational Issues

1. Initialization

The initial state vector x^0 should satisfy $f(x^0) < 0$. So, all of the bus voltages are initialized to 1.0 and phase angles to 0.0. The control variables associated with the UPFC are initialized assuming that these devices are initially out of service, that is, all shunt voltages are set equal to 1.0, i.e. $V_E = 1.0$, $\theta_E = 0.0$, and all series voltages are set equal to 0.0, i.e. $V_B = 0.0$, $\theta_B = 0.0$. However, choosing , V_B, θ_B or θ_E as zero, leads to a singular K matrix. Hence, these variables are assigned very small but non-zero values (e.g. 0.001) in order to avoid singularity during initialization. The barrier parameter μ is initialized to 1.0 and proportional to the duality gap during the iterations [7].

2. Adjusting the barrier parameter and step length

The barrier parameter μ is forced to decrease towards zero during the iterations as the optimal point is approached. The difference between primal and dual objective functions is calculated during each iteration and used to adjust μ. The dual of the problem given in 2.13 is shown below [7]:

$$\begin{array}{lrl} Maximize & \frac{1}{2}r^T R^{-1} r - \lambda^T f(x) - \rho^T g(x) - \pi^T[r - z + h(x)] & \\ & -F^T\lambda - G^T\rho - H^T\pi & = 0 \\ Subject\ to: & R^{-1}r - \pi & = 0 \\ & \lambda & \leq 0 \end{array} \qquad (2.45)$$

The duality gap between the primal and dual objective functions can be expressed as:

$$\delta = \lambda^T f(x) + \rho^T g(x) + \pi^T [r - z + h(x)] \tag{2.46}$$

This gap should always be non-negative and finally become zero at the optimal point. The barrier parameter μ is adjusted according to δ at each iteration as suggested in [7]:

$$\mu = \frac{\delta}{n^2} \tag{2.47}$$

where, n is the dimension of state vector x.

The step length is chosen based on three requirements. The first one is to keep the changes in θ small enough so that the linearity of the first order equation is maintained. The second one is to make the solutions feasible within the inequality constraints so that $f(x) \leq 0$. The third one is to keep $\lambda \leq 0$.

In this implementation, the maximum change for the phase angle is set as 0.2 radians, $\Delta\theta_{lim} = 0.2$. $\widehat{\alpha}_\theta$ is chosen to keep $\widehat{\alpha}_\theta \cdot \Delta\theta_i \leq \Delta\theta_{lim}$.

$$\widehat{\alpha}_\theta = min\left\{\frac{\Delta\theta_{lim}}{|\Delta\theta_i|},\ |\Delta\theta_i| > \Delta\theta_{lim}\right\} \tag{2.48}$$

To keep x in the feasible region, $\widehat{\alpha}_s$ is chosen to keep $f(x^{k+1}) \leq 0$,

$$f(x^{k+1}) \approx f(x^k) + F \cdot \widehat{\alpha}_s \Delta x = -S^k \cdot e - \widehat{\alpha}_s \Delta s \leq 0 \tag{2.49}$$

we get,

$$\widehat{\alpha}_s = min\left\{-\frac{s_i}{\Delta s_i},\ \Delta s_i < 0\right\} \tag{2.50}$$

To keep $\lambda^{k+1} \leq 0$,

$$\lambda^{k+1} = \lambda^k + \widehat{\alpha}_\lambda \cdot (\lambda^{k+1}_{old} - \lambda^k) \leq 0 \tag{2.51}$$

we get,

$$\widehat{\alpha}_\lambda = min\left\{\frac{\lambda^k}{\lambda_i^k - \lambda_{i,old}^{k+1}},\ (\lambda_i^k - \lambda_{i,old}^{k+1}) < 0\right\} \tag{2.52}$$

Step length adjustor α is chosen as,

$$\alpha = min\{1,\ \widehat{\alpha}_\theta,\ \widehat{\alpha}_s,\ \widehat{\alpha}_\lambda\} \tag{2.53}$$

The inequality constraint $f(x^k + \alpha \cdot \Delta x^{k+1}) \leq 0$ is also tested to ensure feasibility.

3. Matrix conditioning

The barrier parameter μ appears in the denominator of the expression for the sub-matrix D in Equation (2.25). As the optimal point is approached, μ will decrease towards zero, causing severe ill conditioning of the K matrix.

One way to overcome ill conditioning is via the application of matrix scaling as follows:

$$\begin{bmatrix} \mu D & 0 & 0 & F \\ 0 & 0 & 0 & G \\ 0 & 0 & R & F \\ \mu F & G^T & H^T & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{\lambda}{\mu} \\ \rho \\ \pi \\ \Delta x \end{bmatrix} = \begin{bmatrix} -f(x^k) \\ -g(x^k) \\ z - h(x^k) \\ 0 \end{bmatrix} \tag{2.54}$$

The effectiveness of scaling will be illustrated via simulations in the next section. The effect of scaling on the conditioning of the K matrix is found to be very significant.

E. Simulation Results

Simulations are carried out on the IEEE 14-bus, IEEE 57-bus and IEEE 118-bus systems. For all the simulations, the tolerance used to define convergence is 10^{-4} and the bad data detection threshold is set to be 3.0. Gaussian errors with zero mean are added to the measurements in all the simulations. The standard deviations are 0.004

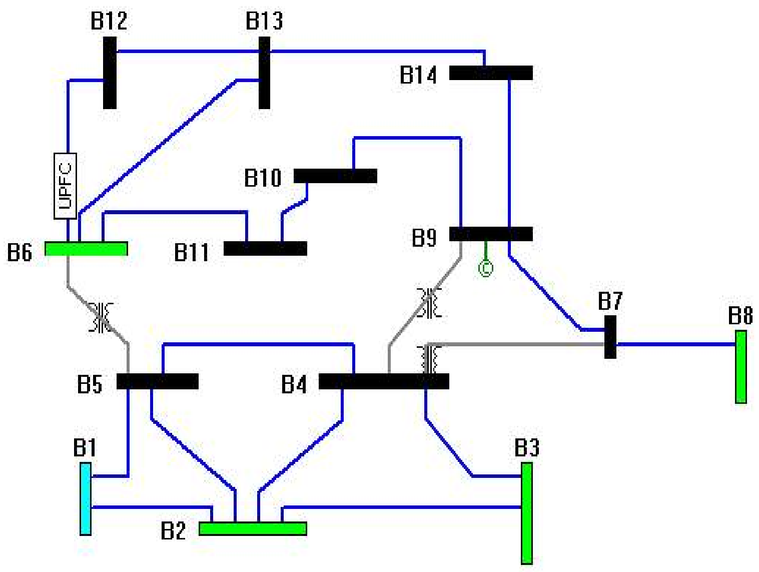


Fig. 5. IEEE 14-bus system embedded with one UPFC

for all the voltage measurements, 0.01 for all the flow measurements and 0.008 for all the injection measurements. The improvements made in the condition number of the coefficient matrix by applying matrix scaling are documented in cases using the IEEE 57-bus and 118-bus test systems.

1. IEEE 14-bus system

There are 4 cases carried on IEEE 14-bus system, which illustrate the response of the estimator to bad data as well as its possible use for setting the parameters of power flow controllers to maintain a desired flow in the network. Bad data processing are carried on cases 1, 2 and 3. For those cases where more than one bad data identification cycle is needed, the largest normalized residual is given for each cycle.

Figure 5 shows the network diagram of the IEEE 14-bus system. A UPFC is installed on line $6-12$ at bus 6. The voltage and capacity constraints and parameters of UPFC are given in Table I.

Table I. Parameters and constraints of UPFC in IEEE 14-bus system

X_B	X_E	$V_{B,max}$	$V_{E,max}$	$S_{B,max}$	$S_{E,max}$
0.05	0.05	1.10	1.10	0.44	0.44

Table II. Measurements data in IEEE 14-bus system for case 1

Voltage Measurement								
Bus	Voltage		Bus	Voltage		Bus	Voltage	
1	1.0684		2	1.0453		3	1.0063	
6	1.0724		8	1.0904				
Flow Measurement								
Line	P	Q	Line	P	Q	Line	P	Q
1-2	1.5709	-0.2062	2-3	0.7412	0.0382	4-7	0.2800	-0.0178
6-11	0.0629	0.0865	6-12	0.1967	0.0229	7-8	0.0065	-0.2017
7-9	0.2822	0.1524	9-14	0.0777	0.0042	12-13	0.1231	0.0186
13-14	0.0703	0.0389	12-6	-0.1852	-0.0278	9-7	-0.2826	-0.1541
Injection Measurement								
Bus	P	Q	Bus	P	Q	Bus	P	Q
3	-0.9410	0.0399	5	-0.0985	-0.0128	6	-0.1171	-0.0386
8	0.0025	0.2171	9	-0.2913	-0.1698	10	-0.0882	-0.0675
11	-0.0354	-0.0157	12	-0.0770	-0.0037	13	-0.1316	-0.0638

Table III. Estimated UPFC control variables and iterations in case 1

	$V\angle\theta$	P	Q	Iteration
Series Source	$0.077\angle 44.08°$	0.0056	0.0127	8
Shunt Source	$1.090\angle$-15.22°	-0.0056	0.4378	

Table IV. Bad data analysis results in case 1

r_N^{max}	Measurement	Result	Cycle
1.8933	$Volt_3$	No Bad Data	1

a. Case 1: State estimation without measurement errors

This case illustrates that the status of the installed UPFC can be estimated provided that there are enough measurements to make the network and the device parameters observable. Measurements that are assumed to be available for this case are listed in Table II.

The estimated UPFC control variables and the number of iterations are shown in Table III. There is no bad data in this case as shown in Table IV.

As can be observed from Table III, $P_B + P_E = 0$, validating that no real power exchange takes place between UPFC and the power system. The voltages and power ratings of the series and shunt voltage sources modeling the UPFC are all within their respective limits.

b. Case 2: Introducing bad data

This case is identical to Case 1, except for a single bad data, which is intentionally introduced by changing the sign of real power injection measurement at bus 9 from

Table V. Estimated UPFC control variables and iterations in case 2

	$V\angle\theta$	P	Q	Iteration
Series Source	0.077∠44.08°	0.0056	0.0127	8
Shunt Source	1.091∠-15.20°	-0.0056	0.4389	

Table VI. Bad data analysis results in case 2

r_N^{max}	Measurement	Result	Cycle
30.5198	$P_{Inj,9}$	Bad Data	1
1.8773	$Volt_3$	No Bad Data	2

negative to positive. The final estimation results of UPFC and number of iterations are shown in Table V. Applying the largest normalized residual test, bad data is identified as shown in Table VI.

c. Case 3: Operating the UPFC at its capacity limit

This case simulates multiple conforming bad data and the use of inequality constraints to aid bad data identification. Case 1 is modified by introducing a 0.10 per unit increase in both injection measurement P_{12} and flow measurement P_{6-12}. Note that these will then become interacting measurements with conforming bad data.

Initially, state estimation is run by excluding all the inequality constraints of the UPFC. The estimation results and the number of iterations are shown in Table VII. In this case, the largest normalized residual test falsely identified flow measurement P_{12-6} as bad data shown in Table VIII.

Next, the operation limits given in Table I for the UPFC are activated. In this case both bad measurements P_{12} and P_{6-12} are correctly identified as bad data. Normalized residuals are shown in Table VIII.

Table VII. Estimated UPFC control variables and iterations in case 3

	$V\angle\theta$	P	Q	Iteration
Series Source	0.090∠45.08°	0.0077	0.0175	8
Shunt Source	1.090∠-15.46°	-0.0077	0.4710	

Table VIII. Bad data analysis results in case 3

Ineq. Constr.	r_N^{max}	Measurement	Result	Cycle
Without	10.5186	$P_{Flow,12-6}$	Bad Data	1
	1.8931	$Volt_3$	No Bad Data	2
With	9.6321	$P_{Inj,12}$	Bad Data	1
	10.1760	$P_{Flow,6-12}$	Bad Data	2
	1.8920	$Volt_3$	No Bad Data	3

d. Case 4: Controller parameter determination

This case illustrates the alternative usage of this estimator in order to determine the controller parameters of UPFC for a desired operating condition. It is assumed that the net power injections at each bus are specified as shown in Table IX and the real power flow through line $6-12$ is desired to be $0.15p.u.$ This is to be accomplished via a UPFC installed on line $6-12$. The series and shunt voltage sources are set equal to 0.073 and $1.09p.u.$ respectively. The required parameter setting for the UPFC to accomplish the required flow is estimated as shown in Table X. Note that no bad data can be identified in this case, because the measurement set is intentionally designed as a minimally observable set, i.e. every measurement is critical in this case.

Table IX. Measurements data in IEEE 14-bus system for case 4

Voltage Measurement								
Bus	Voltage		Bus	Voltage		Bus	VOltage	
1	1.0583							
Injection Measurement								
Bus	P	Q	Bus	P	Q	Bus	P	Q
2	0.1859	0.2199	3	-0.9535	0.0446	4	-0.4661	0.0396
5	-0.0641	-0.0170	6	-0.1124	-0.0436	7	0.0000	0.0000
8	0.0033	0.2233	9	-0.2933	-0.1794	10	-0.0919	-0.0509
11	-0.0277	-0.0018	12	-0.0669	-0.0230	13	-0.1132	-0.0494
14	-0.1504	-0.0374						

Table X. Estimated UPFC control variables and iterations in case 4

	$V\angle\theta$	P	Q	Iteration
Series Source	$0.073\angle 0.64°$	0.0081	0.0096	8
Shunt Source	$1.090\angle\text{-}14.50°$	-0.0081	0.4264	

2. IEEE 57-bus system

In this modified IEEE 57-bus test system, a UPFC is installed on line 6-8 near bus 6 as shown in Figure 6.

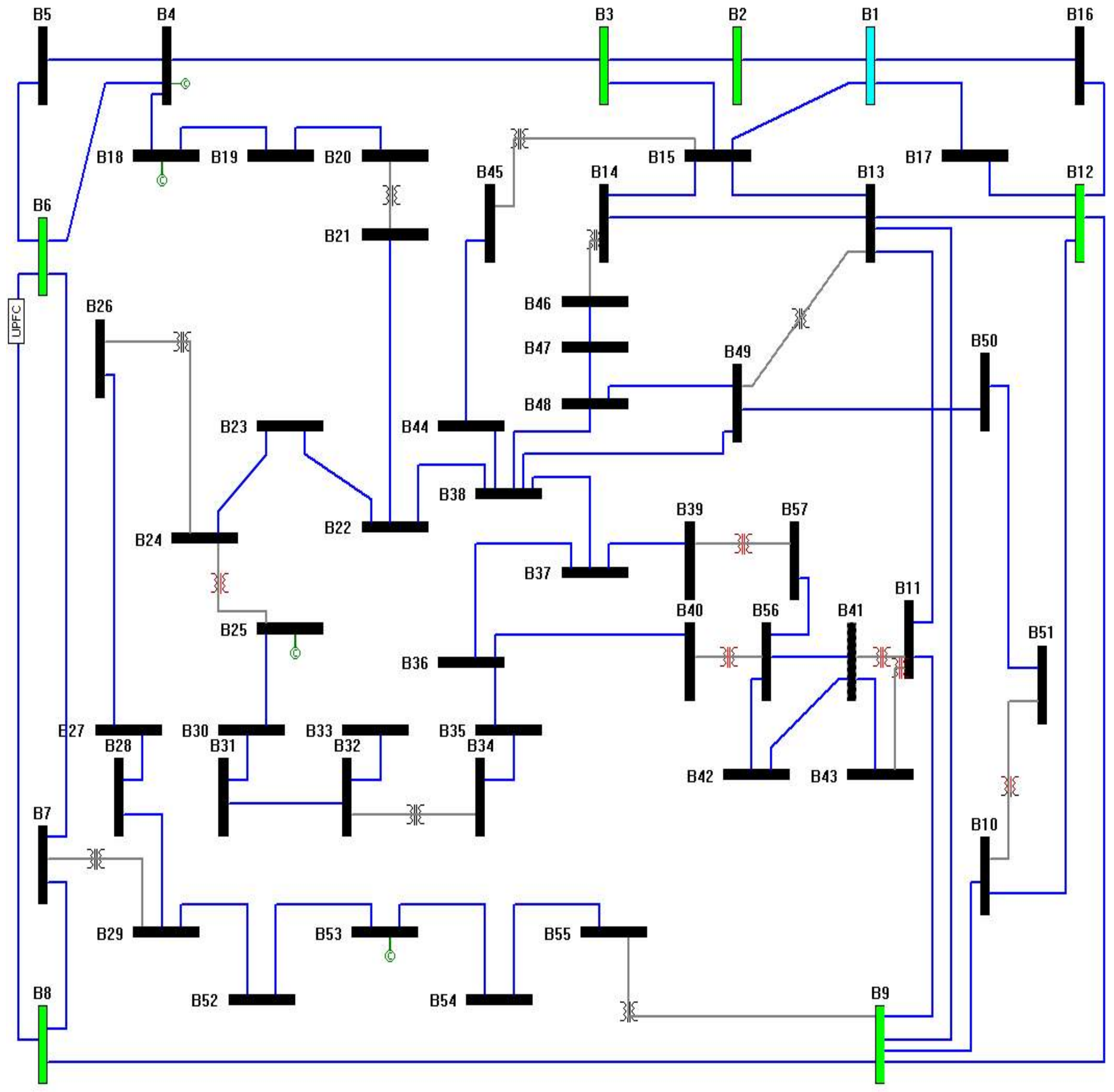


Fig. 6. IEEE 57-bus system embedded with one UPFC

The information of UPFC and measurements is given in Tables XI, XII and XIII.

