UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Department of Electrical Engineering

Effect of Placement of PMUs on State Estimation in a Power System

Iurii Mironov

SHO6262 Master thesis in Electrical Engineering - July 2016

UiT / THE ARCTIC UNIVERSITY OF NORWAY

Title:	Effect of Placement of PMUs on State Estimation in a Power System	*Date:* 04.07.2016 *Classification:* Open
Author:	Iurii Mironov	*Pages:* 57 *Attachments:* 1
Departement:	Department of Electrical Engineering	
Studieretning:	Electrical Engineering	
Supervisors:	Pawan Sharma, Charu Sharma	
Principal:	UiT The Arctic University of Norway (Campus Narvik)	
Principal contact:	Pawan Sharma, Charu Sharma	
Keywords:	Phasor measurement unit (PMU), power system state estimation, IEEE bus system, optimal placement, network observability	

Abstract:

State estimation is one of the most important processes in modelling and monitoring of a power system. Iterative and less accurate conventional means of estimation are now being replaced by fast and direct state vector measurements provided by PMUs. However, the high cost of PMUs forces engineers to choose wisely where the measurement units should be placed.

The given project observes different ways to incorporate the PMU measurements to enhance power system state estimation, depending on the desired depth of observability and the amount of conventional measurements included. It also investigates the outcomes of such schemes in terms of estimation error. To evaluate the outcomes, numerical simulation has been carried out using a model designed in MATLAB.

PREFACE AND ACKNOWLEDGEMENT

This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Science at UiT The Arctic University of Norway (Campus Narvik). The work has been carried out at the Department of Electrical Engineering from January to July 2016.

First of all, I would like to express my sincere gratitude to my supervisors, Associate Professor Dr. Pawan Sharma and Dr. Charu Sharma, for supporting and sharing their expertise with me through the entire work. This thesis would hardly be done without their valuable advices and careful guidance towards problem solutions.

I also want to thank my groupmates Oleksandr Starynets and Carlos Rodriguez Cortez for their motivation and support during the work with my thesis.

Finally, I'm deeply grateful to all my friends who made these two years in Narvik an unforgettable experience.

LIST OF ABBREVIATIONS

CSE	classic state estimator
DFT	discrete Fourier transform
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
ILP	integer linear programming
LSE	linear state estimation
PMU	phasor measurement unit
SCADA	Supervisory Control and Data Acquisition
WLS	weighted least squares

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Problem background

Over the last century electric power has obviously become deeply integrated into our everyday routine. The power network of the Nordic countries developed greatly over time, becoming a complex system combining conventional and renewable energy sources, various consumers and vast transmission system [1]. A single failure in such a system can lead to serious consequences and should preferably be predicted and avoided. Therefore, the ability to monitor such a complicated system is a crucial prerequisite for stable and reliable operation of today's smart grid.

When the network stability issue is addressed, one of the most important functions is to determine the power system state at any point of the network at a given instant of time. The state variables are the voltage magnitudes and relative phase angles at the system buses. The ideas of least-squares estimation appeared in 19th century in applications in the aerospace field [2]. Later, static and dynamic estimators were developed for power systems. Early estimation algorithms used measurements of power flows to produce the best estimate for the system voltage and phase angles [3]. However, they could not measure the system state directly.

Although the concept of using phasors to describe power system operating quantities was introduced in 1893 [4], the earliest application involving direct phasor measurement was reported in early 1980s by Dr. Arun G. Phadke and Dr. James S. Thorp at Virginia Tech and the first commercially available PMUs appeared in early 1990s [3]. The prototype utilized the Global Positioning System (GPS) technology to achieve time synchronization between remote measurements. The implementation of such device not only allowed to measure the system state variables directly, but also made it possible to redesign the state estimation method. Iterative and time-consuming process

could be substituted with a set of linear equations, reducing the number of calculations and increasing the state refresh rate. Continuously developing and worldwide integrating PMU technology can provide the system operators with a better picture of the network and improve the quality of system monitoring.

1.2 Objective of thesis

Although PMUs provide precise measurements of system state, the average cost per PMU ranges between $40 000 and $180 000 and depends on various factors described in [5]. This forces engineers to make compromises on PMU placement and combining phasor measurements with existing estimation techniques.

This project examines different schemes utilizing PMU measurements and evaluates their effect on the system state estimation. The main objective of this thesis is to build up a model of a power system, which allows to make the state variables calculation based on desired method of state estimation, given amount of measurements and observability concern. The outcomes of these schemes will be analyzed and the most optimal placement techniques, producing minimal estimation error, will be proposed.

1.3 Thesis outline

The given thesis is divided into chapters to give a clear and structured picture on the problem.

Chapter 2 discusses the basic principle of classic state estimation, the concept of non-linear weighted least squares and how it is applied for state estimation in power systems. The equations of non-linear functions of system state variables are also derived.

Chapter 3 begins with a short insight into the common structure of PMU and its basic principle of operation. The concept of linear weighted least squares is explained and applied to linear state estimation for two cases:

utilizing PMU measurements exclusively or combined with traditional estimate.

Chapter 4 concerns the optimal PMU placement problem. Complete and incomplete observability cases are discussed. The effect of inclusion of conventional measurements on the optimal placement is also considered. Examples illustrate how integer linear programming is applied to find the optimal solution.

Chapter 5 explains how the theory above was applied to make the model script in MATLAB. The main features of the model are discussed.

In *chapter 6* the simulation results are presented and analyzed.

Chapter 7 concludes the work done and suggests problems for future discussions.

2 CLASSIC STATE ESTIMATION

As it was already mentioned above, state estimation provides the best possible approximation of system state according to the given input data. Such data usually is:

- *network model*, which includes network topology (line connections and circuit breakers status information) and lines and transformers characteristics (impedance, tap ratio);

- *measurements*, which for traditional estimators are measurements from SCADA system (real and reactive power flows, power injections, voltage and current magnitudes) [6].

Based on the given data, the estimator calculates the state vector *x*, which comprises voltage magnitudes and phase angles at the system nodes (buses). As for the classic estimator, this is performed with *non-linear weighted least squares estimation*, described below. Alternative formulations of the minimization criterion are covered in [7].

2.1 Mathematical basis

The measurement equation is formulated as follows [2], [3], [6]:

$$z = h(x) + \varepsilon, \tag{2.1}$$

where z – measurements vector;

$h(x)$ – vector of non-linear functions, relating measurements to the state vector x;

ε – measurement error vector.

The weighted least squares (WLS) method produces a state estimate, such that the sum of squared measurement residuals, weighted by their respective error covariances, is minimal. In other words, the task is to minimize the following objective function:

$$J(x)=\sum_{i=1}^{m}\frac{\left(z_i-h_i(x)\right)^2}{R_{ii}}, \tag{2.2}$$

or in matrix form:

$$J(x)=\left[z-h(x)\right]^T R^{-1}\left[z-h(x)\right], \tag{2.3}$$

where R is the diagonal weighting matrix of variances σ_i^2;

$$R=\begin{bmatrix}\sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \cdots & \\ & & & \sigma_m^2\end{bmatrix}, \tag{2.4}$$

where m is the total number of measurements.

Equation (2.3) is to be minimized recursively until the state variances meet the convergence limit (Δx becomes less than some tolerance value). When the initial conditions ($k = 0$) are set, the state vector estimate for k-th iteration will be defined by

$$x^{k+1}=x^k+\Delta x; \tag{2.5}$$

$$\Delta x=\left(H^T R^{-1}H\right)^{-1}H^T R^{-1}\left(z-h\left(x^k\right)\right). \tag{2.6}$$

Here H is a matrix of first partial derivatives of the elements of h with respect to the components of x (Jacobian matrix) evaluated at k-th iteration:

$$H=\frac{\partial h(x)}{\partial x}=\begin{bmatrix}\frac{\partial h_1(x)}{\partial x_1} & \frac{\partial h_1(x)}{\partial x_2} & \cdots & \frac{\partial h_1(x)}{\partial x_n}\\ \frac{\partial h_2(x)}{\partial x_1} & \frac{\partial h_2(x)}{\partial x_2} & \cdots & \frac{\partial h_2(x)}{\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial h_m(x)}{\partial x_1} & \frac{\partial h_m(x)}{\partial x_2} & \cdots & \frac{\partial h_m(x)}{\partial x_n}\end{bmatrix}, \tag{2.7}$$

where n is the number of state variables.

The numerical values of matrix *H* must be updated for each iteration step.

2.2 Power system application

The measurement functions that relate SCADA measurements to state variables are the following [8].

Power injections:

$$P_i = V_i \sum_{j=1}^{N} V_j \left(G_{ij} \cos\left(\theta_i - \theta_j\right) + B_{ij} \sin\left(\theta_i - \theta_j\right) \right); \quad (2.8)$$

$$Q_i = V_i \sum_{j=1}^{N} V_j \left(G_{ij} \sin\left(\theta_i - \theta_j\right) - B_{ij} \cos\left(\theta_i - \theta_j\right) \right), \quad (2.9)$$

where *N* is the total number of buses, connected to bus *i.*

Power flows:

$$P_{ij} = V_i^2 \left(g_i + g_{ij} \right) - V_i V_j \left(g_{ij} \cos\left(\theta_i - \theta_j\right) + b_{ij} \sin\left(\theta_i - \theta_j\right) \right); \quad (2.10)$$

$$Q_{ij} = -V_i^2 \left(b_i + b_{ij} \right) - V_i V_j \left(g_{ij} \sin\left(\theta_i - \theta_j\right) - b_{ij} \cos\left(\theta_i - \theta_j\right) \right). \quad (2.11)$$

Line current magnitudes:

$$I_{ij} = \frac{\sqrt{P_{ij}^2 + Q_{ij}^2}}{V_i}. \quad (2.12)$$

In the equations above, g_{ij}, b_{ij} correspond respectively to series conductance and susceptance of the line connecting buses *i* and *j*, and g_i, b_i are shunt conductance and susceptance of the line. These parameters are found with an assumption that the line is modelled as the π-equivalent, shown in Figure 2.1. G_{ij}, B_{ij} are respectively real and imaginary parts of *ij*-th element pf admittance matrix *Y* [9].