Table XI. Parameters and constraints of UPFC in IEEE 57-bus system

X_B	X_E	$V_{B,max}$	$V_{E,max}$	$S_{B,max}$	$S_{E,max}$
0.05	0.05	1.10	1.10	1.70	1.70

Table XII. Voltage measurements information in IEEE 57-bus system

Bus	Voltage	Bus	Voltage	Bus	Voltage	Bus	Voltage	Bus	Voltage
1	1.0483	2	1.0083	3	0.9834	6	0.9818	8	1.0022

Table XIII. Flow and injection measurements data in IEEE 57-bus system

Flow Measurement								
Line	P	Q	Line	P	Q	Line	P	Q
6-7	-0.6249	0.1181	6-8	0.3764	0.77900	8-9	2.0259	0.1669
13-15	-0.4001	0.0216	18-19	0.0303	0.0187	19-20	0.0044	0.0035
23-24	0.0335	0.0025	24-25	0.1434	0.0306	24-26	-0.1039	-0.0078
26-27	-0.1041	-0.0325	27-28	-0.1843	-0.0300	25-30	0.0760	0.0495
22-38	-0.1105	-0.0417	41-43	-0.1353	-0.0283	15-45	0.3439	0.0509
50-51	-0.1143	0.0019	40-56	0.0547	-0.0695	9-55	0.1987	-0.0587
3-2	-1.0260	0.0375	5-4	-0.2327	0.0870	8-6	-0.2700	0.0944
11-9	-0.1904	-0.8756	13-11	0.0215	-0.0155	15-14	0.6409	0.0606
28-27	0.2092	0.1324	33-32	-0.03734	0.0360	36-35	0.1159	-0.0212
37-36	0.1669	0.0673	38-37	0.2158	0.1493	42-41	-0.0940	-0.0352
44-38	0.2355	-0.0299	38-44	-0.2095	0.0509	9-55	0.1987	0.1015
Injection Measurement								
Bus	P	Q	Bus	P	Q	Bus	P	Q
3	-0.0015	-0.1370	5	-0.1243	-0.04783	6	-0.7432	-0.2656
8	3.0024	-0.5765	10	-0.0301	-0.02562	12	-0.6742	1.0233
13	-0.1707	-0.0101	16	-0.4416	-0.0311	17	-0.4268	-0.0672
20	-0.0305	-0.0248	27	-0.1090	-0.0063	28	-0.0543	-0.0221
29	-0.1558	-0.0141	30	-0.0423	-0.0138	31	-0.0510	-0.0168
33	-0.0223	-0.0058	38	-0.1611	-0.0613	44	-0.1105	-0.0139
49	-0.1928	-0.0923	52	-0.0430	-0.0112	54	-0.0282	-0.0079
55	-0.0686	-0.0219	56	-0.0890	-0.0324	57	-0.0654	-0.0210

Table XIV. Estimated UPFC control variables and iterations of IEEE 57-bus system

	$V\angle\theta$	P	Q	Iteration
Series Source	$0.298\angle 35.191°$	-0.0948	0.2497	8
Shunt Source	$1.052\angle$-$11.264°$	0.0948	1.5103	

Table XV. Bad data analysis results of IEEE 57-bus system

r_N^{max}	Measurement	Result	Cycle
42.6161	$P_{Flow,38-44}$	Bad Data	1
9.7093	$Q_{Flow,9-55}$	Bad Data	2
2.6750	$Q_{Flow,8-6}$	No Bad Data	3

The estimation results for UPFC control variables are shown in Table XIV. The cases are also run with bad data in the real power flow through branches 38-44 by changing the signs from negative to positive and the reactive power flow through branches 9-55 from positive to negative respectively. Both bad data are successfully identified by the estimator. Maximum normalized residuals obtained after the initial bad data processing cycle are shown in Table XV.

It is noticed that the condition number of the K matrix deteriorates as the system size increases. Table XVI shows the comparisons of the iterations and the condition number of the K matrix with and without using scaling. While scaling is observed to improve the condition number significantly, the iteration count is not affected for this case. Hence, a much larger size test system, namely the IEEE 118-bus test system is considered next.

Table XVI. Comparison of simulations carried on IEEE 57-bus system

Method	Iteration Number	Condition Number
Non-Scaled	9	10^{41}
Scaled	9	10^{7}

Table XVII. Comparison of simulations carried on IEEE 118-bus system

FACTS	Method	Iterations	Condition Number	Measuements		
				Volt	Inj.	Flow
17-18	Non-Scaled	11	10^{57}	12	81	43
	Scaled	11	10^{5}			
17-18, 89-90	Non-Scaled	Not Converged	10^{54}			
	Scaled	8	10^{8}			

3. IEEE 118-bus system

The effect of scaling on the convergence characteristics of the estimator is studied further using the IEEE 118-bus system. Also, the number of UPFCs is increased from one to two, one being placed on line 17-18 at bus 17 terminal and the other on line 89-90 at bus 89 terminal. Simulations are carried out with only one installed UPFC on line 17-18 as well as with both of them installed. The comparative results are shown in Table XVII. It is noted that, not only the increased system size but also adding more UPFCs contribute to the deterioration of the condition number of the K matrix. Use of scaling significantly improves the convergence characteristics, particularly for larger systems.

F. Conclusions

This chapter presents an algorithm for state estimation of power systems embedded with FACTS devices. While only the Unified Power Flow Controller (UPFC) is used in the developed program, other types of controllers can easily be integrated into the developed prototype with minor effort. This program may have dual purpose. It can be used to estimate the controller parameters along with system states during normal operation. It can also be used to determine the required controller settings in order to maintain a desired power flow through a given line.

Several computational issues related to the initialization, choice of the proper step length and ill-conditioning of the matrices are studied and solutions are proposed. Simulations are carried out on IEEE 14-bus, 57-bus and 118-bus test systems to illustrate the performance of the developed state estimator in the presence of bad data. The estimator is also shown to be useful in determining the required UPFC settings corresponding to a desired power flow through the controlled line.

CHAPTER III

OPTIMAL PMU PLACEMENT

Power systems are rapidly becoming populated by Phasor Measurement Unit (PMU), which have multiple uses at substations. They provide valuable phasor information for protection and control of power systems during abnormal operation. In the steady-state operation, these synchronized phasors will still be very useful in monitoring the system state. This chapter describes the study undertaken to determine the optimal locations of PMUs for a given power system.

A. Introduction

Secure operation of power systems requires close monitoring of the system operating conditions. This is traditionally accomplished by the state estimator which resides in the control center computer and has access to the measurements received from numerous substations in the monitored system. These measurements are commonly provided by the remote terminal units (RTU) at the substations and include real/reactive power flows, power injections, and magnitudes of bus voltages and branch currents. Until recently, available measurement sets did not contain phase angle measurements due to the technical difficulties associated with the synchronization of measurements at remote locations. Global positioning system (GPS) alleviated these difficulties and lead to the development of phasor measurement unit.

PMUs are devices, which use synchronization signals from GPS satellites and provide the positive sequence phasor voltages and currents measured at a given substation [15]. While PMUs are not yet found at every substation, their utilization in substations for protection and control functions is rapidly increasing. As they become available in large numbers, they can provide valuable information for EMS applications as well. One such application, which will be significantly affected by the introduction of PMUs, is the state estimator.

The idea of using direct phasor measurements for system monitoring applications including the specific case of state estimation has first been suggested in [15, 16]. The problem of choosing PMU locations for network observability is addressed in [17]. That study assumes that each PMU provides voltage and current measurements at its associated bus and all incident branches. It is therefore possible to fully monitor the system by using relatively less number of PMUs than the number of system buses. This problem is solved by using a graph theoretic observability analysis and an optimization method based on Simulated Annealing in [17]. Possible loss or failure of PMUs is not considered in that study. As the considered systems increase in size such combinatorial optimization solutions may become computationally demanding.

In this chapter, two alternative procedures are introduced to solve the PMU placement problem. First, a numerical method based on an integer programming formulation will be presented. This formulation facilitates the analysis of network observability and is general enough to account for the existence of zero and non-zero power injections and power flow measurements. There are two different ways to treat zero and non-zero power injection measurements using this method. One is via the use of non-linear constraints, and the other one is based on a special topology transformation. These two methods will be illustrated in this chapter using the IEEE 14-bus as an example. The procedure can also be extended to account for loss of single PMUs. Some preliminary results are already presented in [18] and [19].

Another procedure presented in this chapter is a topology-based method. It is more applicable for a system, which has a lot of measurements forming several observable islands. Placing a few extra PMUs at strategic boundary buses will make the entire network observable.

The chapter is organized in four sections.

- Section A is introduction part.
- Section B contains the details of the proposed algorithm for the solution of the optimal PMU placement problem. The two alternative methods, numerical method and the topology-based method, are described first, followed by the discussion of the case where loss of single PMU is considered as an added reliability criterion.
- Section C presents simulation results obtained by applying the developed methods to standard IEEE test systems with typical measurement configurations.
- Conclusions and final remarks are given in the last section.

B. Algorithm for Optimal PMU Placement

PMUs provide two types of measurements: bus voltage phasors and branch current phasors. Depending on the type of PMUs the number of channels used for measuring voltage and current phasors will vary. Here, it is assumed that each PMU has enough channels to record the bus voltage phasor at its associated bus and current phasors along all branches that are incident to this bus. The objective of the PMU placement problem is to render an observable system by using a minimum number of PMUs. An example of an optimally placed set of PMUs in a 14-bus system is shown below in Figure 7.

In this system, there are three PMUs placed at buses 2, 6 and 9 respectively. Bus 7 is the only zero injection bus. The PMU at bus 2 can not only measure the voltage phasor of bus 2, but also the current phasors of branches 2-1, 2-3, 2-4 and 2-5. Using Ohm's law, the voltage phasors at buses 1, 3, 4 and 5 can be obtained from

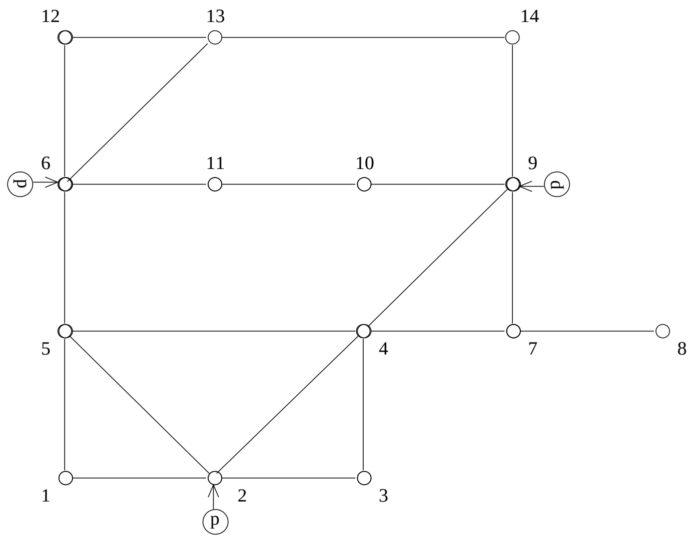


Fig. 7. IEEE 14-bus system with 3 PMUs

the branch currents and the voltage at bus 2. Having determined voltage phasors at buses 1, 2, 3, 4, and 5, the current phasors of branches 1-5, 3-4 and 4-5 can be calculated. Following the same logic, PMU at bus 6 can measure the voltage phasor at bus 6 and the current phasors of branches 6-5, 6-11, 6-12 and 6-13, thus allowing the calculation of the voltage phasors at buses 5, 11, 12, 13 and the current phasor of branch 12-13. PMU at bus 9 can measure the voltage phasor at bus 9 and the current phasors of branches 9-4, 9-7, 9-10, 9-14 and allow the calculation of the voltage phasors at buses 4, 7, 10, 14, and the current phasors of branches 4-7. As voltage phasors of buses 10, 11, 13, 14 are known, current phasors of branches 10-11 and 13-14 can now also be calculated. Using the known current phasors of branches 4-7 and 9-7, and the zero injection at bus 7, the current phasor of branch 7-8 can be derived using the Kirchhoff's Current Law. The only remaining unknown voltage phasor at bus 8 can now be calculated by using the voltage phasor at bus 7 and the current phasor of branch 7-8. Thus the entire system becomes observable by placing only three PMUs at buses 2, 6, 9 and by considering the zero injection at bus 7.

In the following sections, two different procedures are introduced to solve the PMU placement problem. One is an integer based programming method, and the other one is a topology-based method. Both of these methods will be discussed in detail via the use of the IEEE 14-bus example in the following sections.

1. Integer programming based method

In this section, a numerical method based on Integer Programming will be presented to solve the optimal PMU placement problem. For an n-bus system, the PMU placement problem can be formulated as follows:

$$\begin{aligned} &Minimize && \sum_{i}^{n} \omega_i \cdot x_i \\ &Subject\ to && f(X) \geq \hat{1} \end{aligned} \tag{3.1}$$

where, ω_i is the cost of the PMU installed at bus i.

X is a binary decision variable vector, whose entries are defined as

$$x_i = \begin{cases} 1 & if\ a\ PMU\ is\ installed\ at\ bus\ i \\ 0 & otherwise \end{cases} \tag{3.2}$$

$f(X)$ is a vector function, whose entries are non-zero if the corresponding bus voltage is solvable using the given measurement set and zero otherwise. $\hat{1}$ is a vector whose entries are all ones.

The objective function, which is the inner product of the binary decision variable vector and the cost vector represents the total costs of the selected PMUs. Constraint functions ensure full network observability while minimizing the total cost of the PMUs.

The procedure for building the vector function $f(X)$ will be described for four possible cases, where (1) no PMU measurements or conventional measurements exist,

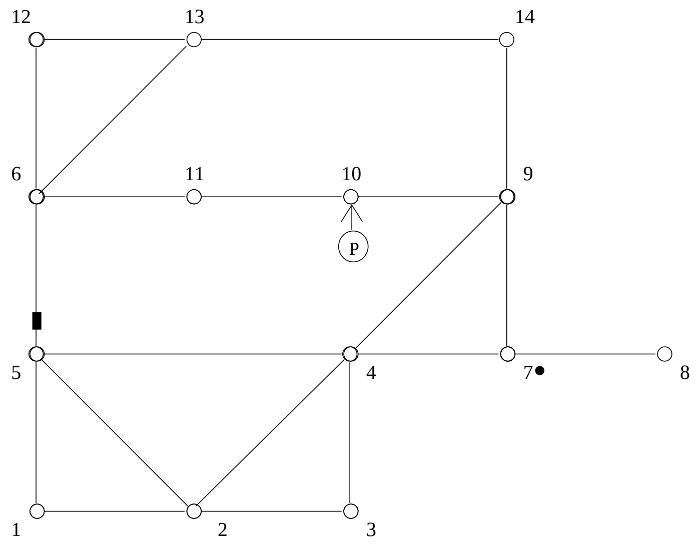


Fig. 8. IEEE 14-bus system with conventional measurements

(2) only flow measurements exist, (3) both flow and injection measurements (they may be zero injections or measured injections) exist or (4) PMU measurements and conventional flow or injection measurements exist. The procedure for each case will be described using IEEE 14-bus system as an example. However, the entire procedure is actually implemented and successfully tested on different size systems with diverse measurement configurations.

Consider the IEEE 14-bus system and its measurement configuration shown in Figure 8. In this system, there is a conventional paired flow measurement on line 5-6, which is represented by a black box, and a PMU measurement at bus 10. Also, note that bus 7 is a zero injection bus which is indicated by the black dot next to it.

a. Case 1: A system with no PMUs or conventional measurements

In this case, the PMU, the flow measurement and the zero injection are ignored. In order to form the constraint set, the binary connectivity matrix A, whose entries are defined as below, will be formed first.

$$A_{k,m} = \begin{cases} 1 & if\ k = m\ or\ k\ and\ m\ are\ connected \\ 0 & otherwise \end{cases} \tag{3.3}$$

Matrix A can be directly obtained from the bus admittance matrix by transforming its entries into binary form. Building the A matrix for the 14-bus system yields:

$$A = \begin{bmatrix} 1&1&0&0&1&0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&1&1&1&1&0&1&0&1&0&0&0&0&0 \\ 1&1&0&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0&1&1&1&0 \\ 0&0&0&1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&1&0&1&1&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&1&0&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0&1&0&0&0&1&1 \end{bmatrix} \tag{3.4}$$

Based on the acknowledgement that by placing a PMU on one bus, the voltage phasors of this bus and its neighbouring buses can be calculated, we can get

$$f(X) = A \cdot X$$

Thus, in order to make all bus voltage phasors solvable for this case, the following inequality constraints should be satisfied:

$$f(X) = A \cdot X = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_9 & \geq 1 \\ f_5 = x_1 + x_2 + x_4 + x_5 + x_6 & \geq 1 \\ f_6 = x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_7 = x_4 + x_7 + x_8 + x_9 & \geq 1 \\ f_8 = x_7 + x_8 & \geq 1 \\ f_9 = x_4 + x_7 + x_9 + x_{10} + x_{14} & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \tag{3.5}$$

The operator "+" serves as the logical "OR" and the use of "1" in the right hand side of the inequality ensures that at least one of the variables appearing in the sum will be non-zero. For example, consider the constraint associated with bus 1 as given below:

$$f_1 = x_1 + x_2 + x_5 \geq 1$$

It implies that at least one PMU must be placed at either one of buses 1, 2 or 5 in order to make bus 1 observable.

b. Case 2: A system with some flow measurements

This case considers the situation where some flow measurements may be present. Flow measurement on branch 5-6 in the 14-bus example system will be used to illustrate the approach on how to deal with existing flow measurements. Existence of this flow measurement will lead to the modification of the constraints for buses 5 and 6 accordingly. Modification follows the observation that having a flow measurement along a given branch allows the calculation of one of the terminal bus voltage phasors when the other one is known. Hence, the constraint equations associated with the terminal buses of the measured branch can be merged into a single constraint. In the case of the example system, the constraints for buses 5 and 6 are merged into a joint constraint as follows,

$$\begin{cases} f_5 = x_1 + x_2 + x_4 + x_5 + x_6 & \geq 1 \\ f_6 = x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \end{cases} \Rightarrow$$
$$f_{5\_new} = f_5 + f_6 = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} \geq 1 \tag{3.6}$$

which implies that if either one of the voltage phasors at bus 5 or 6 is observable, the other one will be observable.

Applying this modification to the constraints for the 14-bus system, the following set of constraints will be obtained,

$$f(X) = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_9 & \geq 1 \\ f_{5\_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_7 = x_4 + x_7 + x_8 + x_9 & \geq 1 \\ f_8 = x_7 + x_8 & \geq 1 \\ f_9 = x_4 + x_7 + x_9 + x_{10} + x_14 & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \tag{3.7}$$

c. Case 3: A system with both injection measurements (some of which may be zero injection pseudo-measurements) and flow measurements

In this case, injection measurements whether they are real measurements or zero injections, are treated the same way.

Consider again the same 14-bus system, where bus 7 is a zero injection bus. It is easy to see that if the phasor voltages at any three out of the four buses 4, 7, 8 and 9 are known, then the fourth one can be calculated using the Kirchhoff's Current Law applied at bus 7 where the net injected current is known.

There are two different ways to treat the injection measurements and form the constraints. One is to form non-linear constraints for the neighbors of the buses, which have injection measurements installed. The alternative approach involves a topology transformation. These will be discussed separately next.

I. Forming non-linear constraints

One way to treat the injection buses is to modify the constraints associated with the neighboring buses of these buses and form a set of non-linear constraints. This is accomplished as shown below.

To treat the zero injection bus 7 in the IEEE 14-bus system, constraints associate with its neighboring buses 4, 8 and 9 will be modified as follows,

$$\begin{aligned} f_4 &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + f_7 \cdot f_8 \cdot f_9 & \geq 1 \\ f_8 &= x_7 + x_8 + f_4 \cdot f_7 \cdot f_9 & \geq 1 \\ f_9 &= x_4 + x_7 + x_9 + x_{10} + x_{14} + f_4 \cdot f_7 \cdot f_8 & \geq 1 \end{aligned} \tag{3.8}$$

Note that the operator "$\cdot$" serves as the logical "AND" in the above equations. For example, the expression of f_8 in Equation (3.8) implies that bus 8 will be observable if there is a PMU on either bus 7 or 8, or if buses 4, 7 and 9 are all observable.

The expressions for f_i in Equation (3.8) can be further simplified by using the following properties of the logical AND ($\cdot$) and OR ($+$) operators:

Given two sets A and B, where set A is a subset of set B, then $A + B = B$ and $A \cdot B = A$. For instance, substituting the expression for f_7 in the expression for f_4, f_4 can be written as:

$$\begin{aligned} f_4 &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + f_7 \cdot f_8 \cdot f_9 \\ &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + (x_4 + x_7 + x_8 + x_9) \cdot f_8 \cdot f_9 \\ &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_4 \cdot f_8 \cdot f_9 + x_7 \cdot f_8 \cdot f_9 + x_8 \cdot f_8 \cdot f_9 + x_9 \cdot f_8 \cdot f_9 \\ &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot f_8 \cdot f_9 \end{aligned}$$

Note that the expression for f_7 should also include an extra product term given by $f_4 \cdot f_8 \cdot f_9$. However this higher order term will be neglected. In all our simulated cases, this approximation is found to have no effect on the optimization.

Carrying on the simplifications, the product $x_4 \cdot f_8 \cdot f_9$ is eliminated because it is the subset of x_4, which already exists in the expression. Using similar reasoning, $x_7 \cdot f_8 \cdot f_9$ and $x_9 \cdot f_8 \cdot f_9$ are also eliminated. Then, substituting the expression of f_8 yields:

$$\begin{aligned} f_4 &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot f_8 \cdot f_9 \\ &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot (x_7 + x_8) \cdot f_9 \\ &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot f_9 \end{aligned}$$

Substituting for f_9:

$$\begin{aligned} f_4 &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot f_9 \\ &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot (x_4 + x_7 + x_9 + x_{10} + x_{14}) \\ &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot x_{10} + x_8 \cdot x_{14} \end{aligned}$$

Finally, the expression for f_4 simplified to the following:

$$f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot x_{10} + x_8 \cdot x_{14}$$

Applying similar simplification logic to all other expressions, the constraint set can be written as follows:

$$\begin{aligned} f_4 &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot x_{10} + x_8 \cdot x_{14} \geq 1 \\ f_8 &= x_4 + x_7 + x_8 + x_9 \geq 1 \\ f_9 &= x_4 + x_7 + x_9 + x_{10} + x_{14} + x_2 \cdot x_8 + x_3 \cdot x_8 + x_5 \cdot x_8 \geq 1 \end{aligned}$$

Note that the constraints corresponding to all other buses will remain. One exception is that the constraint for bus 7 where the injection is measured (or known) will be eliminated from the constraint set. The reason for removing the constraints associated with injection buses is that their effects are indirectly taken into account by the product terms augmented to the constraints associated with the neighboring buses.

The constraints for this case are shown in Equation (3.9).

$$f(X) = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_9 \cdot x_{10} + x_8 \cdot x_{14} & \geq 1 \\ f_{5\_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_8 = x_4 + x_7 + x_8 + x_9 & \geq 1 \\ f_9 = x_4 + x_7 + x_9 + x_{10} + x_{14} + x_2 \cdot x_8 + x_3 \cdot x_8 + x_5 \cdot x_8 & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \tag{3.9}$$

This way of forming constraints for zero injection buses or buses which have injection measurements is complicated and time consuming. It is also noticed that nonlinear part will be introduced in the constraints and it will further slow down the integer programming. Hence, the following alternative method is developed for systems with a relatively large number of injections.

II. Topology transformation

This alternative method referred here as the topology transformation is developed for

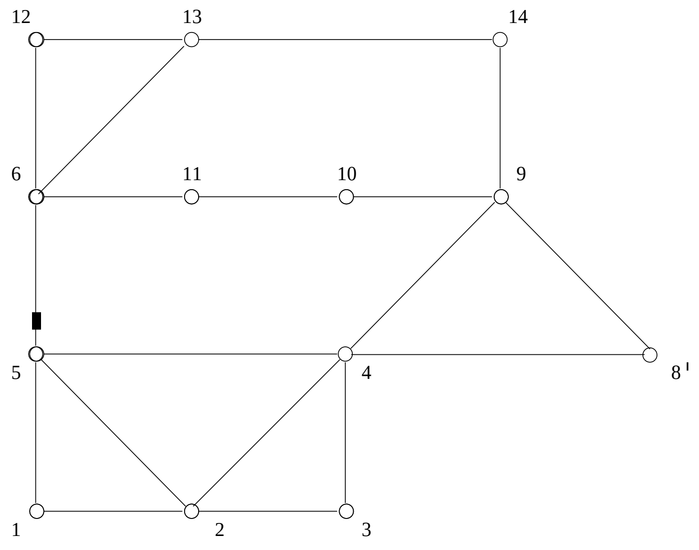


Fig. 9. Apply topology transformation on IEEE 14-bus system

handling injection measurements. The main idea is to merge the bus which has the injection measurement, with any one of its neighbors. This is based on the observation that if the voltage phasors of all its neighbors are known, the voltage phasor of this injection bus can be calculated by the Kirchhoff's Current Law.

Figure 9 shows the updated system diagram after the merger of buses 7 and 8 into a new bus $8'$. The newly created branch $8' - 9$ reflects the original connection between buses 7 and 9.Hence, the constraints vector function can be formed as shown in Equation (3.10).

$$f(X) = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_{8'} + x_9 & \geq 1 \\ f_{5\_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_{8'} = x_4 + x_{8'} + x_9 & \geq 1 \\ f_9 = x_4 + x_{8'} + x_9 + x_{10} + x_{14} & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \tag{3.10}$$

Topology transformation is faster and will not introduce any nonlinear part in constraint set. Yet a word of caution needs to be added here in that, if the optimal solution chooses the newly formed fictitious bus (merger of two actual buses) as a candidate bus, it may indicate to place one PMU on one of these two buses or two PMUs on both. In this case, a topology analysis needs to be applied to check the observability of the system. This also assures that the minimum number of PMUs will be placed.

d. Case 4: A system which contains conventional flow and injection measurements as well as PMU measurements

This case considers the most general situation where both conventional flow or injection measurements and PMU measurements may be present, but not enough to make the entire system observable.

To build the constraints for this case is simple. After forming the constraint equation $f(X)$ according to the procedure described above, simply replace all the x_i by 1, where i represents the bus with an already installed PMU.

Consider again the 14-bus system shown in Figure 8. There is a PMU installed at bus 10, all x_{10} terms appearing in the constraint set in Equations (3.9) and (3.10) will have to be replaced by 1. The modified equations in Equation (3.9) will look as follows:

$$\begin{aligned}
f_4 &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot x_{10} + x_8 \cdot x_{14} \\
&= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \\
f_9 &= x_4 + x_7 + x_9 + x_{10} + x_{14} + x_2 \cdot x_8 + x_3 \cdot x_8 + x_5 \cdot x_8 = 1 \\
f_{10} &= x_9 + x_{10} + x_{11} = 1 \\
f_{11} &= x_6 + x_{10} + x_{11} = 1
\end{aligned}$$

Now that f_9, f_{10} and f_{11} are all ones, they can be removed from the constraint equations, Equation (3.9) will be modified as:

$$f(X) = \begin{cases}
f_1 = x_1 + x_2 + x_5 & \geq 1 \\
f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\
f_3 = x_2 + x_3 + x_4 & \geq 1 \\
f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_8 + x_9 & \geq 1 \\
f_{5\_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\
f_8 = x_4 + x_7 + x_8 + x_9 & \geq 1 \\
f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\
f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\
f_{14} = x_9 + x_{13} + x_{14} & \geq 1
\end{cases} \tag{3.11}$$

Following the same logic, Equation (3.10) can be modified as:

$$f(X) = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_{8'} + x_9 & \geq 1 \\ f_{5\_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_{8'} = x_4 + x_{8'} + x_9 & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \tag{3.12}$$

e. Case 5: Placement strategy against loss of a single PMU

The above discussion implicitly assumes that all PMUs are free of defects and their failure is not considered as a possibility. In practice, this assumption may not always hold true due to unexpected failures in these devices or gross errors introduced by the noise in the communication system. Therefore, it might be prudent to consider at least the case of single PMU failure as a possible contingency in the formulation and solution of the PMU placement problem.

Note that the above-presented formulation yields a set of PMUs that form a critical set, which means loss of any single PMU will lead to an unobservable system. In order to upgrade this set so that it will remain insensitive against the loss of single PMUs, a back-up set is chosen by using the same approach. In the process of choosing the backup set, those PMUs that are already selected as members of the primary set are disregarded. This can be done easily by removing all the x_i terms in the constraint functions, where bus i is in the primary set, in order to avoid picking up the same bus which appears in primary set. Hence, the resulting set of primary and backup PMUs will constitute the new solution that can withstand the loss of single PMUs while maintaining an observable system. As expected, accounting for PMU losses increases the required number of PMUs significantly. This is the cost of added reliability.

2. Topology based method

The integer programming based procedure is quite effective in systematically placing PMUs in a system where there are very few existing measurements. It can also be used for placing PMUs in a system which is to be exclusively monitored by PMUS only. However, most of today's power systems already have a significant number of conventional measurements and PMUs are planned to be installed to enhance the existing measurement system. For such systems which may be unobservable and have few observable islands, a topology based method can be easier to apply. The method will find the strategic locations of PMUs by merging the observable islands. Since only boundaries buses contribute to the process of merging observable islands, these will be the strategic locations for placing PMUs.

First a well-documented numerical observability analysis method [20] is carried out to determine the observable islands. Then the boundary bus, which connects to the maximum number of other islands or the one which has the maximum number of branches connected to other islands will be chosen to place a PMU. In order to take advantage of the injection measurements, after the selection of one PMU location, the numerical observability analysis will be re-applied to update the observable islands. The same procedure of selecting the position of PMU is repeated until all islands merge into a single observable island.

Consider the IEEE 14-bus system example shown in Figure 10, where there are 5 initially observable islands. Boundary buses are identified as buses 1, 2, 5, 6, 9, 10, 11, 13 and 14. Among them bus 5, 9, 10 and 14 are connected to two

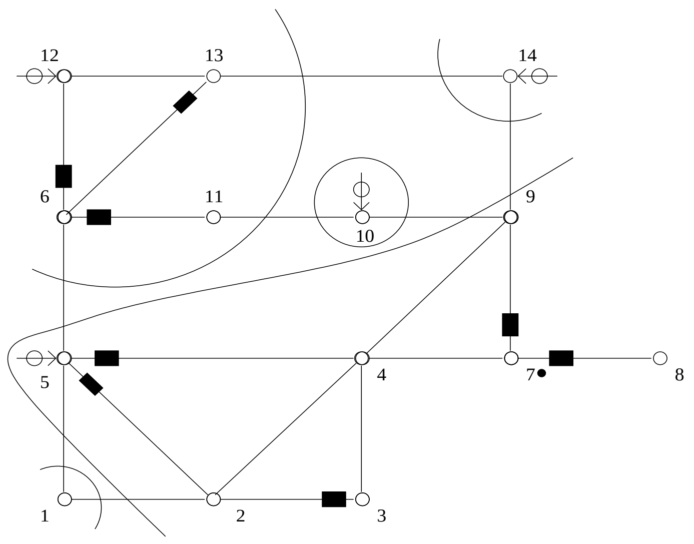


Fig. 10. Observable islands in IEEE 14-bus system

different observable islands respectively. Bus 9 is chosen to install a PMU. Numerical observability analysis is then executed and the system is found to become observable. This implies that installing one PMU at bus 9 merges all of the five observable islands into one observable system. The logic is simple and easy to implement, provided that there are few observable islands and consequently few boundary buses.