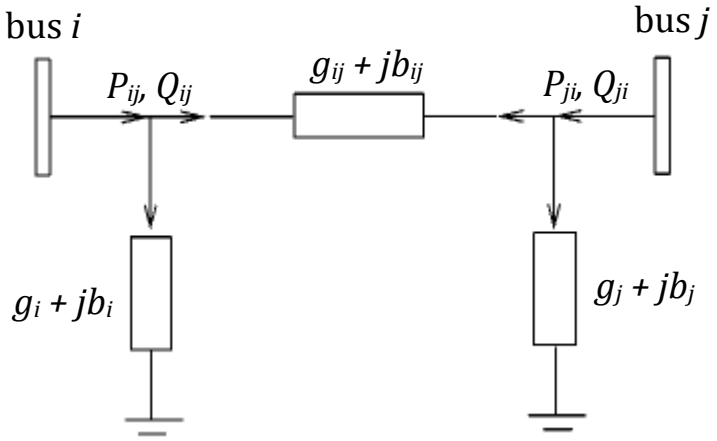

Fig. 2.1. The π-equivalent model of a transmission line

The measurement Jacobian matrix H will be formed as follows [6]:

$$H = \begin{bmatrix} \frac{\partial P_{inj}}{\partial \theta} & \frac{\partial P_{inj}}{\partial V} \\ \frac{\partial P_{flow}}{\partial \theta} & \frac{\partial P_{flow}}{\partial V} \\ \frac{\partial Q_{inj}}{\partial \theta} & \frac{\partial Q_{inj}}{\partial V} \\ \frac{\partial Q_{flow}}{\partial \theta} & \frac{\partial Q_{flow}}{\partial V} \\ \frac{\partial I_{ij}}{\partial \theta} & \frac{\partial I_{ij}}{\partial V} \\ 0 & \frac{\partial V_i}{\partial V} \end{bmatrix} \tag{2.13}$$

The partial derivatives of the equations (2.8) – (2.12) are the following [6]:

- elements corresponding to real power injection measurements:

$$\frac{\partial P_i}{\partial \theta_i} = \sum_{j=1}^{N} V_i V_j \left(-G_{ij} \sin\left(\theta_i - \theta_j\right) + B_{ij} \cos\left(\theta_i - \theta_j\right) \right) - V_i^2 B_{ii}; \tag{2.14}$$

$$\frac{\partial P_i}{\partial \theta_j} = V_i V_j \left(G_{ij} \sin\left(\theta_i - \theta_j\right) - B_{ij} \cos\left(\theta_i - \theta_j\right) \right); \tag{2.15}$$

$$\frac{\partial P_i}{\partial V_i} = \sum_{j=1}^{N} V_j \left(G_{ij} \cos\left(\theta_i - \theta_j\right) + B_{ij} \sin\left(\theta_i - \theta_j\right) \right) - V_i G_{ii}; \tag{2.16}$$

$$\frac{\partial P_i}{\partial V_j} = V_j\left(G_{ij}\cos\left(\theta_i-\theta_j\right)+B_{ij}\sin\left(\theta_i-\theta_j\right)\right); \qquad (2.17)$$

- elements corresponding to reactive power injection measurements:

$$\frac{\partial Q_i}{\partial \theta_i} = \sum_{j=1}^{N} V_i V_j\left(G_{ij}\cos\left(\theta_i-\theta_j\right)+B_{ij}\sin\left(\theta_i-\theta_j\right)\right)-V_i^2 G_{ii}; \qquad (2.18)$$

$$\frac{\partial Q_i}{\partial \theta_j} = -V_i V_j\left(G_{ij}\cos\left(\theta_i-\theta_j\right)+B_{ij}\sin\left(\theta_i-\theta_j\right)\right); \qquad (2.19)$$

$$\frac{\partial Q_i}{\partial V_i} = \sum_{j=1}^{N} V_j\left(G_{ij}\sin\left(\theta_i-\theta_j\right)-B_{ij}\cos\left(\theta_i-\theta_j\right)\right)-V_i B_{ii}; \qquad (2.20)$$

$$\frac{\partial Q_i}{\partial V_j} = V_j\left(G_{ij}\sin\left(\theta_i-\theta_j\right)-B_{ij}\cos\left(\theta_i-\theta_j\right)\right); \qquad (2.21)$$

- elements corresponding to real power flow measurements:

$$\frac{\partial P_{ij}}{\partial \theta_i} = V_i V_j\left(g_{ij}\sin\left(\theta_i-\theta_j\right)-b_{ij}\cos\left(\theta_i-\theta_j\right)\right); \qquad (2.22)$$

$$\frac{\partial P_{ij}}{\partial \theta_j} = -V_i V_j\left(g_{ij}\sin\left(\theta_i-\theta_j\right)-b_{ij}\cos\left(\theta_i-\theta_j\right)\right); \qquad (2.23)$$

$$\frac{\partial P_{ij}}{\partial V_i} = -V_j\left(g_{ij}\cos\left(\theta_i-\theta_j\right)+b_{ij}\sin\left(\theta_i-\theta_j\right)\right)+2V_i\left(g_{ij}+g_i\right); \qquad (2.24)$$

$$\frac{\partial P_{ij}}{\partial V_j} = -V_j\left(g_{ij}\cos\left(\theta_i-\theta_j\right)+b_{ij}\sin\left(\theta_i-\theta_j\right)\right); \qquad (2.25)$$

- elements corresponding to reactive power flow measurements:

$$\frac{\partial Q_{ij}}{\partial \theta_i} = V_i V_j\left(g_{ij}\cos\left(\theta_i-\theta_j\right)+b_{ij}\sin\left(\theta_i-\theta_j\right)\right); \qquad (2.26)$$

$$\frac{\partial Q_{ij}}{\partial \theta_j} = V_i V_j\left(g_{ij}\cos\left(\theta_i-\theta_j\right)+b_{ij}\sin\left(\theta_i-\theta_j\right)\right); \qquad (2.27)$$

$$\frac{\partial Q_{ij}}{\partial V_i} = -V_j\left(g_{ij}\sin\left(\theta_i-\theta_j\right)-b_{ij}\cos\left(\theta_i-\theta_j\right)\right)-2V_i\left(b_{ij}+b_i\right); \qquad (2.28)$$

$$\frac{\partial Q_{ij}}{\partial V_j} = -V_j\left(g_{ij}\sin\left(\theta_i-\theta_j\right)-b_{ij}\cos\left(\theta_i-\theta_j\right)\right); \tag{2.29}$$

- elements corresponding to voltage magnitude measurements:

$$\frac{\partial V_i}{\partial \theta_i}=0; \quad \frac{\partial V_i}{\partial \theta_j}=0; \quad \frac{\partial V_i}{\partial V_i}=1; \quad \frac{\partial V_i}{\partial V_j}=0; \tag{2.30}$$

- elements corresponding to current magnitude measurements:

$$\frac{\partial I_{ij}}{\partial \theta_i} = \frac{g_{ij}^2+b_{ij}^2}{I_{ij}} V_i V_j \sin\left(\theta_i-\theta_j\right); \tag{2.31}$$

$$\frac{\partial I_{ij}}{\partial \theta_j} = -\frac{g_{ij}^2+b_{ij}^2}{I_{ij}} V_i V_j \sin\left(\theta_i-\theta_j\right); \tag{2.32}$$

$$\frac{\partial I_{ij}}{\partial V_i} = \frac{g_{ij}^2+b_{ij}^2}{I_{ij}}\left(V_i - V_j\cos\left(\theta_i-\theta_j\right)\right); \tag{2.33}$$

$$\frac{\partial I_{ij}}{\partial V_j} = \frac{g_{ij}^2+b_{ij}^2}{I_{ij}}\left(V_j - V_i\cos\left(\theta_i-\theta_j\right)\right). \tag{2.34}$$

The state vector obtained by classic state estimator is in the form:

$$x_{CSE} = \left[\theta_1, \theta_2, ..., \theta_n, V_1, V_2, ..., V_n\right]^T, \tag{2.35}$$

where n is total number of buses; V_i, θ_i – voltage magnitude and phase at the i-th bus.

3 STATE ESTIMATION UTILIZING PMU MEASUREMENTS

With the ability of PMUs to measure the system state directly, the use of phasor measurements for state estimation enhances both speed and accuracy of the process. Unlike classic estimation technique which has to deal with iterative solution of non-linear equations, the PMU measurements are linear functions of state variables. Therefore, the computation process can be significantly simplified.

Direct measurements of state variables can also be augmented by the state vector obtained in the classic estimator. This so called hybrid linear state estimation with a post-processing step [10] will also be discussed in this chapter.

3.1 Phasor measurement unit

A *phasor measurement unit (PMU)* is a device that measures the parameters of the electrical waves and produces output time-stamped measurement data. The hardware configuration of PMUs may depend on the manufacturer; however, its common components are depicted in Figure 3.1.

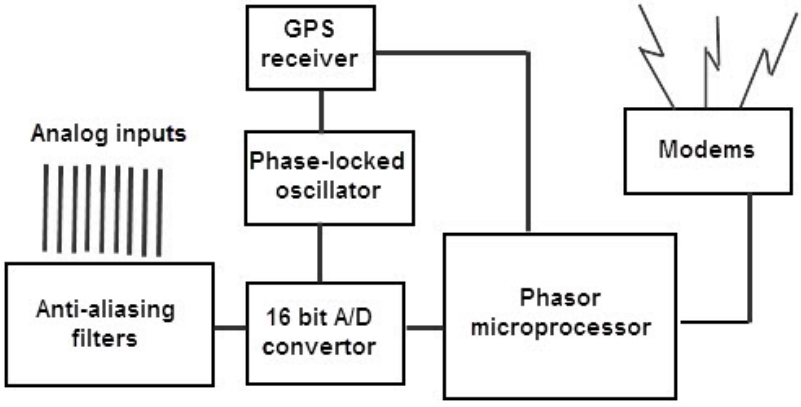

Fig. 3.1. Configuration of modern PMU [11]

The analog inputs are voltages and currents obtained via measurement transformers. These signals are converted to suitable format for the analog-to-digital converters and sampled typically at a rate of 48 samples per second [4].

The Global Positioning System (GPS) acts as a common time source. With a GPS receiver, all measurements are assigned a time stamp to synchronize the data that is time-skewed during transmission via communication link. The sampling clock is also phase-locked with the GPS clock pulse.

Phasor microprocessor calculates positive-sequence estimates of the current and voltage signals using techniques based on discrete Fourier transform (DFT).

Finally, the time-tagged measurement data is transferred via modems to higher levels of the measurement system hierarchy.

Detailed structure of PMU devices, its hierarchy, communication options and applications are discussed in [3] and [12].

Table 3.1 illustrates the key advantages of PMU measurements compared to conventional SCADA measurements.

	SCADA	**PMU**
Resolution	1 sample every 2-4 seconds (steady state observability)	10-60 samples per second (dynamic observability)
Measured quantities	magnitude only	magnitude and phase
Time synchronization	no	yes
Total input/output channels	100+ analog and digital	~10 phasors, 16+ analog, 16+ digital
Focus	local monitoring and control	wide area monitoring and control

Table 3.1. Comparison of SCADA and PMU measurement systems [13]

3.2 Mathematical basis of linear state estimation

As it was mentioned above, PMUs provide direct measurements of system state variables, making the relation between state vector and measurements vector linear.

The objective function that must be minimized, is the same as in chapter 2:

$$J(x)=\left[z-h(x)\right]^{T} R^{-1}\left[z-h(x)\right]. \qquad (3.1)$$

The only difference is that measurement functions *h(x)* are now linear. Therefore, equation (2.1) can be expressed as follows:

$$z = h(x) + \varepsilon = Bx + \varepsilon, \tag{3.2}$$

where B is the system matrix.

The state vector x can therefore be calculated as follows [3]:

$$x = \left[B^T R^{-1} B\right]^{-1} B^T R^{-1} z = Mz, \tag{3.3}$$

where R is the diagonal weighting matrix of measurement variances σ_i^2;

$$R = \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \dots & \\ & & & \sigma_m^2 \end{bmatrix} \tag{3.4}$$

The matrix M is constant as long as the system structure and parameters do not change. It can be computed offline and stored for real-time use [3].

3.3 Power system application

Consider again the π-equivalent model of a transmission line, shown in Figure 3.2:

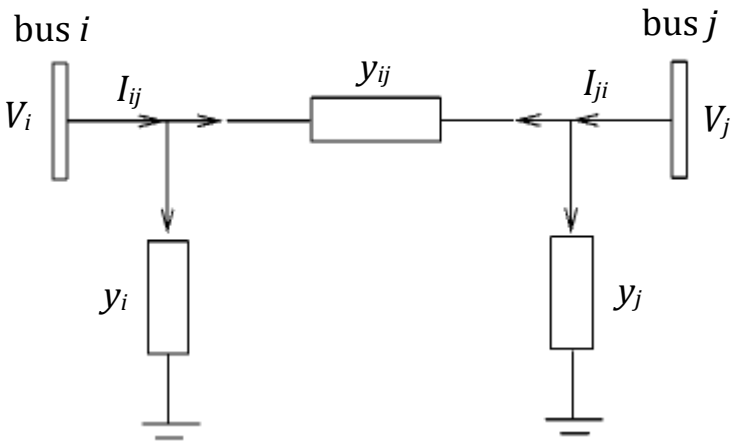

Fig. 3.2. The π-equivalent model of a transmission line

Here V_i, V_j – complex voltages measured at bus i and j respectively; I_{ij}, I_{ji} – complex line currents measured near bus i and j respectively.

Assuming state vector:

$$x = \begin{bmatrix} V_i \\ V_j \end{bmatrix} \tag{3.5}$$

and measurements vector:

$$z = \begin{bmatrix} V_i \\ V_j \\ I_{ij} \\ I_{ji} \end{bmatrix}, \tag{3.6}$$

equation (3.2) is then expressed as follows [3]:

$$\begin{bmatrix} V_i \\ V_j \\ I_{ij} \\ I_{ji} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ y_{ij} + y_i & -y_{ij} \\ -y_{ij} & y_{ij} + y_j \end{bmatrix} \begin{bmatrix} V_i \\ V_j \end{bmatrix}, \tag{3.7}$$

where the system matrix *B*:

$$B = \begin{bmatrix} II \\ yA + y_s \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ y_{ij} + y_i & -y_{ij} \\ -y_{ij} & y_{ij} + y_j \end{bmatrix} \tag{3.8}$$

To explain how the elements of the matrix *B* are constructed, consider 4-bus system, shown in Figure 3.3 with PMUs installed at buses 1, 2 and 4.

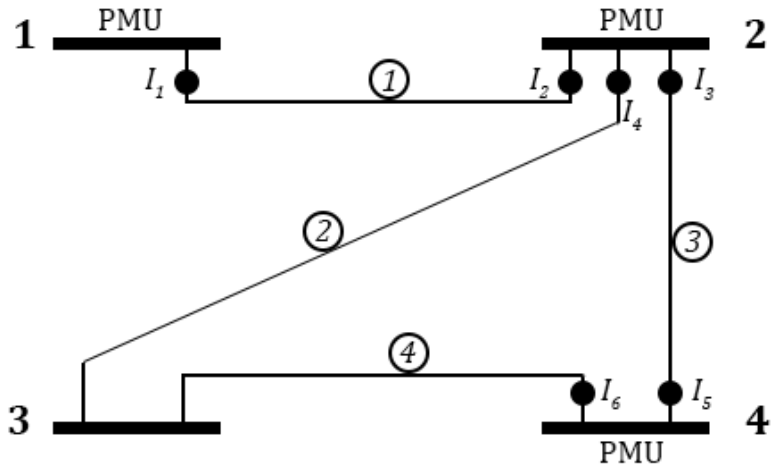

Fig. 3.3. Example 4-bus system

The elements of the matrix *B* are as follows [3]:

II – a unit matrix of width *n* (*n* – number of buses). Rows corresponding to buses without voltage measurements, are removed:

$$II = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

A – current measurement incidence matrix. Rows correspond to current measurements, columns correspond to buses; 1 and –1 indicate respectively the start and the end of the line being measured:

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.10}$$

y – diagonal matrix of series admittances of measured branches:

$$y = \begin{bmatrix} y_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & y_4 \end{bmatrix} \tag{3.11}$$

y_s –matrix of shunt admittances of measured branches. Rows correspond to measurements, columns correspond to buses:

$$y_s = \begin{bmatrix} y_{10} & 0 & 0 & 0 \\ 0 & y_{10} & 0 & 0 \\ 0 & y_{30} & 0 & 0 \\ 0 & y_{20} & 0 & 0 \\ 0 & 0 & 0 & y_{30} \\ 0 & 0 & 0 & y_{40} \end{bmatrix} \tag{3.12}$$

The resulting system matrix:

$$B = \begin{bmatrix} II \\ yA + y_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ y_1 + y_{10} & -y_1 & 0 & 0 \\ -y_1 & y_1 + y_{10} & 0 & 0 \\ 0 & y_3 + y_{30} & 0 & -y_3 \\ 0 & y_2 + y_{20} & -y_2 & 0 \\ 0 & -y_3 & 0 & y_3 + y_{30} \\ 0 & 0 & -y_4 & y_4 + y_{40} \end{bmatrix} \tag{3.13}$$

The measurement function is

$$\begin{bmatrix} V_{PMU} \\ I_{PMU} \end{bmatrix} = \begin{bmatrix} II \\ yA + y_s \end{bmatrix} V, \tag{3.14}$$

where V_{PMU}, I_{PMU} – vectors of measured complex voltages and currents; V – state vector of complex voltages.

Note that this method described in [3] utilizes complex voltages as state variables, which is suitable for simulations in MATLAB. Other papers [11][14] use the notation with voltage real and imaginary parts as separate state variables.

3.4 Hybrid linear state estimation

There are two general techniques that are used to combine PMU measurements with traditional SCADA data.

The first method integrates PMU measurements into classic state estimator and processes them in the same iterative procedure.

Another algorithm utilizes the estimate obtained in classic state estimator through a post-processing step. The state vector is converted to rectangular form, comprising real and imaginary parts of bus voltages. Then it is fed into the linear estimator along with voltage and current measurements.

Tests show [10], that the results of both methods are identical. Therefore, it is preferable to use linear estimator as less complicated and non-iterative algorithm. Also, such approach avoids the problem of physical integration of PMU measurements and rewiring existing traditional estimators.

The measurements vector is augmented by the estimate from the classic state estimator V_{CSE} (in complex form):

$$z_H = \begin{bmatrix} V_{CSE} \\ V_{PMU} \\ I_{PMU} \end{bmatrix} \tag{3.15}$$

The derived earlier system matrix B is augmented by unit matrix I:

$$B_H = \begin{bmatrix} I \\ II \\ yA + y_s \end{bmatrix} \tag{3.16}$$

Also, the covariance matrix of linear estimator, defined here as R_2, is diagonally concatenated with the CSE covariance matrix:

$$R_H = \begin{bmatrix} H_1^T R_1^{-1} H_1 & 0 \\ 0 & R_2 \end{bmatrix} \tag{3.17}$$

The state variable in this case is calculated as follows (index H refers to hybrid linear state estimation):

$$x_H = \left[B_H^T R_H^{-1} B_H \right]^{-1} B_H^T R_H^{-1} z_H . \tag{3.18}$$

4 OPTIMAL PLACEMENT PROBLEM

Placing PMUs at all buses in a power system results in direct measuring of the system state instead of estimating it. However, such a solution could be rather costly. On the other hand, measuring line currents can extend voltage measurements to the buses where no PMUs are installed. Therefore, a minimal number of PMUs can be installed to indirectly measure all the bus voltages in the system. Finding out this smallest number of PMUs as well as their locations in the network has always been a subject to optimization problem. An overview of solution methods for this problem is discussed in [15]. In this project the methods discussed in [16], [17], [18] and [3] were used. All of the methods utilize *integer linear programming (ILP)* to solve optimization problems.

4.1 Complete observability case

As it was mentioned above, a PMU can make installed bus and all connected buses observable. Figure 4.1 describes a system completely observed by two PMUs (marked by large circles). The shade of smaller circles indicates which PMU provides observability to the neighboring buses [3].