C. Simulation Results

This section contains various simulation examples, which are carried out using the IEEE 14-bus, 30-bus, 57-bus and 118-bus systems. Network data for these systems is in public domain [21]. TOMLAB/MINLP and MILP [22] software package is used to solve the Integer Linear/Nonlinear Programming problem. Detailed system information and simulation results are given in the following sections.

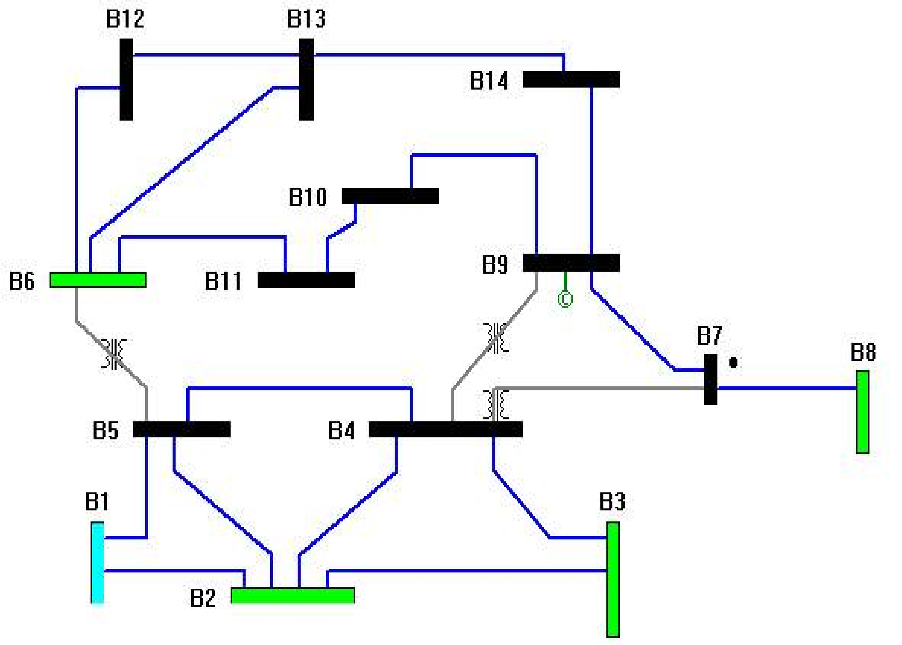


Fig. 11. IEEE 14-bus system

1. IEEE 14-bus system

IEEE 14-bus system is shown in Figure 11. The information of the system and zero injections are given in the Table XVIII.

Table XVIII. System information of IEEE 14-bus system

System	Num. of branches	Num. of zero injections	Zero injection buses
IEEE 14-bus	20	1	7

a. Case 1: Effect of considering zero injections

In this case, two sets of simulations are carried out on the IEEE 14-bus system, which initially have no flow measurements. Integer programming is used to solve the

Table XIX. Results for the 14-bus system without considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
4	2, 6, 7, 9	3	2, 6, 9	3	2, 6, 9

Table XX. Results for 14-bus system considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
9	1, 2, 3, 6, 7, 8, 9, 10, 13	7	1, 2, 4, 6, 9, 10, 13	7	1, 2, 4, 6, 9, 10, 13

optimal placement problem. Loss of PMUs is not considered in this case. In the first set of simulations, zero injection is simply ignored while in the second set, it is used as existing measurement. The two ways of handling zero injection, namely by forming nonlinear constraints and by applying a topology transformation to the zero injection bus and one of its neighbors are used for the second set. Comparative simulation results are shown in Table XIX. Having zero injections will reduce the number of required PMUs as can be seen from these results. In this case, the topology transformation method and the nonlinear constraints method present the same solution.

b. Case 2: Considering single PMU loss

In this case, simulations of case 1 are repeated by accounting for the loss of single PMUs. Primary and backup locations are found and combined results are shown in Table XX. When compared with those of Table XIX, the results show a marked increase in the required number of PMUs when loss of single PMUs is taken into account.

2. IEEE 30-bus system

IEEE 30-bus system is shown in Figure 12.

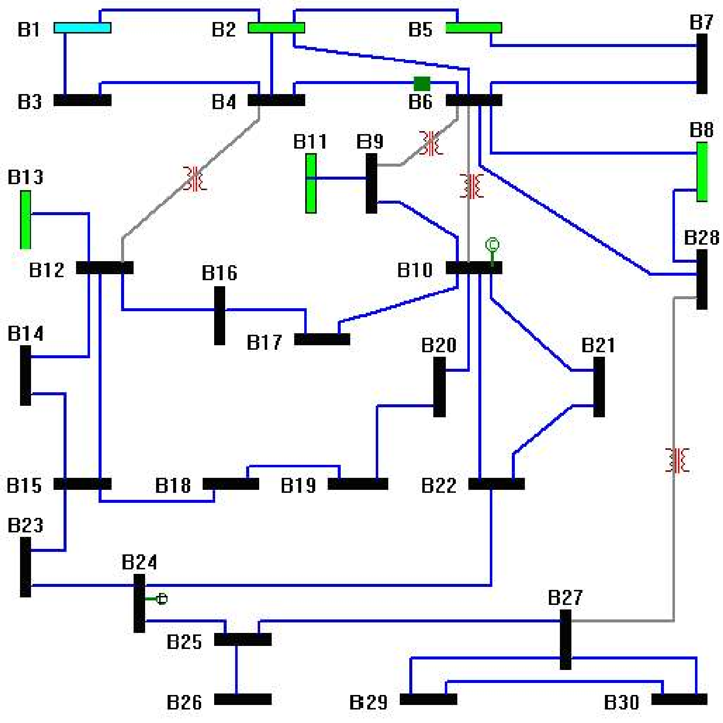


Fig. 12. IEEE 30-bus system

The Information of the system and zero injections are given in the Table XXI.

Table XXI. System information of IEEE 30-bus system

System	Num. of branches	Num. of zero injs.	Zero injection buses
IEEE 30-bus	41	5	6, 9, 11, 25, 28

a. Case 1: Effect of considering zero injection

In this case, Integer Programming method is used to solve the optimal PMU placement problem without considering the loss of single PMU. Simulations are carried out with and without considering zero injections. Results are given in Table XXII.

Table XXII. Results for the 30-bus system without considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
10	2, 4, 6, 9, 10, 12, 15, 18, 25, 27	7	3, 5, 10, 12, 18, 23, 27	8	2, 3, 6, 10, 12, 18, 23, 27

b. Case 2: Considering single PMU loss

In this case, single PMU loss is considered. Integer Programming is used, simulations are carried out with and without considering zero injections. Results are given in Table XXIII.

Table XXIII. Results for the 30-bus system considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
22	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 21, 23, 25, 26, 27, 29	17	2, 3, 4, 5, 6, 10, 12, 13, 15, 17, 18, 19, 21, 23, 24, 27, 29	18	1, 2, 3, 4, 7, 8, 9, 13, 14, 15, 16, 19, 20, 21, 23, 24, 28, 29

3. IEEE 57-bus system

IEEE 57-bus system is shown in Figure 13. The Information of the system and zero injections are given in the Table XXIV.

Table XXIV. System information of IEEE 57-bus system

System	Num. of branches	Num. of zero injections	Zero injection buses
IEEE 57-bus	78	15	4, 7, 11, 21, 22, 24, 26, 34, 36, 37, 39, 40, 45, 46, 48

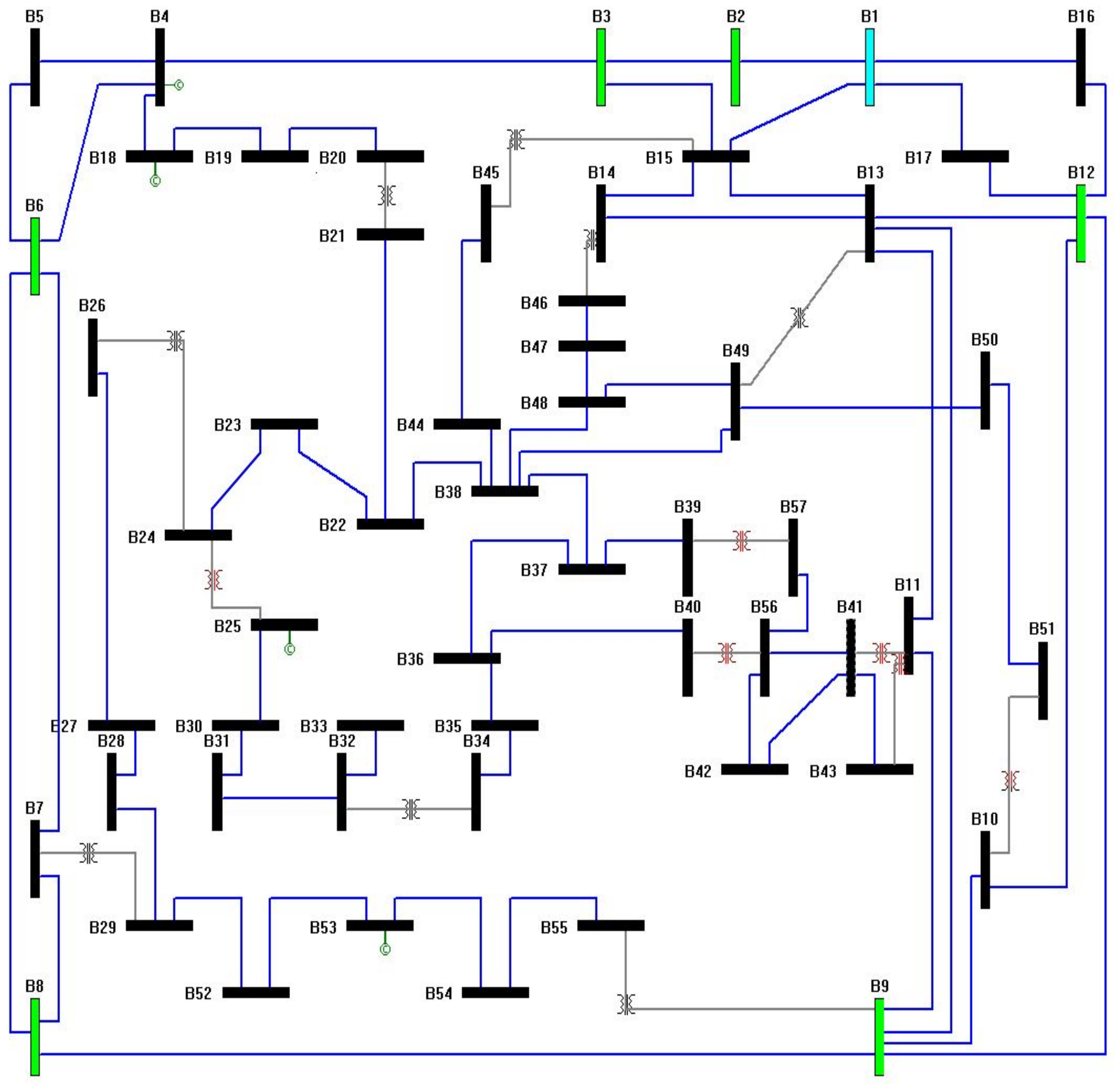


Fig. 13. IEEE 57-bus system

a. Case 1: Effect of considering zero injection

In this case, Integer Programming method is used to solve the optimal PMU placement problem without considering the loss of single PMU. Simulations are carried out with and without considering zero injections. Results are given in Table XXV.

Table XXV. Results for the 57-bus system without considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
17	1, 4, 7, 9, 15, 20, 24, 25, 27, 32, 36, 38, 39, 41, 46, 50, 53	13	1, 6, 9, 15, 20, 25, 27, 32, 38, 47, 50, 53, 56	12	1, 5, 9, 14, 15, 20, 25, 28, 32, 50, 53, 56

b. Case 2: Considering single PMU loss

In this case, single PMU loss is considered. Integer Programming is used, simulations are carried out with and without considering zero injections. Results are given in Table XXVI.

Table XXVI. Results for the 57-bus system considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
35	1, 2, 4, 6, 7, 9, 11, 12, 13, 15, 19, 20, 22, 24, 25, 26, 27, 29, 30, 32, 33, 34, 36, 37, 38, 39, 41, 44, 46, 47, 50, 51, 53, 54, 56	30	1, 2, 6, 7, 9, 10, 12, 14, 15, 18, 20, 21, 24, 25, 27, 29, 31, 32, 33, 34, 37, 38, 41, 44, 47, 49, 50, 53, 54, 56	26	1, 2, 4, 5, 9, 12, 13, 14, 15, 18, 20, 23, 25, 28, 29, 30, 32, 33, 35, 38, 41, 50, 51, 53, 54, 56

Table XXVII. System information of IEEE 118-bus system

System	Num. of branches	Num. of zero injections	Zero injection buses
IEEE 118-bus	179	10	5, 9, 30, 37, 38, 63, 64, 68, 71, 81

Table XXVIII. Results for the 118-bus system without considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
32	2, 5, 9, 11, 12, 17, 21, 24, 25, 28, 34, 37, 40, 45, 49, 52, 56, 62, 63, 68, 73, 75, 77, 80, 85, 86, 90, 94, 101, 105, 110, 114	29	2, 8, 11, 12, 15, 19, 21, 27, 31, 32, 34, 40, 45, 49, 52, 56, 62, 65, 72, 75, 77, 80, 85, 86, 90, 94, 101, 105, 110	28	2, 8, 11, 12, 17, 21, 25, 28, 33, 34, 40, 45, 49, 52, 56, 62, 72, 75, 77, 80, 85, 86, 90, 94, 101, 105, 110, 114

4. IEEE 118-bus system

IEEE 118-bus system is shown in Figure 14.

The Information of the system and zero injections are given in the Table XXVII.

a. Case 1: Effect of considering zero injection

In this case, Integer Programming method is used to solve the optimal PMU placement problem without considering the loss of single PMU. Simulations are carried out with and without considering zero injections. Results are given in Table XXVIII.