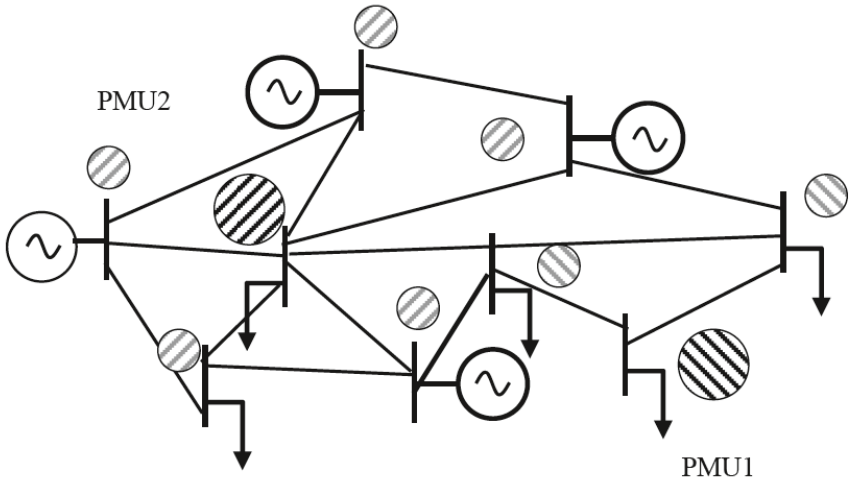

Fig. 4.1. Example of fully observable bus system [3]

The placement problem for complete observability is solved by finding a minimal set of PMUs such that each bus is reached by a PMU at least once [17].

Define incidence matrix T_{PMU}:

$$T_{PMU\ i,j} = \begin{cases} 1, \text{ if } i = j \\ 1, \text{ if } i \text{ and } j \text{ are connected} \\ 1, \text{ otherwise.} \end{cases} \tag{4.1}$$

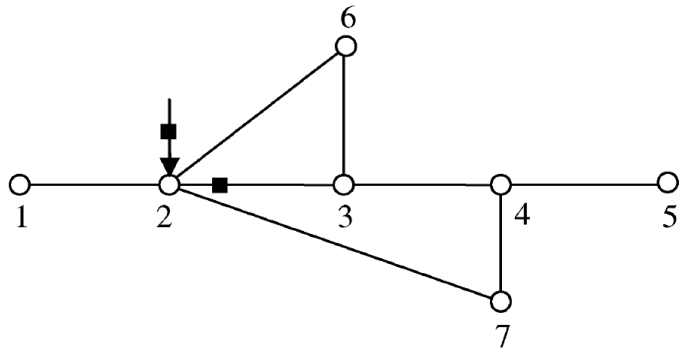

Fig. 4.2. Example 7-bus system [16]

For a 7-bus system illustrated in Figure 4.2 the incidence matrix is constructed as follows [16]:

$$T_{PMU} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Formulation of the optimal placement problem in terms of ILP for N-bus system is the following:

$$\begin{aligned} &\min \quad \sum_{k=1}^{N} x_k \\ &\text{subject to} \quad T_{PMU} X \geq b_{PMU} \\ &X = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T \\ &x_i \in \{0, 1\} \end{aligned} \tag{4.3}$$

where X is PMU placement vector of binary values, with 1 (ones) indicating placement buses; b_{PMU} is the inequality constraints vector:

$$b_{PMU} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T_{1 \times N} \tag{4.4}$$

The solution for the example 7-bus system is

$$X = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T \tag{4.5}$$

This means that PMUs should be installed at buses 2 and 5.

4.2 Incomplete observability cases

The system is incompletely observable when some of the buses cannot be reached by a PMU i.e. the voltages of such buses cannot be indirectly measured by PMUs.

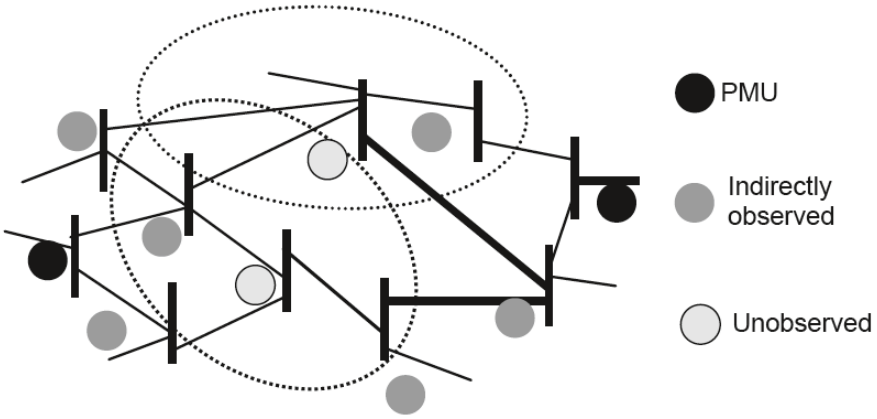

Fig. 4.3. Example of incompletely observable bus system [3]

4.2.1 Depth-of-one unobservability

Depth-of-one unobservability implies that all unobserved buses must be connected only to observed buses. Such condition is illustrated in Figure 4.3.

The ILP formulation for depth-of-one unobservability case is similar to that for full observability. The difference is that the incidence matrix T_{PMU} is modified by matrix A [16]:

$$\begin{aligned} &\min_1 \quad \sum_{k=1}^{N} x_k \\ &\text{subject to} \quad AT_{PMU}X \geq b_1 \\ &X = \begin{bmatrix} x_1 \; x_2 \cdots x_N \end{bmatrix}^T \\ &x_i \in \{0,1\} \end{aligned} \tag{4.6}$$

where A is the branch-to-node incident matrix; b_1 is a unit vector of the same length as the number of branches.

Each row in matrix *A* corresponds to its respective branch and the indexes of 1's (ones) in that row indicate the two buses connected by that branch. For the 7-bus system in Figure 4.2 matrix *A* is constructed as follows [16]:

$$A=\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

The solution for this system is

$$X=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.8}$$

This means that for depth-of-one unobservable system a PMU should be installed at bus 3.

4.2.2 Larger depths of unobservability

For *larger depths of unobservability* an approach proposed in [3] appears to be preferable due to its simplicity. According to that, the depth-of-*M* unobservability is achieved by taking (*M*+1)-th power of the incidence matrix T_{PMU}.

The ILP formulation of this problem is as follows [3]:

$$\begin{aligned} \min_M \quad & \sum_{k=1}^{N} x_k \\ \text{subject to} \quad & \left(T_{PMU}\right)^{M+1} X_M \geq b_{PMU} \\ & \left(\hat{1}-X_{M-1}\right)^T X_M = 0 \\ X_M=\begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T \quad & x_i \in \{0,1\} \end{aligned} \tag{4.9}$$

where $\hat{1}$ is a unit vector; X_{M-1} is the set of PMU locations calculated for depth (M–1) unobservability case.

4.3 Inclusion of conventional measurements

4.3.1 Complete observability case

If the conventional measurements (flow, injection) are considered in optimal placement problem, then some modifications must be made to ILP algorithm.

In previous cases when PMU measurements were utilized exclusively, the constraints were defined in the form:

$$T_{PMU}X \geq \hat{1} \tag{4.10}$$

Each element y_i of the vector $Y = T_{PMU}X$ represent the number of times bus i is reached by PMUs.

Conventional measurements also introduce inequalities that must be considered in ILP [17]:

- if a power flow measurement in on line i–j, then the following needs to be held:

$$y_i + y_j \geq 1 \tag{4.11}$$

which means that at least one of two buses must be reached by PMU.

- if an injection measurement is at bus k which is connected to buses l, p and q, then the following needs to be held:

$$y_k + y_l + y_p + y_q \geq 3 \tag{4.12}$$

These constraints form a matrix T_{meas} in the way which is explained in the following example.

Recall the 7-bus system in Figure 4.2 and consider injection measurement at bus 2 and flow measurement on the line between buses 2 and 3 [16]. These measurements introduce the following constraints:

$$\begin{cases} y_2 + y_3 \geq 1 \\ y_1 + y_2 + y_3 + y_6 + y_7 \geq 4 \end{cases} \tag{4.13}$$

In matrix form:

$$T_{meas} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}; \qquad b_{meas} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} \tag{4.14}$$

where each column represents a bus associated to conventional measurements (respectively 1, 2, 3, 6, 7).

The matrices can be reduced:

$$T_{meas} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}; \qquad b_{meas} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{4.15}$$

Buses that are not associated to conventional measurements are added in the following way:

$$T_{con} = \begin{bmatrix} I_{M\times M} & 0 \\ 0 & T_{meas} \end{bmatrix} = \left[\begin{array}{cc:ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right]; \qquad b_{con} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 2 \end{bmatrix} \tag{4.16}$$

where $I_{M\times M}$ is a unit matrix; M is the number of buses not associated to conventional measurements.

Formulation of the optimization problem in terms of ILP:

$$\begin{aligned} &\min \quad \sum_{k=1}^{N} x_k \\ &\text{subject to} \quad T_{con} P T_{PMU} X \geq b_{con} \\ &X = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T \\ &x_i \in \{0, 1\} \end{aligned} \tag{4.17}$$

where P is a permutation matrix.

As the order of columns (buses) in matrix T_{con} was changed, matrix P changes the order of rows in T_{PMU} in the same manner.

$$P=\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.18}$$

The optimal solution for the given example is

$$X=\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T \tag{4.19}$$

This means that PMUs should be installed at buses 2 and 5.

4.3.2 Depth-of-one unobservability case

The ILP formulation for depth-of-one unobservability case with the inclusion of conventional measurements is similar to that with exclusive PMU measurements. The difference is that matrix P_1 is introduced [16]:

$$\begin{aligned} &\min_1 \quad \sum_{k=1}^{N} x_k \\ &\text{subject to} \quad P_1 A T_{PMU} X \geq P_1 b_1 \\ &X=\begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T \\ &x_i \in \{0,1\} \end{aligned} \tag{4.20}$$

where P_1 is the matrix that removes the branches associated to conventional measurements.