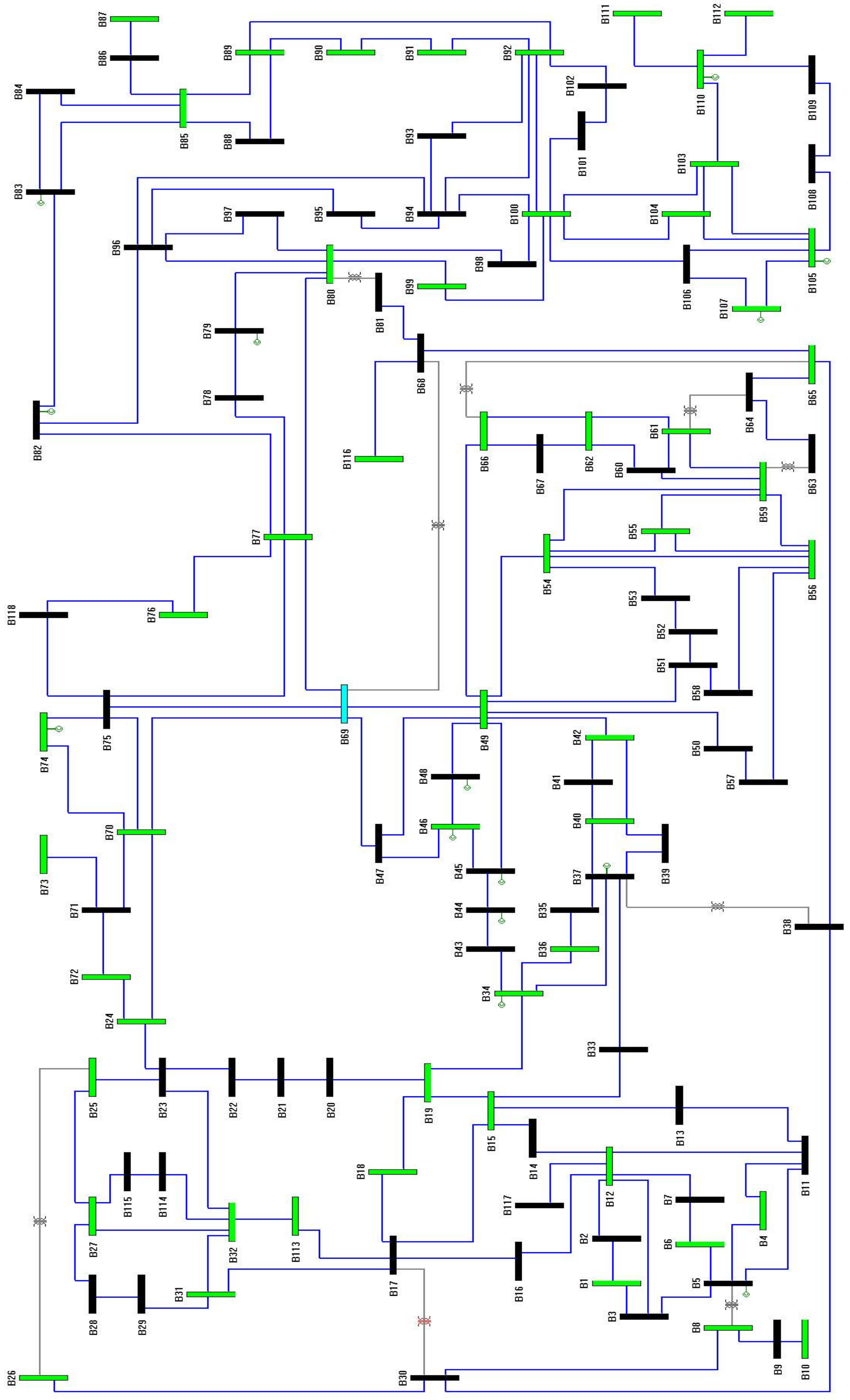


Fig. 14. IEEE 118-bus system

Table XXIX. Results for the 118-bus system considering single PMU loss

Ignore zero injections		Consider zero injections			
		Non-linear constraints		Topology transformation	
PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)	PMUs	Loc. (bus No.)
72	1, 2, 4, 5, 6, 9, 10, 11, 12, 15, 16, 17, 19, 21, 22, 24, 25, 27, 28, 29, 30, 32, 34, 35, 37, 39, 40, 41, 43, 45, 46, 49, 50, 51, 52, 54, 56, 59, 62, 63, 64, 66, 68, 70, 71, 73, 75, 76, 77, 78, 80, 81, 83, 85, 86, 87, 89, 90, 92, 94, 96, 100, 101, 105, 106, 108, 110, 111, 112, 114, 116, 117	65	1, 2, 6, 8, 9, 11, 12, 13, 14, 15, 17, 19, 20, 21, 23, 24, 27, 28, 31, 32, 34, 35, 37, 40, 41, 43, 45, 46, 49, 51, 52, 54, 56, 57, 59, 62, 65, 67, 68, 70, 72, 75, 76, 77, 78, 80, 83, 85, 86, 87, 89, 90, 92, 94, 96, 100, 101, 105, 106, 108, 110, 111, 112, 114, 117	65	1, 2, 5, 6, 8, 10, 11, 12, 15, 16, 17, 19, 21, 22, 24, 25, 27, 28, 31, 32, 33, 34, 35, 39, 40, 41, 43, 45, 46, 49, 50, 51, 52, 53, 56, 59, 62, 65, 66, 70, 72, 75, 76, 77, 78, 80, 83, 85, 86, 87, 89, 90, 92, 94, 96, 100, 101, 105, 106, 108, 110, 111, 112, 114, 117

b. Case 2: Considering single PMU loss

In this case, single PMU loss is considered. Integer Programming is used, simulations are carried out with and without considering zero injections. Results are given in Table XXIX.

c. Case 3: System having several conventional measurements

This case illustrates the application of the topology-based method to place PMUs in the IEEE 118-bus system. Several conventional flow and injection measurements are introduced and several observable islands are formed. The list of flow and injection measurements is given in Table XXX. Zero injections shown in Table XXVII are also considered and treated the same as injection measurements. Applying the topology-based method to merge these observable islands yields the number of PMUs shown in Table XXXI. Loss of single PMUs is not considered in this case.

D. Conclusions and Future Work

In this chapter, two practical methods for determining optimal locations for PMUs are developed and applied to IEEE 14-bus, 30-bus, 57-bus and 118-bus systems.

Placement of PMUs can be carried out using different criteria depending on the objective of the investigator. In this research, the main focus is on the state estimation function and therefore the objective is to make the entire system observable by optimal placement of PMUs. Various scenarios are considered where the system is first assumed to be observed by PMUs only. While this appears impractical today, it may very well be the case in a few years when these devices become standard equipment at substations. Next, the placement problem is considered for a system with existing measurements, some of which may be PMUs. Case studies which are

Table XXX. Measurements information for IEEE 118-bus system

Flow measurements		Injection measurements		Num. of observable islands
Num.	Locations	Num.	Locations	
49	6-7, 5-6, 1-3, 3-12, 3-5, 8-30, 8-5, 8-9, 26-25, 25-27, 29-31, 28-29, 23-32, 32-114, 27-32, 70-74, 74-75, 47-69, 46-47, 82-83, 83-84, 93-94, 92-94, 94-100, 99-100, 98-100, 106-107, 105-107, 51-52, 51-58, 55-59, 54-59, 59-60, 15-19, 19-20, 19-34, 12-16, 12-117, 35-37, 34-37, 35-36, 38-37, 43-44, 49-50, 65-68, 68-116, 68-69, 110-111, 110-112	29	5, 9, 12, 19, 21, 27, 28, 30, 32, 37, 38, 41, 44, 47, 50, 53, 59, 62, 63, 64, 68, 71, 81, 83, 86, 94, 96, 108, 110	58

Table XXXI. Results for 118-bus system considering conventional measurements

Num. of PMUs	19
Loc.(bus No.)	2, 11, 17, 21, 24, 40, 49, 56, 62, 71, 77, 80, 86, 89, 91, 100, 102, 108, 118

carried out on IEEE test systems indicate that strategically placing PMUs at roughly one third of the system buses, the entire system can be made observable with only PMUs. Furthermore, zero injections, which can be considered free measurements, can significantly reduce the required number of PMUs for a given system.

PMU placement problem does not have a unique solution. Depending upon the starting point, the developed optimization scheme may yield different sets of optimal solutions, each one providing the same minimum number of PMUs but at different locations. On the other hand, it is not unusual to have additional considerations apart from strict observability criterion, when deciding on the location of PMUs. These considerations can be taken into account by appropriately modifying the optimization problem which is formulated in this research. This can be done as an extension to this research in the future. One of the important functions of state estimators is to detect and eliminate bad measurements in the system. Bad data processing is strongly dependent upon the measurement redundancy as well as accuracy of the measurements used. Even for fully observable systems, strategic placement of few PMUs can significantly improve bad data detection and identification capability. This aspect of PMU placement can also be investigated in the future so that the operation of the existing state estimators can be improved via PMU placement.

CHAPTER IV

3D GUI FOR POWER SYSTEM VISUALIZATION

Visualization of power system operation state has become of interest for the system operators as well as planning engineers alike. A good visualization method can aid system operators to gain a better insight into the system state and identify system violations in a quick and intuitive manner. This chapter presents a user-friendly 3D visualization tool. It focuses on two application functions, namely the power flow and the state estimation. One of the essential goals of the envisioned tool is to aid system operators by providing images of the system with easily identifiable characteristics related to violations of various operating limits. Effectiveness of the tool is illustrated via different scenarios, which are created using the IEEE 118-bus system as an example.

A. Introduction

Power system operators have the difficult job of maintaining continuity of service to power customers without violating any operating limits as the operating conditions change during the daily operation. This task requires close monitoring of the system conditions, in particular those, which might cause temporary or permanent service interruption, possible damage to power equipment as well as the customers.

Power system operators are challenged by the overwhelming amount of data that are transmitted to the control centers for their use. A good visualization method can aid system operators to gain a better insight into the system state and identify system violations in a quick and intuitive manner.

Earlier work on this topic showed the effectiveness of providing better communication between massive amounts of data and the users [23, 24, 25, 26, 27]. In [28], programs to visualize line flow and bus data are presented. In [29], bad measurements used in state estimation are displayed in an easily identifiable way.

This chapter describes a flexible software tool, which takes advantage of different visualization techniques such as animation, color contouring and 3D visualization in order to extract and highlight certain user specified information from the data acquired by the Supervisory Control and Data Acquisition (SCADA) system. Two essential and critical applications in today's Energy Management Systems (EMS) namely the state estimation and power flow analysis are used as examples. In power flow analysis mode, animation and bus contouring technique is used for visualizing power flows and bus voltages. In state estimation mode, color contour is used to visualize different observable islands and bad measurements, which are identified through the largest normalized residues test. Effectiveness of the tool is illustrated via different scenarios, which are created using the IEEE 118-bus system as an example.

The developed program uses 3D visualization and animation techniques to interpret system information. This window-based interface is an upgraded version of the existing software package, Power Education Toolbox (PET), which is previously developed in Texas A&M University for educational purposes [30]. The new interface is developed in C++ using Fast Light Tool Kit (FLTK). OpenGL technique is used to create 3D visualization. The following sections present a detailed description of the program. Different scenarios including unobservable cases with several observable islands, measurement systems containing bad data and operating conditions with heavily loaded buses are created using the IEEE 118-bus system.

The chapter is organized as below.

- Section A is the introduction.
- Section B is the description of the program.

- Section C uses different scenarios to demonstrate that proper visualization method can provide a better understanding of the operation of power system.
- Section D explains in details about the structures of the main classes and the data exchange between the GUI and the FORTRAN subroutines.
- Section E is conclusion.

B. Program Description

This graphical user interface contains two windows3, an edit window and a view window. Object Oriented Programming (OOP) technique is used to allow the user to interact with the objects, which represents different components in the power system.

In edit window, the top view of a power system is shown as a one-line diagram. It allows the user to edit and build the power system by adding/moving/deleting buses and lines. View window shows 3D visualization of power system where the user can choose different view angles. The user can easily pan or zoom in/out in these two windows to locate the point of interest in the system.

Figure 15 shows an example of the interface. The left window is edit window and the right window is view window. The towers in the view window represent buses in power system while wavy lines represent the transmission lines. In this example, a map of the state of Texas is used to show the geographic information of a real power system. Note that a color map can instead be loaded to show the system information during simulation.

Once a system is opened, all the associated information is read by the corresponding variables for running the supported applications. Each power system case has two associated data files. One file specifies the 3D coordinates of all the buses while the other file defines power system information by using IEEE common data

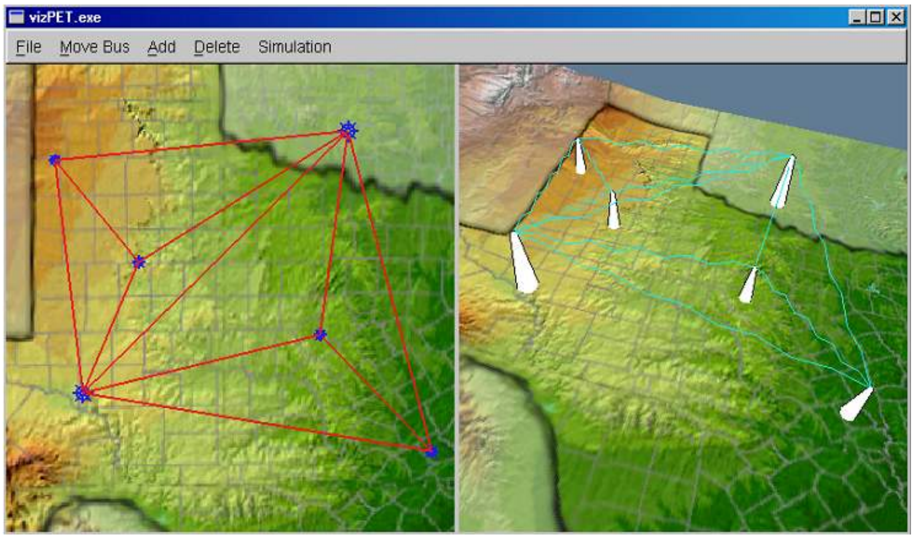


Fig. 15. Graphical user interface

format [21]. Once a power system case is opened, the system information is read and 2D/3D visualizations are displayed. User can use the main menu to choose between options of editing the system or running simulations for applications such as power flow analysis and state estimation.

1. Power flow analysis

During system operation, three attributes are required to be visualized simultaneously in the system diagrams. These are:

- Magnitudes of bus voltages: There are lower and upper limits, which should not be violated in order to avoid voltage instability and equipment safety violations.
- Phase angles of bus voltages: These values should be monitored with respect to each other among network buses so that wide separation of angles are detected before they lead to power angle instability.
- Real power flow along lines: Direction and amount of power flow along lines

must be monitored in order to avoid violating thermal and stability limits. They also provide information about existing margins of load ability for a given line.

In this software, magnitude of a bus voltage is visualized via coloring of the terrain immediately around the transmission tower representing that bus. Phase angles of bus voltages are represented as heights (altitudes) of the terrain at the associated transmission tower. Hence, the resulting terrain reflects the variation of phase angle in terms of peaks and valleys in the system scene. Real power is known to follow closely the variation of the phase angles; hence such representation provides information about the distribution of power flows in the entire system. Thus, two kinds of information can be displayed simultaneously. Animated waves traveling along transmission lines are used to visualize the power flows, whose directions are indicated by the direction of the traveling waves. Furthermore, numerical values of these visualized quantities such as voltage magnitudes, phase angles and power flows can be displayed using a dialog box by simply clicking on the object of interest. Also, the objects in the system can be hidden upon user request if only the terrain pattern is desired to be visualized.

2. State estimation

State estimation is a function, which acts like a filter to the system measurements and provides the best estimate of the system state using the available measurement system. Usually, two types of information are of crucial interest to the operator when executing this application: network observability and measurement errors.

Network observability refers to the ability of the estimator to provide a network-wide solution using the existing set of measurements. If the measurements were insufficient, then the operator would like to identify the observable islands, within which power flows could still be monitored. On the other hand, measurement errors creep in without warning and bias the estimated state. In order to avoid this, a post estimation test is typically carried out and suspected bad data are identified. Observable islands and bad data are both visualized via color contours on the terrain. The details are presented in the next section.

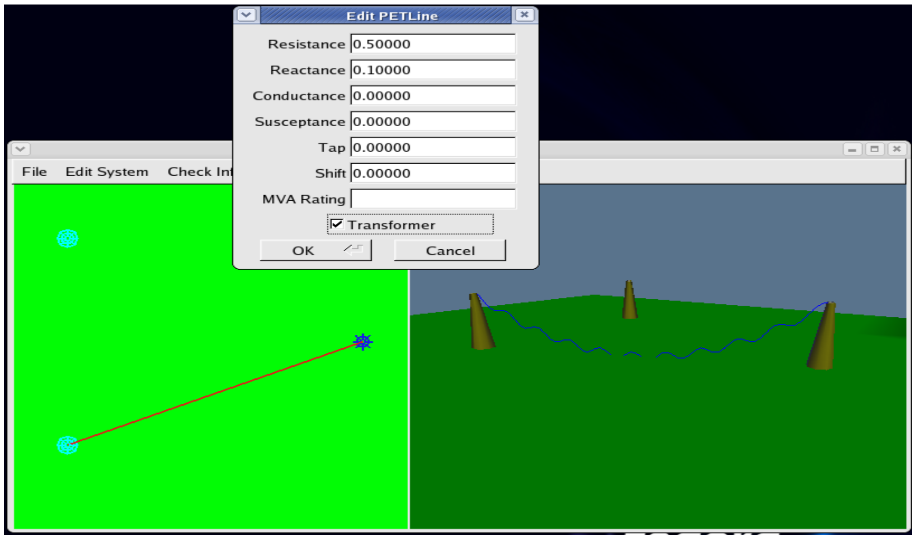
C. Cases Studies

In this section, different scenarios are created using IEEE 118-bus system to illustrate the benefits of using proper visualization methods in monitoring power system operation.