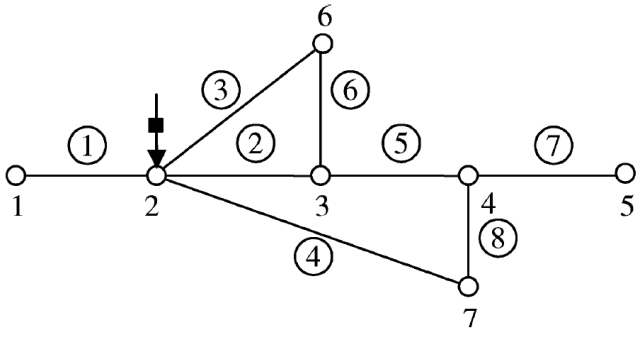

Fig. 4.4. Example 7-bus system for incomplete observability [16]

For the 7-bus example (branches are numbered in Figure 4.4) with injection measurement at bus 2, the matrix P_1 is set as follows:

$$P_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.21}$$

5 MODEL DESCRIPTION

The state estimation model is based on the theory discussed in previous chapters. It was constructed and run in MATLAB version R2016a with reference to [19].

Required input for the model is the following:

1) *bus data* in IEEE common data format [20].

Bus parameters must be stored in a matrix described in Table 5.1.

Column	Value
1	Bus number
2	Bus type: 0 – Unregulated (load, PQ) 1 – Hold MVAR generation within voltage limits, (PQ) 2 – Hold voltage within VAR limits (gen, PV) 3 – Hold voltage and angle (swing)
3	Voltage, pu
4	Angle, degrees
5	Active load, MW
6	Reactive load, MVAR
7	Active generation, MW
8	Reactive generation, MVAR
9	Base voltage, kV
10	Desired voltage (for buses of type 2 and 3), kV
11	Maximum MVAR or voltage limit
12	Minimum MVAR or voltage limit
13	Shunt conductance, pu
14	Shunt susceptance, pu

Table 5.1. Bus data format

2) *branch data* in IEEE common data format [20].

Branch parameters must be stored in a matrix described in Table 5.2.

Column	Value
1	Start bus number
2	End bus number
3	Branch resistance, pu
4	Branch reactance, pu
5	Line charging susceptance, pu
6	Transformer turns ratio
7	Transformer phase shift angle
8	Minimum tap or phase shift
9	Maximum tap or phase shift
10	Step size

Table 5.2. Branch data format

Bus and branch data for IEEE 14-, 30-, 57- and 118-bus systems [20] are stored in files *ieee14.m*, *ieee30.m*, *ieee57.m* and *ieee118.m* respectively. However, any other bus system can be loaded to the model if it is stored in the format described earlier.

Optional input that can be fed to the model:

1) conventional measurements;
2) phasor measurements.

The format for measurements matrix is presented in Table 5.3.

Column	Value
1	Measurement number
2	Type of measurement *For conventional measurements:* 1 – voltage magnitude 2 – real power injection 3 – reactive power injection 4 – real power flow 5 – reactive power flow *For phasor measurements:* 1 – voltage in complex form 2 – current in complex form
3	Measured value, pu
4	Bus number *(for bus measurements)* Start bus number *(for branch measurements)*
5	End bus number *(for branch measurements)*
6	Measurement variance

Table 5.3. Measurements data format

The model script is divided into consecutive sections for clear overview of each step and better understanding of the whole process. The full script can be found in Appendix A as well as on the attached CD.

5.1 Conventional measurements setup

As it was mentioned earlier, the conventional measurements matrix is an optional input. If user does not have a pre-defined measurements data set, it can be created during this section.

The measurements positions are defined with vectors. Vectors *volt* and *inj* contain bus numbers with respectively voltage magnitude and power injection

measurements on them. Vector *flow* contains branch numbers with power flow measurements on them. These vectors can be set manually or using function *randperm*, which picks *N* random buses/branches for measuring.

Measurement variances are also set in vectors *Rvolt*, *Rinj* and *Rflow*, corresponding to vectors of measurement positions.

Function *createZdatas* generates "measured" values on specified positions with specified error variance and writes them to the file *zdatas.m*.

5.2 Classic state estimation

This section utilizes measurements stored in *zdatas.m* to produce the best estimate of the system state applying non-linear weighted least squares method (function *WLS.m*). This function is based on [21] with some modifications made. The output is stored in vectors of bus voltage angles *th_est* and magnitudes *V_est*. They are combined to produce state vector of complex voltages *X_est*, which will be used for hybrid linear state estimation in section 5.5.

5.3 Optimal placement problem

This section utilizes MATLAB function *intlinprog* to solve optimal placement problems. The constraints matrices for ILP are formed as described in chapter 4.

For the cases when PMU measurements are used exclusively, the output is stored in vectors *pmu0*, *pmu1* and *pmu2* for complete observability, depth-of-one and depth-of-two unobservability respectively. These vectors contain binary values, indicating incidence of PMU measurements on the bus.

For the cases when PMU measurements are complemented by conventional measurements, the output is stored in vectors *pmu0conv* for complete observability case and *pmu1conv* for depth-of-one unobservability. The numbers of buses and branches with conventional measurements are set for these cases in vectors *inj2* and *flow2*.

5.4 PMU measurements setup

The PMU measurements file *PMUdatas.m* is constructed in the same way as described in 5.1. The only difference is that the measurement positions are defined by PMU incidence vectors obtained in previous section.

5.5 Linear state estimation

This section utilizes PMU measurements data stored in *PMUdatas.m* to produce the best estimate of the system state applying linear weighted least squares method.

The matrix components are formed in a way explained in chapter 3. Function *LSE.m* produces linear estimate of bus voltage magnitudes *V_lse* and angles *th_lse* with phasor measurements from *PMUdatas.m* only. Function *LSEconv.m* also includes vector *X_est* as a measurement to produce hybrid state estimate *V_hyb* and *th_hyb*.

6 SIMULATION RESULTS

6.1 Optimal placement simulation

The optimal placement algorithm was tested on IEEE 14-, 30-, 57- and 118-bus systems. Simulation results for cases with no conventional measurements are shown in Table 6.1.

Bus system	Complete observability	Depth-of-one unobservability	Depth-of-two unobservability
IEEE 14 Bus	4	2	4
IEEE 30 Bus	10	4	3
IEEE 57 Bus	17	11	7
IEEE 118 Bus	32	18	9

Table 6.1. Optimal number of PMUs for exclusive PMU measurements

Simulation results for cases when conventional measurements are included, are shown in Table 6.2.

Bus system	Number of conventional measurements	Complete observability	Depth-of-one unobservability
IEEE 14 Bus	2	3	2
IEEE 30 Bus	6	7	4
IEEE 57 Bus	15	7	11
IEEE 118 Bus	30	23	15

Table 6.2. Optimal number of PMUs for inclusion of conventional measurements

The results correlate to those shown in [16], so the algorithm can be considered effective. However, it must be taken into consideration, that the results in Table 6.2. depend on the number of conventional measurements and their locations in the system.

6.2 Linear state estimation simulation

The state estimation algorithms were tested on IEEE 14-, 30-, 57- and 118-bus systems. Simulation results are shown graphically on the following figures for comparison of effectiveness of different estimation methods.