1. Create/edit a system

Users can interactively build new a power system from scratch in the edit window. To add a bus in the system, the user can choose “Add a Bus” option from the main menu and click on the desired location in the edit window, a tower will then appear in that position and a dialog box pops up where the parameters associated with the bus can be entered. To add a new line, simply choose “Add a Line” option from the main menu and click on two towers as from-bus and to-bus, a waved line representing the transmission line will be added in between and a dialog box appears where the user can input line parameters. The program provides flexibility of moving the existing bus while all the incident lines move with it. This makes it possible to neatly create even very large scaled power system diagrams together with the panning and zooming in/out features.

Figure 16 illustrates creation of a new line by the user.



Resistance 0.50000
Reactance 0.10000
Conductance 0.00000
Susceptance 0.00000
Tap 0.00000
Shift 0.00000
MVA Rating
[✓ Transformer]

Fig. 16. Creation of a new line by the user

2. Observability analysis

State estimator can identify observable islands and label the branches that connect these islands as unobservable branches by analyzing the existing measurements set.

A method referred to as color contouring is used to visualize different observable islands. This is done as described below.

For each and every observable island, all the buses that belong to that island are assigned an identical island number. Note that a bus can be assigned one and only one island number since islands cannot overlap. Then, each and every vertex in the grid that defines the terrain is assigned a unique color ID according to the island number of its nearest bus. Hence, a map representing observable islands can be created by color mapping. In the resulting map, all unobservable branches are colored in red and observable branches in blue. This allows the operator to quickly identify the islands as well as candidate locations where additional meters can be placed to merge these islands.

Figure 17 shows the observability analysis of IEEE 118-bus system, where four

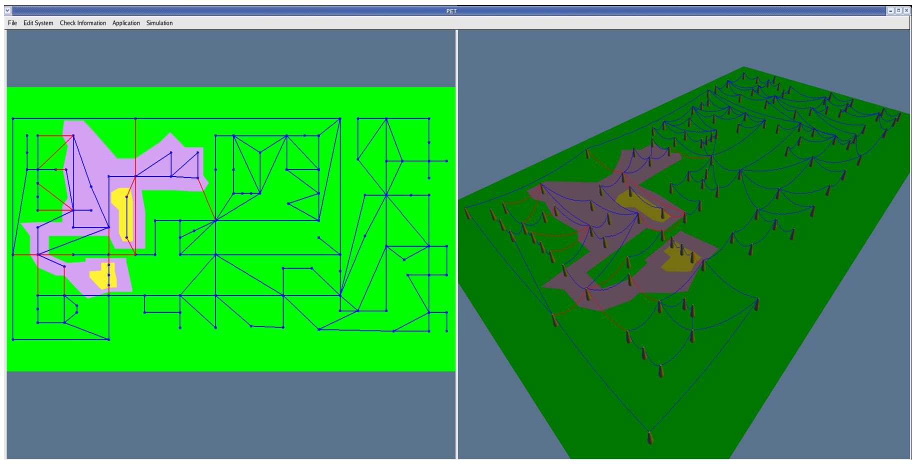


Fig. 17. Observable islands in the IEEE 118-bus system

observable islands can easily be identified. The unobservable branches are shown in red color.

3. Bad data detection/identification

It is customary to perform post-estimation tests to detect and identify bad data in the measurement set. Success of these tests in detecting and identifying bad data depends among other factors on the measurement redundancy. A commonly used test is the largest normalized residue test where the normalized residue of each measurement is calculated and the largest one is identified as bad data provided it exceeds a statistical threshold. In order to quickly determine the region that is most affected by the identified bad data, a color contour is used. This contour is created as described below.

Every system bus is assigned a value, which is determined as the largest normalized residue among all measurements measured at that bus. Bus injections are measured at the associated buses and line flows are measured at their sending-end bus. If a bus has no measurement associate with it, the value is set to be zero. Because an error introduced by a bad measurement can spread in the system and affect others, not all the measurements whose normalized residues are larger than the threshold are bad. In this program, only the measurements whose normalized residues are larger than the threshold and among the largest five are considered. These measurements and their associated buses are marked as “bad” and needed to be identified. In order to create a spatially continuous contour, a virtual value is assigned to each and every vertex in the terrain. The virtual value of the vertex related to a bus is calculated as the scaled value of that bus. The multiplier is calculated in a way such that all the ”bad” buses appear in red color and other buses appear in blue when applying texture mapping. All other terrain vertices, which are not related to a bus, are assigned average values based on the already assigned values to their closest vertices. This way, one mapping using blue color for the lower values and red color for higher values can create a color map.

The advantage of color contour method is obvious. For a large scaled power system, in which the measurements may not be visible when showing the entire system map, the user can easily identify the red region in the color map and then locate the bad measurement by panning and zooming into the identified suspect area.

To show the bad measurements, all those measurements identified as “bad” are rendered in red color. This way, bad data can be easily identified when the users notice an abnormal in the system and want to take a close look.

Figure 18 shows bad data analysis results of IEEE 118-bus system. A red area, which indicates existence of bad data in the system, can easily be noticed. In this case, the measurements set is hide to render a neat scene. Figure 19 shows a close-up view of the suspected area. Measurements set is shown this time. The flow measurement in red is identified as bad data. And by clicking it, a dialog box pops up showing the numerical results from the bad data test associated with this specific measurement.

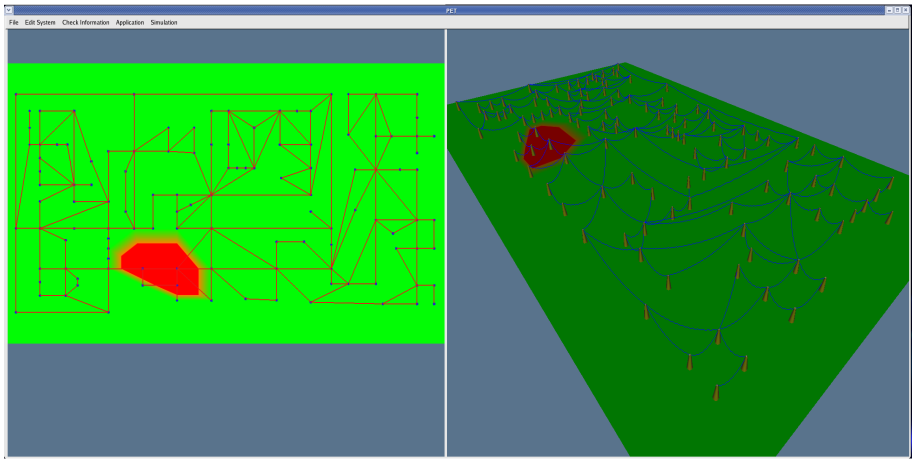
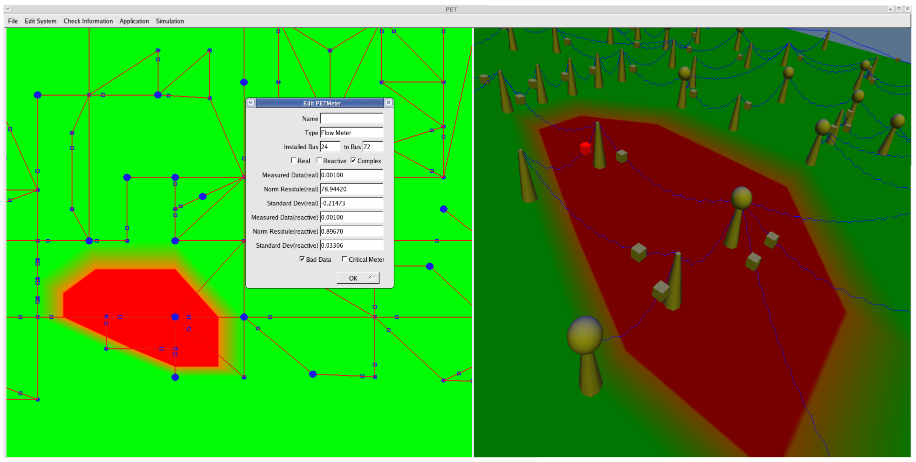


Fig. 18. Bad data in the IEEE 118-bus system



Name	
Type	Flow Meter
Installed Bus	24
to Bus	72
☐ Real ☐ Reactive ☑ Complex	
Measured Data(real)	0.00100
Norm Residule(real)	78.94420
Standard Dev(real)	-0.21473
Measured Data(reactive)	0.00100
Norm Residule(reactive)	0.89670
Standard Dev(reactive)	0.03306
☑ Bad Data ☐ Critical Meter	

Fig. 19. Detailed numerical results of bad data test

4. Power flow analysis

Power flow analysis calculates the state variables, which are voltage magnitudes and phase angles for all the buses in the entire power system. Two kinds of visualization methods are used to visualize these two bus-related data. Color contouring is used for visualizing voltage magnitude while the varying height of the underneath terrain represents the phase angles.

Color map ranges from red to blue, which represents higher voltage and lower voltage. The color map provides an overview of voltage profile of entire power system and helps the user to identify voltage violations at a glance. The varying height of the vertices creates a mountain shaped terrain with peaks and valleys. Note that real power is more related to phase angles, it typically flows from the bus having relatively higher phase angle to the bus with a smaller phase angle. So, the mountain shaped terrain provides the user a quick overview of the distribution of real power flows in the power system: real power flows from the higher tower to the lower tower along the transmission lines. This is analogous to water flow in a similar terrain with water pipes. Animation is also used to represent the direction of the line flows. Direction of the traveling wave matches the direction of the real power flow in the transmission line. Numerical values for bus voltages and line flows can be easily recovered via a dialog box by simply clicking on the object.

Figure 20 shows the 3D visualization of estimated state variables for IEEE 1118-bus system. The operator can conveniently focus on the red area, which indicates a voltage violation. Further clicking on the bus produces a dialog box as shown in Figure 21 displaying the numerical values associated with this bus.

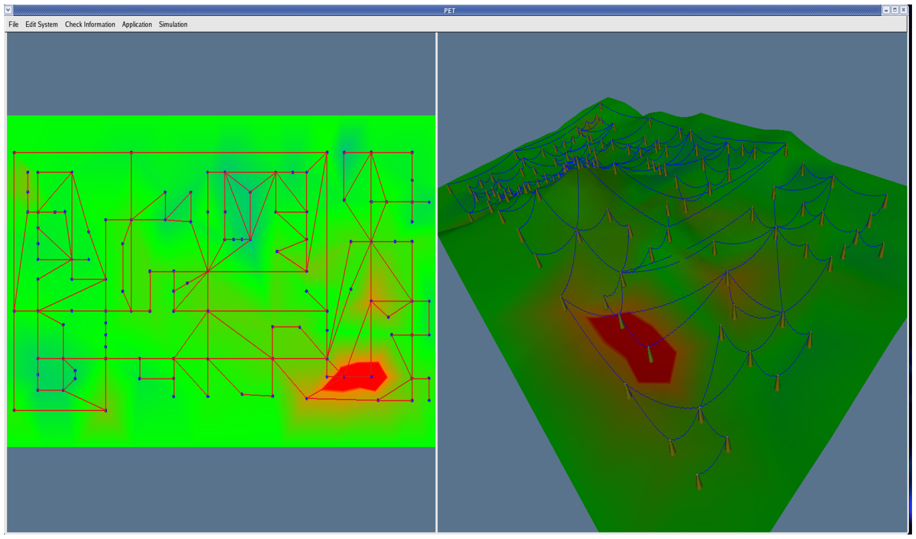
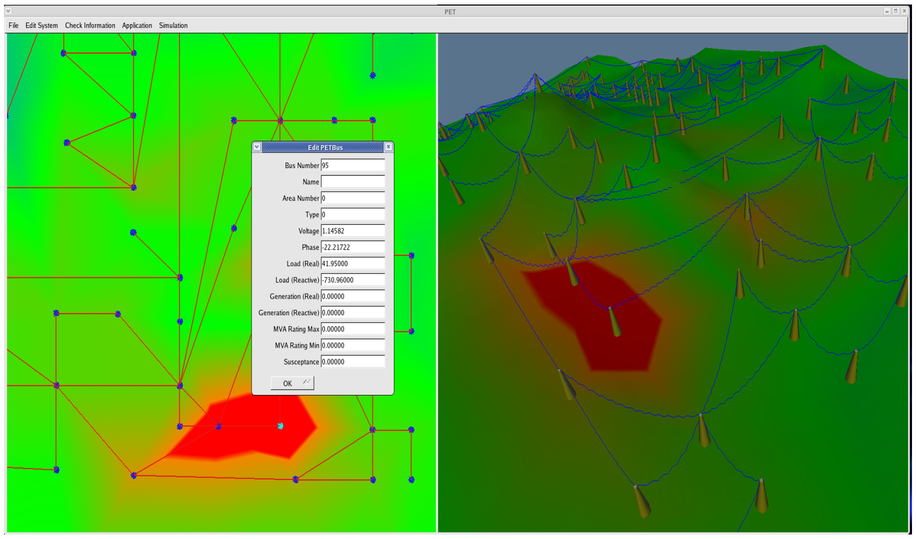


Fig. 20. Power flow analysis of IEEE 118-bus system



Bus Number	95
Name	
Area Number	0
Type	0
Voltage	1.14582
Phase	-22.21722
Load (Real)	41.95000
Load (Reactive)	-730.96000
Generation (Real)	0.00000
Generation (Reactive)	0.00000
MVA Rating Max	0.00000
MVA Rating Min	0.00000
Susceptance	0.00000

Fig. 21. Detailed bus information

D. Software Design Overview

This software provides a windows-based graphical user interface linked with the power system applications. This 3D GUI is developed in C++ programming language under Linux Fedora by using FLTK version 1.1.6, while the linked application are developed as subroutines in FORTRAN 90. The following sections explain in details about the data structure of the software, the main classes, their functions and the data exchange between the GUI and the FORTRAN subroutines.

1. Main classes and their functions

This program is developed using FLTK, which provides a group of classes for construct framework and components used for a windows-based GUI. Furthermore, it also supports 3D graphical via OpenGL. All the objects used in this program such as terrain, tower, meter etc., are built from the base class OBJObject, which is already developed by the visualization lab in Texas A&M University. The introduction of FLTK, OpenGL and the base classes used in this program are given in Appendix A, B and C.

The edit window and view window in this interface are represented by two global variables, *petewin* and *petvwin* respectively. Each of them contains a pointer to the global variable *petscene*, which contains the information of a power system and shows its behavior. *petscene* is composed of two main member variables *petsystem* and *terrain*. *petsystem* contains all the information of the buses and lines of a power system while *terrain* represents the terrain beneath the system. The data structure of the GUI is shown in Figure 22.

The descriptions of main classes used in this program and their functions are given in the following sections.

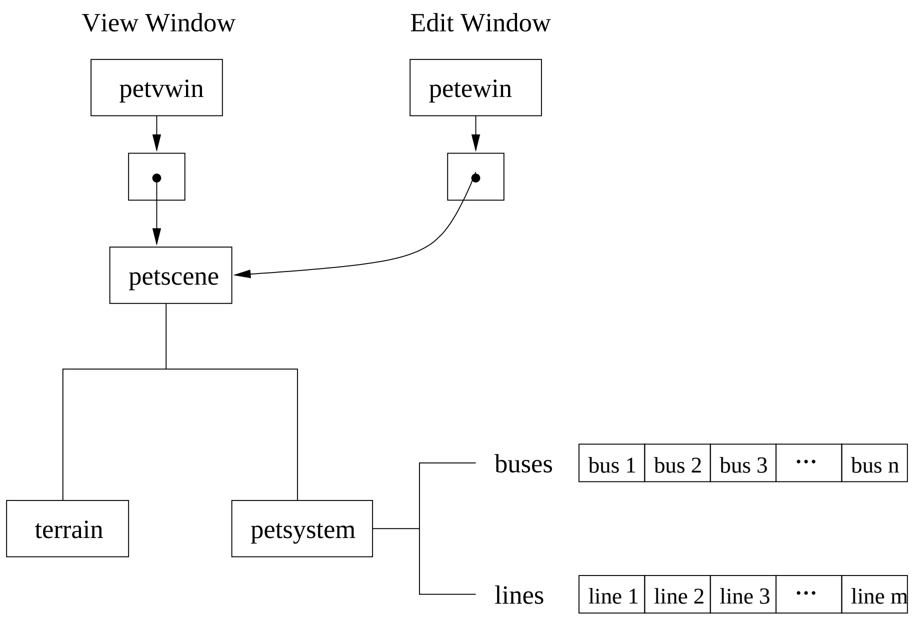


Fig. 22. The data structure of PET GUI

a. PETViewWindow class

There are two classes for the view window and the edit window, PETViewWindow and PETEditWindow. Both of them are subclasses from Fl_Gl_Window class (provided by FLTK), which supports OpenGL commands.