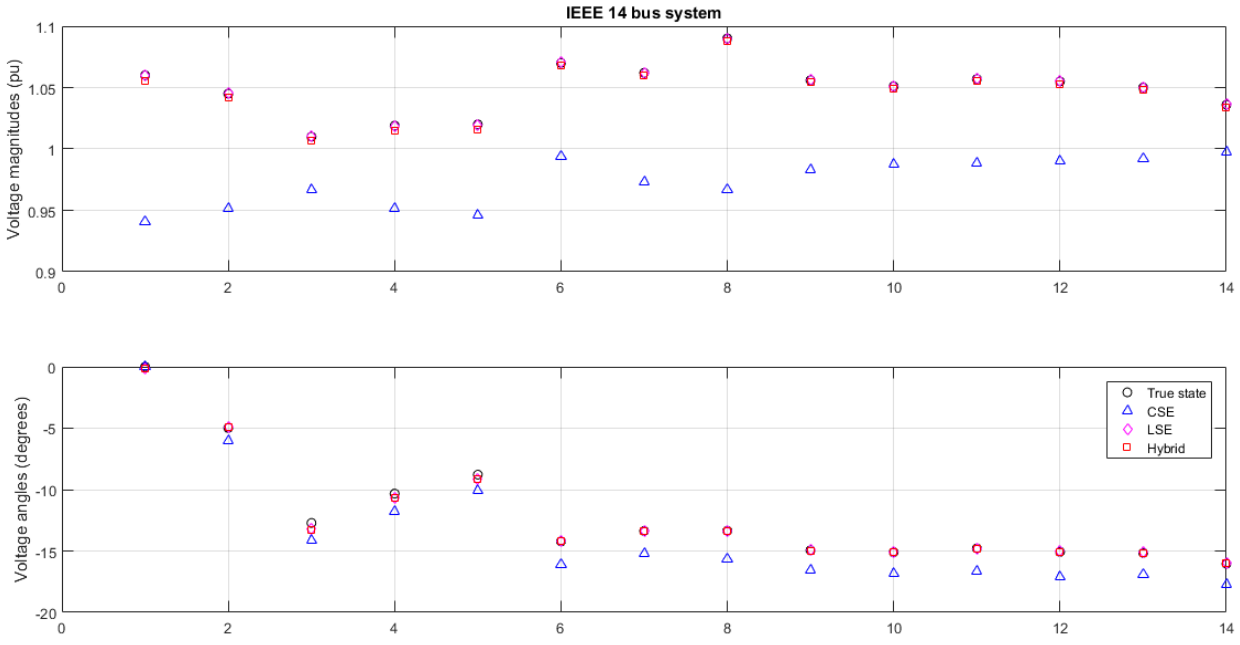

Fig. 6.1. State estimation simulation results for IEEE 14 bus system

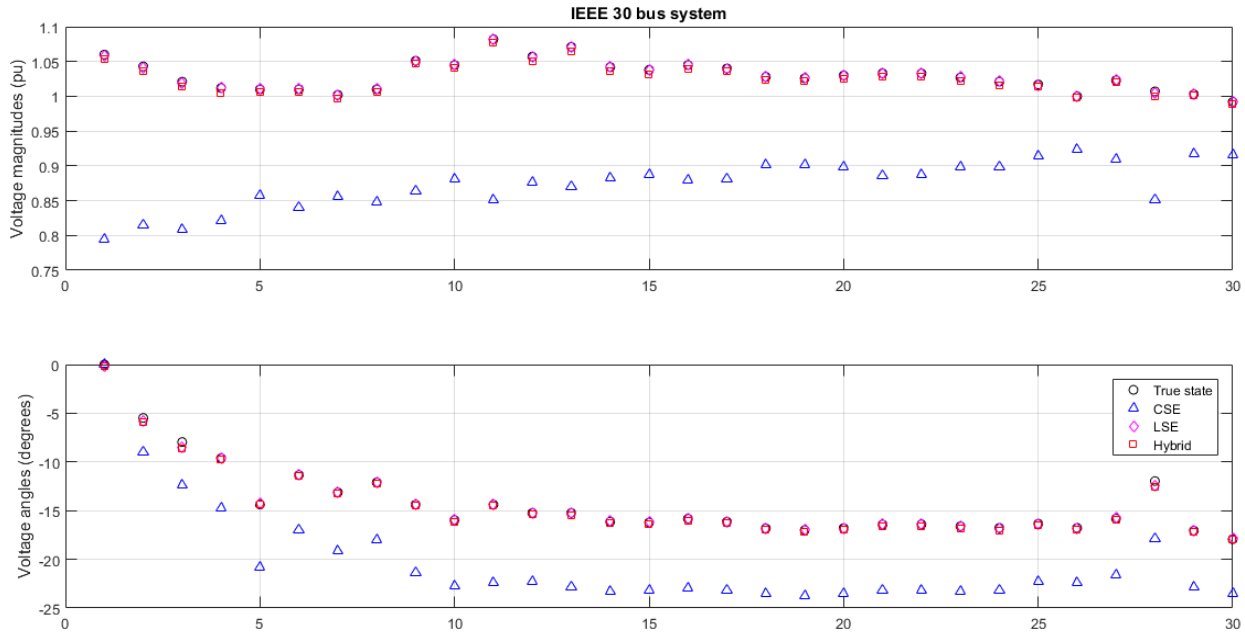

Fig. 6.2. State estimation simulation results for IEEE 30 bus system

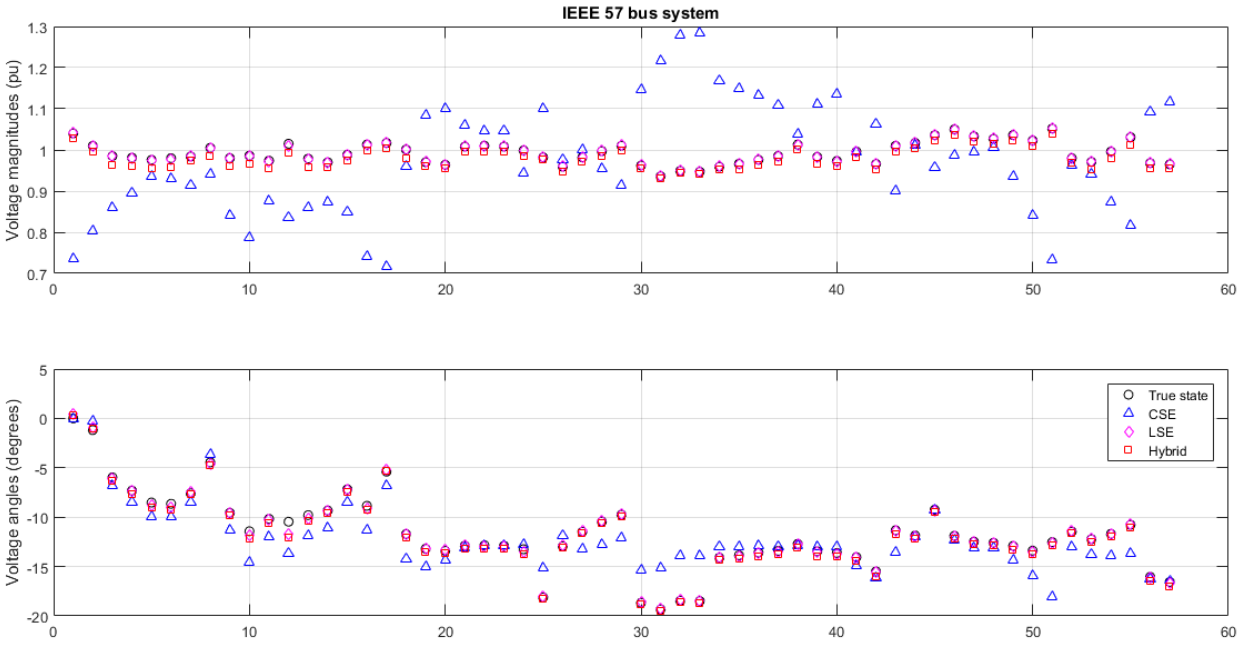

Fig. 6.3. State estimation simulation results for IEEE 57 bus system

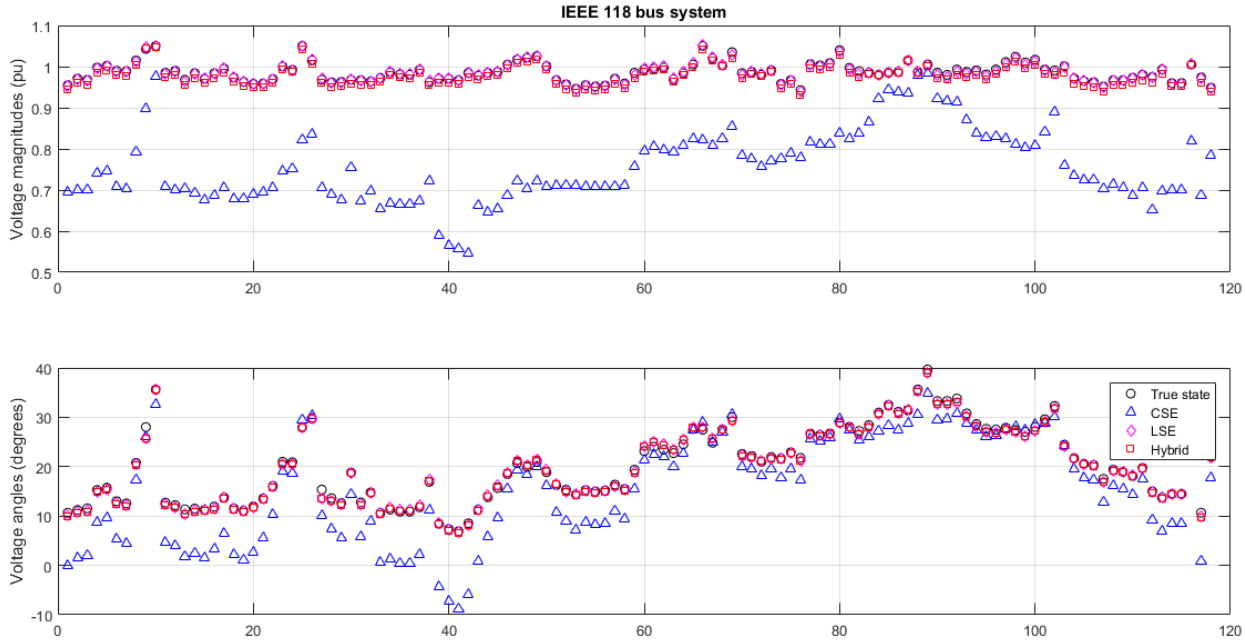

Fig. 6.4. State estimation simulation results for IEEE 118 bus system

The results show, that CSE algorithm shows large deviation of estimated value from the true value as the number of buses increases. It may be caused

by imperfection of the estimation technique for large bus systems or errors in derivation of system equations.

On the other hand, the LSE algorithm with PMUs located at the optimal positions shows very good precision of estimate and little estimation error.

The inclusion of classic estimate in linear estimator does not affect much the overall result obtained in hybrid estimator.

7 CONCLUSIONS AND FUTURE WORK

State estimation is a key function in determining real-time models for power system networks. This thesis discusses how the estimation algorithm is enhanced by placement of PMUs. A short overview of the problem background is followed by theoretical concepts that are implemented in practice, such as classic state estimation, linear state estimation incorporating phasor measurements as well as conventional measurements. Optimal placement problem was also addressed. The discussed theory was implemented in MATLAB model to simulate different test cases.

All simulations were made in MATLAB, however the future work can include translating the existing algorithms to another programming language that is used in real measuring and estimating systems.

REFERENCES

[1] *Nordic Grid Code (Nordic collection of rules)*, 2007.

[2] A. J. Wood, B. F. Wollenberg, G. B. Sheble, *Power generation, operation, and control*, 3rd ed., Wiley, 2014.

[3] A. G. Phadke, J. S. Thorp, *Synchronized Phasor Measurements and Their Applications.* New York: Springer, 2008.

[4] *"Phasor measurement unit"* [Online]. Available: https://en.wikipedia.org/wiki/Phasor_measurement_unit [Accessed 03.07.2016]

[5] U.S. Department of Energy, *Factors Affecting PMU Installation Costs*. Oct. 2014.

[6] V. Presada, M. Eremia, L. Toma, "Modified state estimation in presence of PMU measurements", *UPB Scientific Bulletin, Series C: Electrical Engineering*, vol. 76(1), pp 237–248, Jan. 2014.

[7] A. Monticelli, "Electric power system state estimation", *Proceedings of the IEEE*, vol. 88, no. 2, pp 262–282, Feb. 2000.

[8] J. D. Glover, M. S. Sarma, T. J. Overbye, *Power System Analysis and Design*, 5th ed., Cengage Learning, 2012.

[9] Y. Chen, *YBUS Admittance Matrix Formulation* [Online] NPPL, Dec. 2015. Available: https://www.gridpack.org/wiki/images/7/7e/Ybus.pdf [Accessed 03.07.2016]

[10] M. Zhou, V. Centeno, J. S. Thorp and A. G. Phadke, "An Alternative for Including Phasor Measurements in State Estimators", *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1930–1937, Nov. 2006.

[11] S. Soni, S. Bhil, D. Mehta, S. Wagh, "Linear state estimation model using phasor measurement unit (PMU) technology", *Electrical Engineering, Computing Science and Automatic Control (CCE), 2012 9th International Conference*, pp 1–6, Sep. 2012.

[12] J. De La Ree, V. Centeno, J. S. Thorp and A. G. Phadke, "Synchronized Phasor Measurement Applications in Power Systems", *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 20–27, June 2010.

[13] D. Kajjam, K. R. Mekala, *Phasor Measurement Unit or Synchrophasors* [Online] The BEST Group. Available: http://best.eng.buffalo.edu/Teaching/EE611/Phasor_Measurement_Unit.pdf [Accessed 03.07.2016]

[14] R. F. Nuqui and A. G. Phadke, "Hybrid linear state estimation utilizing synchronized phasor measurements", *Power Tech, 2007 IEEE Lausanne*, pp 1665–1669, Jul. 2007.

[15] R. F. Nuqui and A. G. Phadke, "Phasor Measurement Unit Placement Techniques for Complete and Incomplete Observability", *IEEE Transactions on Power Delivery*, vol. 20, no. 4, pp 2381–2388, Oct. 2005.

[16] B. Gou, "Generalized integer linear programming formulation for optimal PMU placement", *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp 1099–1104, Aug. 2008.

[17] B. Gou, "Optimal placement of PMUs by integer linear programming", *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp 1525–1526, Aug. 2008.

[18] B. Xu and A. Abur, "Observability analysis and measurement placement for systems with PMUs", *Power System Conference and Exposition*, vol. 2, pp 943–946, Oct. 2004.

[19] *"MATLAB Documentation – MathWorks Nordic"* [Online]. Available: http://se.mathworks.com/help/matlab/index.html [Accessed 03.07.2016]

[20] *"Power Systems Test Case Archive - UWEE"* [Online]. Available: http://www.ee.washington.edu/research/pstca/ [Accessed 03.07.2016]

[21] *"Power System State Estimation using WLS by Praviraj PG" [Online]. Available*:

http://www.mathworks.com/matlabcentral/fileexchange/23052-power-system-state-estimation-using-wls [Accessed 03.07.2016]