PETViewWindow class contains the functions to render the entire system in 3D. It has its own events handlers to render the system and redraw the window when the user zooms in/out, rotates and translates the scene.

- PETViewWindow(int x, int y, int w, int h, const char * l = 0) creates a view window. This window is derided from Fl_Gl_Window class.

- void draw(void) prepares the background color and the transformation matrix for view port, sets up texture, and draw the system (petscene) in 3D.

- int handle(int event) handles all the events from keyboard and mouse.

b. PETEditWindow class

PETEditWindow contains all the methods needed to create and edit a power system. It renders the top view of the system. PETEditWindow has its own redraw function and events handlers to deal with the messages and redraw the whole window.

- PETEditWindow(int x, int y, int w, int h, const char * l = 0) creates an edit window, which is derided from Fl_Gl_Window class.
- void draw(void) prepares the background and the transformation matrix, and draws the top view of the system.
- selectBus(int mx, int my) returns a pointer to the selected bus.
- selectLine(int mx, int my) returns a pointer to the selected line.
- int handle(int event) dispatches the events to different event handlers for editing the system and zooming in/out or translating the scene.

c. Classes for the components in the system

Several classes are created to represent the components in a power system. Theses classes are:

- PETBus: It contains all the functions and variables associate with a bus. It has a member variable *object* of PETGraphicBus class.
- PETGraphicBus: It contains the geographic information of a bus and all the functions needed to render and redraw a bus object when information is updated.
- PETLine: It contains all the functions and variables associate with a transmission line. It has a member variable *object* of PETGraphincLine class.

- PETGraphicLine: It contains the functions for rendering and redrawing a line object. The lines are represented by waved lines, and the power flows in the system is shown by animation of moving waves along the line.
- PETKnob: PET Know is similar to PETBus. It contains a member variable *object* of PETGraphicKnow class, which renders a knob in view and edit windows. Knob contains no system information. It is used to adjust long transmission lines and to avoid the crossing of two lines.
- PETMeter: It contains all the functions and variables associate with a measurement in power system. It has a member variable *object* of PETGraphicMeter class.
- PETGraphicMeter: It contains all the functions needed to render and redraw a measurement object.

d. PETScene class

PETScene has two member variables, *petsystem* and *terrain*, which are instances from PETSystem and PETTerrain classes respectively. PETScene takes care of reading network information of the system and the geographic positions of all the buses from the input data files, creating a corresponding terrain and rendering a system in 3D.

e. PETSystem class

PETSystem represents a real power system, which contains arrays of buses, lines and measurements. It manipulates all the system information and takes care of data exchanges between the GUI and the FORTRAN subroutines.

f. PETTerrain class

PETTerrain is derived from OBJObject class. Terrain is automatically generated in the program according to the geographic positions of all the buses in the system, and its color and shape changes during system operation reflecting different system information.

The steps to generate a terrain are described as follows.

- First, the positions of all the buses are recorded as generating points and a Delaunay triangulation function is called to partition a plane into a set of triangles, which have the properties that the out circle of every triangle does not contain any other generating points.

- Once the system data is updated, the y-positions of the vertices in the terrain can be changed to reflect the system information, which is customized by the user.

- Doo-Sabin [31] subdivision algorithm is then applied to smooth the surface of the terrain.

- A texture ID is assigned to each and every vertex in the terrain. For the generating points, the texture ID is calculated according to the voltage magnitudes of the corresponding buses, while for the new generated vertices during the subdivision procedure, the texture id is the average of the texture IDs of all the surrounding generating points.

- A customized image is loaded and mapped on to the terrain. Once the global light placed on top of the scene is lightened, a vivid 3D image of terrain is presented, which contains the operation information of the power system.

In this program, a one-dimension image is used which has red color on the left side, indicating one type of fault in the system such as over voltage, and blue color on the right side, indicating another kind of fault such as lower voltage. The color gradually changes from red to green then to blue in the middle, representing the continuous data of the system. The texture map can be customized and reloaded by the user. A two-dimension map can also be used to represent the geographic system information as shown in Figure 15.

2. Interface between the FORTRAN subroutines and the GUI

The subroutine functions are written in FORTRAN and the GUI calls external object functions complied using FORTRAN 90 compiler whenever the user chooses to run applications.

The structure *petblock* is defined in this program. It is an equivalent structure of the common block defined in file "defblk.inc", which contains variables shared between the user interface and the FORTRAN subroutines. The FORTRAN subroutines are declared as external functions. For example, the subroutine for power flow analysis is declared as

```
extern "C" extern void fdpf_(void);
```

When the user select "Power Flow Analysis" option from the menu, the function PETSystem::updateDPF(void) is called. First, it exchanges data to get all the needed information for buses and lines into the global variable *petblock_*, and then the FORTRAN subroutine fdpf_(void) is called to do the calculations. After that, another data exchange is done to update the bus and line data with the newly computed value in *petblock_*.

E. Conclusion

This chapter describes a user-friendly GUI, which is developed as an aid to the system operator. It allows the user to construct the system diagram for a new power system or to edit an existing one for desired modifications. 3D animation and color contouring techniques are used to visualize the results of two application functions, namely the state estimation and power system analysis. Examples of different cases are created to illustrate the performance of the developed software.

CHAPTER V

CONCLUSION

This dissertation focuses on three topics related to the optimal monitoring and visualization of steady state power system operation.

In Chapter II, the detailed state steady model of UPFC is studied and a new state estimation algorithm is implemented to integrate the UPFC model into the estimation applications. The developed program has dual purpose. it can be used to estimate the system state variables as well as the controller parameters of UPFC devices and also to determine the required settings of UPFC control parameters in order to deliver a desired power flow along a transmission line. Several computational issues are discussed and the effectiveness of this algorithm is demonstrated by simulation results on IEEE test systems.

The optimal PMU placement problem is to find the strategy locations of PMUs to present a fully observable system using minimum number of PMUs. Chapter III proposes two algorithms, a numerical method and a topology-based algorithm, to solve this problem.

The numerical is an integer-programming based optimization problem. It is effective in systematically placing PMUs in a system where there are very few existing measurements. In order to guard against unexpected failures of PMUs, this method is extended to account for single PMU loss.

The concept of topology-based method is to merge the existing observable islands by placing extra PMUs at strategic boundary buses. It is more applicable for a system, which has lots of measurements forming several observable islands.

Chapter IV describes a user-friendly 3D graphical interface for power system analysis. This GUI supprots two essential and critical applications on today's energy management system (EMS), power flow analysis and state estimation.

This program adopts Object-Oriented Programming (OOP) concepts, so that the user can interact with the objects in the system. All the components in the system can be easily replaced by reloading the corespondent objects designed by the user, which makes the program reusable. OpenGL technique is used in developing this interface to create 3D visualization of power system operation which can reveal several kinds of system information simultaneously. The program also provides easily identifiable images of violations of operation limits. Furthermore, its modular design also allows easy addition of new applications.

A. Summary of Contributions

The main contributions of this dissertation are listed as below.

1. An algorithm of state estimation for power system with embedded FACTS devices is implemented.
2. Two algorithms, a integer-programming based numerical method and a topology-based method, are developed to solve optimal PMU placement problem.
3. A novel 3D graphical user interface is developed to visualize the operation of electrical power system.

B. Future Work

There are still room for further work in the following directions.

1. Further research on state estimation of system with PMUs can be done in the future. A state estimator integrating PMUs can be developed and the issue of redundancy and bad data detection/identification should be addressed.

2. Upgrade the 3D graphical user interface. Extend it by adding more functions such as optimal power flow and transient stability.

REFERENCES

[1] F.C. Schweppe, J. Wildes, and D.B. Rom, "Power system static-state estimation, parts I, II and III," *IEEE Transactions on Power Apparatus and Systems*, vol. 89, pp. 120–135, January 1770.

[2] N.G. Hingorani, "High power electronics and flexible AC transmission system," *IEEE Power Engineering Review*, vol. 8, no. 7, pp. 3–4, July 1988.

[3] K.A. Clements, G.W. Woodzell, and R.C. Burchett, "A new method for solving equality-constrained power system static-state estimation," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1260–1266, November 1990.

[4] L. Gyugyi, C.D. Schauder, S.L. Williams, T.R. Rietman, D.R. Torgerson, and A. Edris, "The unified power flow controller: A new approach to power transmission control," *IEEE Transactions on Power Delivery*, vol. 10, no. 2, pp. 1085–1097, April 1995.

[5] A. Nabavi-Niaki and M.R. Iravani, "Steady-state and dynamic models of unified power flow controller for power system studies," *IEEE Transactions on Power Systems*, vol. 11, no. 4, pp. 1937–1943, November 1996.

[6] K.A. McShane, C.L. Monma, and D.F. Shanno, "An implementation of a primal-dual interior point method for linear programming," *ORSA Journal on Computing*, vol. 1, pp. 70–83, Spring 1989.

[7] K.A. Clements, P.W. Davis, and K.D. Frey, "Treatment of inequality constraints in power system state estimation," *IEEE Transactions on Power Systems*, vol. 10, no. 2, pp. 567–574, May 1995.

[8] G.L. Torres and V.H. Quintana, "On a nonlinear multiple-centrality-corrections interior-point method for optimal power flow," *IEEE Transactions on Power Systems*, vol. 16, no. 2, pp. 222–228, May 2001.

[9] S. Granville, "Optimal reactive dispatch through interior point methods," *IEEE Transactions on Power Systems*, vol. 9, no. 1, pp. 136–146, February 1994.

[10] H. Singh, F.L. Alvarado, and W-H.E. Liu, "Constrained LAV state estimation using penalty functions," *IEEE Transactions on Power Systems*, vol. 12, no. 1, pp. 383–388, February 1997.

[11] F.F. Wu, W-H.E. Liu, L. Holten, A. Gjelsvik, and S. Aam, "Observability analysis and bad data processing for state estimation using Hachtel's augmented matrix method," *IEEE Transactions on Power Systems*, vol. 3, no. 2, pp. 604–611, May 1988.

[12] A. Monticelli and F.F. Wu, "Network observability: Theory," *IEEE Transactions on Power Apparatus and Systems*, vol. 104, no. 5, pp. 1042–1048, May 1985.

[13] G. Peters and J.H. Wilkinson, "The least squares problem and pseudo-inverses," *The Computer Journal*, vol. 13, no. 3, pp. 309–316, August 1970.

[14] A. Monticellli and A. Garcia, "Reliable bad data processing for real-time state estimation," *IEEE Transactions on Power Apparatus and Systems*, vol. 102, no. 5, pp. 1126–1139, August 1983.

[15] A.G. Phadke, "Synchronized phasor measurements in power systems," *IEEE Computer Applications in Power*, vol. 6, no. 2, pp. 10–15, April 1993.

[16] A.G. Phadke, J.S. Thorp, and K.J. Karimi, "State estimation with phasor measurements," *IEEE Transactions on Power Systems*, vol. 1, no. 1, pp. 233–241, February 1986.

[17] T.L. Baldwin, L. Mili, M.B. Boisen, and R. Adapa, "Power system observability with minimal phasor measurement placement," *IEEE Transactions on Power Systems*, vol. 8, no. 2, pp. 707–715, May 1993.

[18] B. Xu and A. Abur, "Observability analysis and measurement placement for systems with PMUs," in *Power Systems Conference and Exposition*, New York, October 2004.

[19] B. Xu, Y.J. Yoon, and A. Abur, "Optimal placement and utilization of phasor measurements for state estimation," in 15th *Power Systems Computation Conference*, Liège, Belgium, August 2005.

[20] A. Abur and A.G. Expósito, *Power System State Estimation: Theory and Implementation*, Marcel Dekker, Inc., New York, NY, 1st edition, 2004.

[21] Power Systems Test Case Archive,
http://www.ee.washington.edu/research/pstca/.

[22] The Tomlab Optimization Environment,
http://tomlab.biz/.

[23] P.M. Mahadev and R.D. Christie, "Minimizing user interaction in energy management systems: Task adaptive visualization," *IEEE Transactions on Power Systems*, vol. 11, no. 3, pp. 1607–1612, August 1996.

[24] P.R. D'Amour and W.R. Block, "Modern user interface revolutionizes supervisory systems," *IEEE Computer Applications in Power*, vol. 7, no. 1, pp. 34–39,

January 1994.

[25] G.P. de Azevedo, C.S. de Souza, and B. Feijo, “Enhancing the human-computer interface of power system applications,” *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 646–653, May 1996.

[26] T.J. Overbye, P.W. Sauer, C.M. Marzinzik, and G. Gross, “A user-friendly simulation program for teaching power system operations,” *IEEE Transactions on Power Systems*, vol. 10, no. 4, pp. 1725–1733, November 1995.

[27] T.J. Overbye, G. Gross, M.J. Laufenberg, and P.W. Sauer, “Visualizing power system operations in an open market,” *IEEE Computer Applications in Power*, vol. 10, no. 1, pp. 53–58, January 1997.

[28] T.J. Overbye, D.A. Wiegmann, A.M. Rich, and Y. Sun, “Human factors aspects of power system voltage contour visualizations,” *IEEE Transactions on Power Systems*, vol. 18, no. 1, pp. 76–82, February 2003.

[29] A.P.S. Meliopoulos, G.J. Cokkinides, M. Ingram, S. Bell, and S. Mathews, “Visualization and animation of state estimation performance,” in 38^{th} *Annual Hawaii International Conference on Systems Science*, Waikoloa, HI, January 2005.

[30] A. Abur, F.H. Magnago, and Y. Lu, “Educational toolbox for power system analysis,” *IEEE Computer Applications in Power*, vol. 13, no. 4, pp. 31–35, October 2000.

[31] D. Doo and M. Sabin, “Behaviour of recursive division surfaces near extraordinary,” *Computer Aided Design*, vol. 10, no. 6, pp. 356–360, September 1978.

APPENDIX A

INTRODUCTION OF FLTK

FLTK was originally developed by B. Spitzak and is currently maintained by a small group of developers across the world. It is an open-source and cross-platform C++ graphic user interface toolkit for UNIX, Microsoft Windows and MacOS. It composes a group of classes and functions to provide the framework and components used to develop GUI. FLTK is designed to be small and modular enough to be statically linked and works as fine as a shared library. FLTK is chosen to develop vizPET program because,

- It is one of the best free, open-source GUI took kits available.
- It supports 3D graphic via OpenGL and its built-in GLUT emulation.
- It is convenient and it works on almost all popular platforms.

A. Drawing and Events Handling

FLTK has a virtual method *fltk::Widget::draw()*, where the user can write their codes. By making a subclass of one of the existing *fltk::Widget* classes, the users can implement their own version of draw().

To handle the events, FLTK provides a virtual method *fltk::Widget::handle()*. All the events are identified by the integer argument passed to this method. The user has to realize implement this virtual method to handle all the events from mouse, keyboard or widgets.

B. Using OpenGL in FLTK

There are two ways to use OpenGL in FLTK. One way is to make a subclass of *fltk::GlWindow*. FLTK's <*FL/gl.h*> header file must be included in the subclass. It will include the file <*GL/gl.h*> and define some extra drawing functions provide by FLTK. To make subclass *fltk::GlWindow*, a *draw()* method and a *handle()* method must be provided. The *draw()* method uses OpenGL calls to draw the display, it is where the actual OpenGL drawing happens and *handle()* method handles mouse and keyboard events for the window.

Another way is to put OpenGL code into *fltk::Widget::draw()* method, and include <*fltk/gl.h*> folder. The OpenGL drawing code starts with *gl_start()* and ends with *gl_finish()*, and all the OpenGL drawing functions must be put in between.

APPENDIX B

INTRODUCTION OF OPENGL

OpenGL was fist introduced in 1992, and it is the most widely used Application Programming Interface (API) for developing interactive 2D and 3D graphics applications. OpenGL uses a client-server model. It is portable and hardware independent, which means that all the OpenGL applications can produce the same visual display results on any OpenGL API-compliant hardware, regardless of the operating system.

The default language for OpenGL is C/C++. OpenGL incorporates about 150 commands for the users to do rendering, texture mapping, special effects and other visualization function, which simplify the development of software graphics. To the programmer, OpenGL works as a software interface between user and the graphic hardware.

OpenGL provides software developers geometric and image primitives, display lists, modeling transformations, lighting and texturing, antialiasing, blending and many other features. It doesn't provide high-level commands to describe 3D models; instead, all the complex models have to be built up from geometric primitives, such as points, lines and polygons. OpenGL provides ways that the user can change certain states that control how OpenGL renders the specified objects.

The order of series of processing states of OpenGL operations is called OpenGL rendering pipeline. It is shown as Figure B-1. Geometric data (vertices, lines and polygons) is first transformed to camera co-ordinate system, and then all the vertices that outside the view will be eliminated, this procedure is called 3D clipping; next step is to project 3D objects onto 2D plane and then convert them into pixel values. Pixels data (images, and bitmaps) is treated slightly different, it will goes through the pixel operations and then enter the final *rasterization* procedure. Both data will be written into framebuffer. Framebuffer is an area of memory, which holds all the information that graphics display needs to control the color and intensity of all the pixels on the screen.

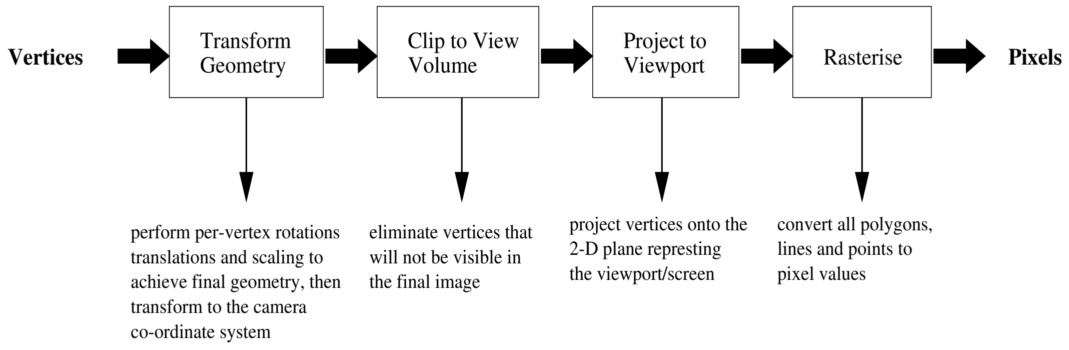

Vertices
Transform Geometry
Clip to View Volume
Project to Viewport
Rasterise
Pixels
perform per-vertex rotations translations and scaling to achieve final geometry, then transform to the camera co-ordinate system
eliminate vertices that will not be visible in the final image
project vertices onto the 2-D plane represting the viewport/screen
convert all polygons, lines and points to pixel values

Fig. B-1. OpenGL rendering pipeline

APPENDIX C

THE BASE CLASSES USED IN THE 3D GUI

All the bases class used in the developed 3D graphic user interface are introduced in this appendix.

A. OBJObject Class

OBJObject class is the base class for all the 3D object classes used in vizPET program, such as PETGraphicBus, PETGraphicMeter, PETGraphicKnob and PETTerrain. It provides all the methods related to 3D object constructing and rendering. OBJObject class consists of arrays of vertices, faces and materials. In the next sections, the data structure of OBJObject, its member variables and functions, and the classes of OBJVertex, OBJFace, OBJFaceVertex and OBJMaterial will be introduced.

1. Data structure of OBJObject class

In real world, an object consists of a series of vertices and faces; each face is composed of corners. In OBJObject class, an object contains a face array and a vertex array. Each face in the face list contains a list of face vertices, which compose the face. (Corner is defined as face vertex in OBJObject class.) Each vertex in the vertex list contains a list of pointers pointing to the corresponding face vertices that belong to this vertex. This data structure can clearly describe an object and support the functions such as tracing a vertex/face, inserting and deleting a vertex/face etc.

Figure C-1 gives an illustration of the data structure for a tetrahedron. The object has a vertex list of vertices 1, 2, 3, 4 and a face list of faces 1, 2, 3, 4. Face

1 contains face vertices 1, 3, 2. It has to be pointed out that the orientation of face vertex has to be unique. Right hand orientation is used in this class. For vertex 1, it has a list of pointers pointing to all the face vertices (corners) belong to it. In this way, it is easy to find a face from a vertex, and it is also possible to find the vertex from a give face.

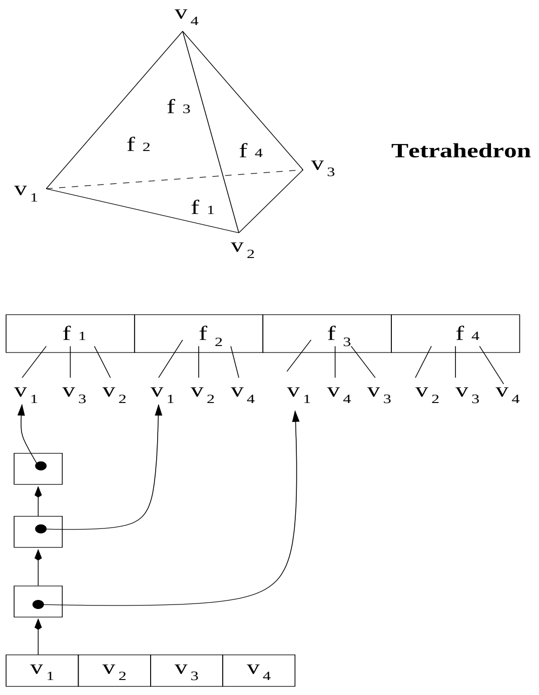


Fig. C-1. Data structure of OBJObject class

2. Variables and functions of OBJObject class

OBJObject class has member variables *position, scale, rotation*, which record the position, scale and the rotation of the object. It contains *vertex_array, matl_array, face_array*, which record the information of all the vertices, faces, materials of an object. Tow Vertex3d arrays, *normals* and *texcoords* contain information for normals and textured IDs of each and every faces, these information is used for loading texture and lighting the object. A Transformation variable *tr* is used for OpenGL transformations.

```
class OBJObject
  {
    public :
      Vector3d           position;     // Position of object
      Vector3d           scale;        // Scale of object
      Quaternion         rotation;     // Rotation of object

    protected :
      OBJVertexArray     vertex_array; // Array of vertices
      OBJMaterialList    matl_list;    // List of materials
      OBJFaceArray       face_array;   // Array of faces

      static Vector3dArray normals;    // Array for reading normals
      static Vector2dArray texcoords;  // Array for reading texture coordinates
      static Transformation tr;        // For doing GL transformation
      ...
  }
```

OBJObject class provides a series subroutines to render the object under different occasions.

```
  // Use material information to render faces.
  // Use GL\_TRIANGLES/GL\_QUADS/GL\_POLYGON as parameters to render faces.
void OBJObject::render(void) const

  // Use material properties to outline faces.
  // Use GL\_LINE\_LOOP as parameters to render outline.
void OBJObject::outline(void) const

  // Render faces without using material properties.
  // Uses face vertex colors and normals.
void OBJObject::renderFaces(void) const

  // Outline faces without using material information.
  // Uses face vertex colors.
void OBJObject::outlineFaces(void) const

  // Render faces without using material information or face vertex colors.
  // Uses the Face normals.
void OBJObject::plainRender(void) const

  // Outline faces without using material properties or face colors.
void OBJObject::plainOutline(void) const

  // Use material and texture information to render faces.
void OBJObject::renderT(void) const

  // Render faces without using material properties.
  // Uses texture information, face vertex colors and normals.
void OBJObject::renderFacesT(void) const

  // Render faces without using material properties or face colors.
  // Uses texture information and face normals.
void OBJObject::plainRenderT(void) const
```

OBJObject also provide the method to read and write an object from a data file. The file is usually ended with ".obj". This file contains the coordinates of all the vertices and faces information. Vertices information is listed starting with "v" followed by the x, y, z coordinates of the vertex. Face information is listed starting with "f" followed by the vertex numbers, which compose the face. Right hand orientation is adopted for the face. The lines followed by # are comment lines.

For example, for the tetrahedron given in Figure C-1, the format of the object data file is given bellow.

```
#Tetrahedron.obj
 v -10 -5 -5
 v  0  -5 10
 v  10 -5 -5
 v  0  10  0
# 4vertices
 f 1 3 2
 f 1 2 4
 f 1 4 3
 f 2 3 4
# 4 faces
```

3. OBJVertex class

OBJVertex implements a vertex class for OBJObject class. It contains the coordinates of this vertex and a list of pointers to each and every face vertices that belong to this vertex.

```
class OBJVertex
  {
    public :
      Vector3d             coords;     // Coordinates of vertex
         // List of face vertices sharing this vertex
      OBJFaceVertexPtrList fvptr_list;
      ...
  }
```

4. OBJFace class

OBJFace implements a face class for OBJObject class. It consists a list of face vertices that compose this face and a list of pointers to the materials for this face.

```
class OBJFace
  {
    protected :
      OBJFaceVertexList  fv_list;   // List of face vertices
      OBJMaterialPtr     matl_ptr;  // Pointer to material for this face
      ...
  }
```

5. OBJFaceVertex class

OBJFaceVertex implements a face vertex class for the OBJObject class. It contains a pointer to the associate vertex, the normal, color and texture ID of this corner.

```
class OBJFaceVertex
  {
    public :
      OBJVertexPtr  vertex;      // Associated vertex pointer
      Vector3d      normal;      // Normal
      RGBColor      color;       // Color
      Vector2d      texcoord;    // Texture coordinate
      ...
  }
```

6. OBJMaterial class

OBJMaterial implements a material used in OBJObject class.It contains the name and the color of the material. It also has a list of pointers pointing to the faces that use this material.

```
class OBJMaterial
  {
    public :
      char *          name;     // Name of material
      RGBColor        color;    // Material diffuse color
      OBJFacePtrList  faces;    // Pointers to faces using this material
      ...
  }
```

B. Other Base Classes

1. BaseObject class

BaseObject is an abstract base class. It can be used to build container classes. BaseObject has no member data; it only has protected constructors, virtual destructors and an assignment operator.

2. Vector3D class

Vector3d class is derived from BaseObject class. It is a class for a 3D vector. All the vector operations are defined in this class, such as vector addition (+), vector subtraction (−), scalar/dot product (∗), cross product (%), scalar division (/), calculating the norm of the vector and normalizing a given vector.

Vector2D, Vector4D are two classes for 2D and 4D vectors, which are similar to Vector3D.

3. Quaternion class

Quaternion is a class that is mostly used for rotation. It has a Vector3d component and a double scalar component. They function as axis and angle respectively. The class provides the following arithmetic operations such as addition (+), subtraction (−), product (∗), division (/), and the functions to get/set angle and axis and to scale angle.

4. Matrix3x3 class

Matrix3x3 is a class for 3×3 matrix. It is derived from BaseObject and built from Vector3d. Each row of the matrix is a Vector3d. It defines the operations needed for matrix such as addition (+), subtraction (−), product (∗), division (/), transpose, determinant and inverse of a matrix.

Matrix4x4 is a modified version of class Matrix3x3 for 4×4 matrix.

5. Transformation class

Transformation is the class for transformations such as translation, scaling and rotation. It has to be noted that the class has an operation *lookat*, which functions as *gluLookAt* in OpenGL. It creates a 4×4 matrix for the given eye, center and upvector.

6. Camera class

Camera is one of the important base classes. This class encapsulates an OpenGL type camera. It contains the eye position; the point of interest (*lookat)*, up vector and it also has projection parameters (field-of-view, aspect ratio, near/far). It supports both perspective and orthogonal projections, and it also supports symmetric-view (along each axis).

```
enum ProjectionType { Orthographic=0, Perspective=1 };

class Camera
  {
    protected :
      Vector3d  center;                 // Point of interest
      Vector3d  eye;                    // Eye position
      Vector3d  up;                     // Up-vector
      double    nearl, farl;            // Near/far clipping planes

        // For orthographic/perspective projection
      double    umin, umax, vmin, vmax;  // Viewing volume
      double    fovy, aspect;           // Y field-of-view, aspect ratio
      double    dist;                   // Dist from eye to center
      ProjectionType projtype;          // Type of projection

        // For GL selection mode
      double    mousex, mousey, pickw, pickh;  // Selection region
                                               // for gluPickMatrix
      GLint *   viewport;               // The viewport
      bool      pickmode;               // in selection mode
      ...
  }
```

In the camera class, a set of OpenGL commands are encapsulated in the amera::applyTransform() function.

```
  //--- Apply the camera transformation ---//
void Camera::applyTransform(void) const
  {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if ( pickmode )
      gluPickMatrix(mousex, mousey, pickw, pickh, viewport);
    if ( projtype == Perspective )
      gluPerspective(fovy,aspect,nearl,farl);
    else if ( projtype == Orthographic)
      glOrtho(umin,umax,vmin,vmax,nearl,farl);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
      double e0=eye[(uint)0];
      double e1=eye[(uint)1];
      double e2=eye[(uint)2];
      double c0=center[(uint)0];
      double c1=center[(uint)1];
      double c2=center[(uint)2];
      double u0=up[(uint)0];
      double u1=up[(uint)1];
      double u2=up[(uint)2];
    gluLookAt(e0,e1,e2,c0,c1,c2,u0,u1,u2);
  }
```

7. Arcball class

This class implements an Arcball, which takes mouse events (mouse down, mouse drag, mouse up) as input and creates the appropriate quaternion and the 4×4 matrices to present the rotation given by the mouse.

8. TransControl class

TransControl class implements a translate controller. It contains a Matrix4x4 class member to record the transformation matrix, and Vector3d vectors to record the current/updated mouse points and the current/updated translations.

9. DollyControl class

DollyControl class implements a dolly translate controller. It is very similar to TransControl class except it has a scale factor, which can scale the given coordinate value of the mouse.

10. ZoomControl class

ZoomControl class implements a zoom controller by scaling in the current XY plane.

11. Viewport class

Viewport implements a generic view port class. It is another important base class used in the program. It supports rotation, panning, zooming and dollying actions. It uses a transformation matrix and its own subroutines to handle the corresponding events.

```
enum VPView {VPPersp=0, VPFront=1, VPRight=2, VPTop=3, VPBack=4,
VPLeft=5,
             VPBottom=6};
enum VPTransformType {VPNone=0, VPPan=1, VPRotate=2, VPZoom=3,
VPDolly=4 }; enum VPMouseEvent {VPUnknown=0, VPPush=1, VPRelease=2,
VPDrag=3};

class Viewport {
  protected :
    int width, height;                  // Width and height of viewport
    Arcball arcball;                    // The arcball controller
    TransControl trcontrol;             // Translation controller
    ZoomControl zoomcontrol;            // Zoom controller
    DollyControl dollycontrol;          // Dolly controller
    Transformation transform;              // Combined transformation
    VPTransformType currenttr;             // Current transformation
    VPView   view;                         // view

  public :
      // Made public to allow changing camera settings easily
    Camera camera;                         // Camera attached to this window
    ...
}
```

In its function member Veiwport::switchTo(), the camera settings are changed to get the specified view.

```
  // Change camera settings to get specified view
void Viewport::switchTo(VPView v)
```

In Viewport::MouseToViewport() function, mouse coordinates are converted to real world coordinates.

```
  // Convert mouse coordinates to real-world coordinates
void Viewport::mouseToViewport(double& x, double& y, double& z)
```

The class also provides a series of virtual subroutines to handle mouse events for rotation, panning, dollying.

```
  // Handle rotation by mouse
virtual void Viewport::handle_rotate(VPMouseEvent event, int
event_x, int event_y)

  // Handle panning by mouse
virtual void Viewport::handle_pan(VPMouseEvent event, int event_x,
int event_y)

  // Handle zooming by mouse (only x movement is used)
virtual void Viewport::handle_zoom(VPMouseEvent event, int event_x,
int event_y=0)

  // Handle dollying by mouse (only x movement is used)
virtual void Viewport::handle_dolly(VPMouseEvent event, int event_x,
int event_y=0)
```

The transformation under different cases (VPPan, VPZoom, VPRotate) is applied in Viewport::apply_transform() function. It composes a series of OpenGL commands to implement the transformation.

```
void Viewport::apply_transform(void) const // Apply the
transformation
```

VITA

Bei Xu received her B.S. degree and M.S. degree in electrical engineering from Shanghai Jiao Tong University, China, in 1998 and 2001, respectively. In January 2002, she began pursuing her Ph.D. degree in the Department of Electrical and Computer Engineering at Texas A&M University. She can be reached at:

Bei Xu

Department of Electrical and Computer Engineering

Texas A&M University

College Station, Texas 77843

The typist for this dissertation was Bei Xu.