APPENDIX A. MATLAB MODEL SCRIPT

```
clear all;
clc;

% 1. CONVENTIONAL MEASUREMENTS SET-UP
% ========================================
% This section automatically creates the measurements file 'zdatas.m'
% from bus matrix, according to specified measurement positions and
% specified measurement errors.

global bus branch nbus nbra ybus bbus zdata;

% Set up bus and branch matrices, pick up from corresponding data file
% (ieee14, ieee30, ieee57, ieee118):
[bus, branch] = ieee14;
nbus = size(bus,1);     % Get number of buses
nbra = size(branch,1);  % Get number of branches

% Get the Ybus matrix:
ybus = getybus();

% Get the Bbus matrix:
bbus = getbbus();

% Set up voltage magnitude measurement positions (buses) and measurement
% variances:
volt = [1];
% ...or use the following to pick slack bus and PV buses from bus matrix:
%       volt = find(bus(:,2) >= 2);
Rvolt = 9e-4 * ones(length(volt),1);

% Set up injection measurement positions (buses) and measurement
% variances:
inj = [2 3 7 8 9 10 11 12 13 14];
%inj = [4 5 6 8 10 11 12 14 15 16 18 20 22 23 24 25 26 27 29]; % ieee30
% ...or use the following to pick N random buses:
%       rng;
%       inj = sort(randperm(nbus, N));
Rinj = 1e-4 * ones(length(inj),1);

% Set up flow measurement positions (branches) and measurement variances:
flow = [1 3 4 8 9 5 7 10 13 15 11 19];
%flow = [1 3 5 6 8 10 11 12 13 15 16 17 19 20 21 25 26 28 29 30 31 32 ...
%    34 36 37 38 39 41]; % ieee30
% ...or use the following to pick N random branches:
%       rng;
%       flow = sort(randperm(nbra, N));
Rflow = 64e-6 * ones(length(flow),1);

createZdatas(volt, Rvolt, inj, Rinj, flow, Rflow)

% Set up a matrix with conventional measurements from the file:
zdata = zdatas();

clear volt inj flow Rvolt Rinj Rflow;

% ========================================
```

```
% Function creates and fills the file 'zdatas.m' with conventional
% measurements.

function [] = createZdatas(volt, Rvolt, inj, Rinj, flow, Rflow)

global bus branch ybus;

% Create a file 'zdatas.m':
fid = fopen('1 Measurements Set-up\zdatas.m','w');
fprintf(fid, '%s\n%s\n%s\n%s\n%s\n%s\n\n%s\n\n%s\n%s\n',...
'% Measurement data', ...
'% ============================', ...
'% Type of measurement:', ...
'% 1 - Voltage magnitude;          4 - Real power flow;', ...
'% 2 - Real power injection;         5 - Reactive power flow.', ...
'% 3 - Reactive power injection;', ...
'function zdata = zdatas()', ...
'zdata   = [',...
'%          |Msnt |Type | Value | From | To | Rii |');

count = 1;

% Print voltage magnitude measurements:
fprintf(fid, '%s\n',...
'           %---- Voltage Magnitude -------------%');
type = 1;
rng;       % Randomize...
for i = 1:length(volt)
    % Generate a random value from a normal distribution with actual
    % voltage as a mean value and R as variance:
    value = sqrt(Rvolt(i)).*randn(1) + bus(volt(i),3);
    from = volt(i);
    to = 0;
    R = Rvolt(i);
    fprintf(fid, '%13d %5d %8.3f %5d %5d %9g%s\n',...
        count, type, value, from, to, R,';');
    count = count + 1;
end
fprintf(fid, '%s\n',...
'           %-------------------------------------%');

% Print power injection measurements:
fprintf(fid, '%s\n',...
'           %---- Real Power Injection -----------%');
type = 2;
rng;
for i = 1:length(inj)
    value = sqrt(Rinj(i)).*randn(1) + 0.01*(bus(inj(i),7)-bus(inj(i),5));
    from = inj(i);
    to = 0;
    R = Rinj(i);
    fprintf(fid, '%13d %5d %8.3f %5d %5d %9g%s\n',...
        count, type, value, from, to, R,';');
    count = count + 1;
end
fprintf(fid, '%s\n',...
'           %-------------------------------------%');

fprintf(fid, '%s\n',...
'           %---- Reactive Power Injection -------%');
type = 3;
for i = 1:length(inj)
    value = sqrt(Rinj(i)).*randn(1) + 0.01*(bus(inj(i),8)-bus(inj(i),6));
```

```
    from = inj(i);
    to = 0;
    R = Rinj(i);
    fprintf(fid, '%13d %5d %8.3f %5d %5d %9g%s\n',...
        count, type, value, from, to, R,';');
    count = count + 1;
end
fprintf(fid, '%s\n',...
'           %-------------------------------------%');

% Print power flow measurements:
fprintf(fid, '%s\n',...
'           %------ Real Power Flow ------------- %');
type = 4;
rng;
for i = 1:length(flow)
    from = branch(flow(i),1);
    to = branch(flow(i),2);
    R = Rflow(i);
    % Calculate the actual power flow according to the bus data:
    vs = bus(from,3).* exp(1j*deg2rad(bus(from,4)));
    vr = bus(to,3).* exp(1j*deg2rad(bus(to,4)));
    value = real(vr * conj(ybus(from,to)) * (vs - vr));
    value = sqrt(Rflow(i)).*randn(1) + value;
    fprintf(fid, '%13d %5d %8.3f %5d %5d %9g%s\n',...
        count, type, value, from, to, R,';');
    count = count + 1;
end
fprintf(fid, '%s\n',...
'           %-------------------------------------%');

fprintf(fid, '%s\n',...
'           %------ Reactive Power Flow --------- %');
type = 5;
for i = 1:length(flow)
    from = branch(flow(i),1);
    to = branch(flow(i),2);
    R = Rflow(i);
    % Calculate the actual power flow according to the bus data:
    vs = bus(from,3).* exp(1j*deg2rad(bus(from,4)));
    vr = bus(to,3).* exp(1j*deg2rad(bus(to,4)));
    value = imag(vr * conj(ybus(from,to)) * (vs - vr));
    value = sqrt(Rflow(i)).*randn(1) + value;
    fprintf(fid, '%13d %5d %8.3f %5d %5d %9g%s\n',...
        count, type, value, from, to, R,';');
    count = count + 1;
end

fprintf(fid, '%s\n%s\n',...
'           %-------------------------------------%',...
'           ];');

fclose(fid);
disp('File "zdatas.m" was updated!');

end
```

```
% Function returns shunt susceptance matrix Bbus.

function bbus = getbbus()

global branch nbus nbra;

fb = branch(:,1);        % From bus
tb = branch(:,2);        % To bus
b = branch(:,5);         % Charging susceptance, B/2 (pu)

% Populate Bbus:
bbus = zeros(nbus,nbus);
 for k=1:nbra
     bbus(fb(k),tb(k)) = b(k);
     bbus(tb(k),fb(k)) = bbus(fb(k),tb(k));
 end



% Function returns admittance matrix Ybus.

function ybus = getybus()

global branch nbus nbra;

fb = branch(:,1);                % From bus
tb = branch(:,2);                % To bus
r = branch(:,3);                 % Resistance, R (pu)
x = branch(:,4);                 % Reactance, X (pu)
b = branch(:,5);                 % Charging susceptance, B/2 (pu)
t = branch(:,6);                 % Turns ratio
for i = 1:nbra
    if t(i) == 0
        t(i) = 1;
    end
end

z = r + 1i*x;                    % Impedance, Z (pu)
y = 1./z;                        % Invert each element
b = 1i*b;                        % Make B imaginary

% Initialize Ybus:
ybus = zeros(nbus,nbus);

% Populate the diagonal elements:
for m =1:nbus
    for n =1:nbra
        if fb(n) == m
             ybus(m,m) = ybus(m,m) + y(n)/(t(n)^2) + b(n);
        elseif tb(n) == m
             ybus(m,m) = ybus(m,m) + y(n) + b(n);
        end
    end
 end

% Populate the off-diagonal elements:
for k=1:nbra
     ybus(fb(k),tb(k)) = ybus(fb(k),tb(k))-y(k)/t(k);
     ybus(tb(k),fb(k)) = ybus(fb(k),tb(k));
end
```

```
% 2. CLASSIC STATE ESTIMATION
% ==========================================
% This section estimates the state variables using non-linear weighted
% least squares method (Newton-Raphson method):

global nbus Xest;

% Estimate the power system state:
[th_est, V_est] = WLS();

Xest = V_est.*exp(1j*deg2rad(th_est));      % Estimated state vector

disp('WLS State Estimation');
disp('  Bus | V (pu) |  Angle (Deg)');
disp('--------------------------');
for i = 1:nbus
    fprintf('%4g', i); fprintf('  %8.4f', V_est(i)); fprintf('   %8.4f',...
        th_est(i)); fprintf('\n');
end
fprintf('\n');
clear i;

% NB. If there is singularity warning, consider increasing the number of
% conventional measurements in (1).

% ==========================================



% Function returns two vectors of state variables (angles and voltages)
% after the state estimation using Weighted Least Squares method.

function [Th, V] = WLS()

global zdata nbus ybus bbus;

set_tol = 1e-4;             % Set tolerance for iterations

type = zdata(:,2);          % Type of measurement:
% 1 - Voltage measurement;          4 - Real power flow;
% 2 - Real power injection;         5 - Reactive power flow.
% 3 - Reactive power injection;
z = zdata(:,3);             % Measuement values
fbus = zdata(:,4);          % From bus
tbus = zdata(:,5);          % To bus
Ri = diag(zdata(:,6));      % Make diagonal matrix of covariances
V = ones(nbus,1);           % Initialize the bus voltages
th = zeros(nbus,1);         % Initialize the bus angles (theta)
X = [th(2:end); V];         % State Vector
G = real(ybus);
B = imag(ybus);

vm = find(type == 1);    % Indices of voltage magnitude measurements
pin = find(type == 2);   % ---------- real power injection measurements
qin = find(type == 3);   % ---------- reactive power injection measurements
pf = find(type == 4);    % ---------- real power flow measurements
qf = find(type == 5);    % ---------- reactive power flow measurements

nvi = length(vm);        % Number of voltage measurements
npin = length(pin);      % --------- real power injection measurements
nqin = length(qin);      % --------- reactive power injection measurements
npf = length(pf);        % --------- real power flow measurements
```

```
nqf = length(qf);        % --------- reactive power flow measurements

iter = 1;
tol = 5;

while(tol > set_tol)

    % Measurement Function, h
    h1 = V(fbus(vm),1);
    h2 = zeros(npin,1);
    h3 = zeros(nqin,1);
    h4 = zeros(npf,1);
    h5 = zeros(nqf,1);

    for i = 1:npin
        m = fbus(pin(i));
        for k = 1:nbus
            h2(i) = h2(i) + V(m)*V(k)*(G(m,k)*cos(th(m)-th(k)) + ...
                B(m,k)*sin(th(m)-th(k)));
        end
    end

    for i = 1:nqin
        m = fbus(qin(i));
        for k = 1:nbus
            h3(i) = h3(i) + V(m)*V(k)*(G(m,k)*sin(th(m)-th(k)) - ...
                B(m,k)*cos(th(m)-th(k)));
        end
    end

    for i = 1:npf
        m = fbus(pf(i));
        n = tbus(pf(i));
        h4(i) = -V(m)^2*G(m,n) - V(m)*V(n)*(-G(m,n)*cos(th(m)-th(n)) - ...
            B(m,n)*sin(th(m)-th(n)));
    end

    for i = 1:nqf
        m = fbus(qf(i));
        n = tbus(qf(i));
        h5(i) = -V(m)^2*(-B(m,n)+bbus(m,n)) - V(m)*V(n)*(-G(m,n)* ...
            sin(th(m)-th(n)) + B(m,n)*cos(th(m)-th(n)));
    end

    h = [h1; h2; h3; h4; h5];

    % Residual
    r = z - h;

    % Jacobian
    % H11 - Derivative of V with respect to theta
    H11 = zeros(nvi,nbus-1);

    % H12 - Derivative of V with respect to V
    H12 = zeros(nvi,nbus);
    for k = 1:nvi
        for n = 1:nbus
            if n == k
                H12(k,n) = 1;
            end
        end
    end
```

```
% H21 - Derivative of real power injections with respect to theta
H21 = zeros(npin,nbus-1);
for i = 1:npin
    m = fbus(pin(i));
    for k = 1:(nbus-1)
        if k+1 == m
            for n = 1:nbus
                H21(i,k) = H21(i,k) + V(m)* V(n)*(-G(m,n)* ...
                    sin(th(m)-th(n)) + B(m,n)*cos(th(m)-th(n)));
            end
            H21(i,k) = H21(i,k) - V(m)^2*B(m,m);
        else
            H21(i,k) = V(m)* V(k+1)*(G(m,k+1)*sin(th(m)-th(k+1)) - ...
                B(m,k+1)*cos(th(m)-th(k+1)));
        end
    end
end

% H22 - derivative of real power injections with respect to V
H22 = zeros(npin,nbus);
for i = 1:npin
    m = fbus(pin(i));
    for k = 1:(nbus)
        if k == m
            for n = 1:nbus
                H22(i,k) = H22(i,k) + V(n)*(G(m,n)*cos(th(m)-th(n)) ...
                    + B(m,n)*sin(th(m)-th(n)));
            end
            H22(i,k) = H22(i,k) + V(m)*G(m,m);
        else
            H22(i,k) = V(m)*(G(m,k)*cos(th(m)-th(k)) + B(m,k)* ...
                sin(th(m)-th(k)));
        end
    end
end

% H31 - Derivative of reactive power injections with respect to theta
H31 = zeros(nqin,nbus-1);
for i = 1:nqin
    m = fbus(qin(i));
    for k = 1:(nbus-1)
        if k+1 == m
            for n = 1:nbus
                H31(i,k) = H31(i,k) + V(m)* V(n)*(G(m,n)* ...
                    cos(th(m)-th(n)) + B(m,n)*sin(th(m)-th(n)));
            end
            H31(i,k) = H31(i,k) - V(m)^2*G(m,m);
        else
            H31(i,k) = V(m)* V(k+1)*(-G(m,k+1)*cos(th(m)-th(k+1)) - ...
                B(m,k+1)*sin(th(m)-th(k+1)));
        end
    end
end

% H32 - Derivative of reactive power injections with respect to V
H32 = zeros(nqin,nbus);
for i = 1:nqin
    m = fbus(qin(i));
    for k = 1:(nbus)
        if k == m
            for n = 1:nbus
                H32(i,k) = H32(i,k) + V(n)*(G(m,n)*sin(th(m)-th(n)) ...
```

```
                    - B(m,n)*cos(th(m)-th(n)));
            end
            H32(i,k) = H32(i,k) - V(m)*B(m,m);
        else
            H32(i,k) = V(m)*(G(m,k)*sin(th(m)-th(k)) - B(m,k)* ...
                cos(th(m)-th(k)));
        end
    end
end

% H41 - Derivative of real power flows with theta
H41 = zeros(npf,nbus-1);
for i = 1:npf
    m = fbus(pf(i));
    n = tbus(pf(i));
    for k = 1:(nbus-1)
        if k+1 == m
            H41(i,k) = V(m)* V(n)*(-G(m,n)*sin(th(m)-th(n)) + ...
                B(m,n)*cos(th(m)-th(n)));
        else if k+1 == n
            H41(i,k) = -V(m)* V(n)*(-G(m,n)*sin(th(m)-th(n)) + ...
                B(m,n)*cos(th(m)-th(n)));
            else
                H41(i,k) = 0;
            end
        end
    end
end

% H42 - Derivative of real power flows with V
H42 = zeros(npf,nbus);
for i = 1:npf
    m = fbus(pf(i));
    n = tbus(pf(i));
    for k = 1:nbus
        if k == m
            H42(i,k) = -V(n)*(-G(m,n)*cos(th(m)-th(n)) - ...
                B(m,n)*sin(th(m)-th(n))) - 2*G(m,n)*V(m);
        else if k == n
            H42(i,k) = -V(m)*(-G(m,n)*cos(th(m)-th(n)) - ...
                B(m,n)*sin(th(m)-th(n)));
            else
                H42(i,k) = 0;
            end
        end
    end
end

% H51 - Derivative of reactive power flows with theta
H51 = zeros(nqf,nbus-1);
for i = 1:nqf
    m = fbus(qf(i));
    n = tbus(qf(i));
    for k = 1:(nbus-1)
        if k+1 == m
            H51(i,k) = -V(m)* V(n)*(-G(m,n)*cos(th(m)-th(n)) - ...
                B(m,n)*sin(th(m)-th(n)));
        else if k+1 == n
            H51(i,k) = V(m)* V(n)*(-G(m,n)*cos(th(m)-th(n)) - ...
                B(m,n)*sin(th(m)-th(n)));
            else
                H51(i,k) = 0;
            end
        end
```

```
        end
    end

    % H52 - Derivative of reactive power flows with V
    H52 = zeros(nqf,nbus);
    for i = 1:nqf
        m = fbus(qf(i));
        n = tbus(qf(i));
        for k = 1:nbus
            if k == m
                H52(i,k) = -V(n)*(-G(m,n)*sin(th(m)-th(n)) + ...
                    B(m,n)*cos(th(m)-th(n))) - 2*V(m)*(-B(m,n)+ bbus(m,n));
            else if k == n
                H52(i,k) = -V(m)*(-G(m,n)*sin(th(m)-th(n)) + ...
                    B(m,n)*cos(th(m)-th(n)));
                else
                    H52(i,k) = 0;
                end
            end
        end
    end

    % Measurement Jacobian, H
    H = [H11 H12; H21 H22; H31 H32; H41 H42; H51 H52];

    % State Vector
    dX = (H'*(Ri\H))\(H'*(Ri\r));
    X = X + dX;
    th(2:end) = X(1:nbus-1);
    V = X(nbus:end);
    iter = iter + 1;
    tol = max(abs(dX));

end

Th = 180/pi*th;          % Convert radians to degrees

end
```

```
% 3. OPTIMAL PLACEMENT PROBLEM
% ========================================
% This section produces optimal placement schemes for PMUs considering
% branch topology of the system, depth of unobservability and conventional
% measurements.

global Tpmu nbus nbra A pmu0 pmu0conv pmu1 pmu1conv pmu2;

% COMPLETE OBSERVABILITY CASE
% Set up the connectivity matrix Tpmu:
Tpmu = getTpmu();

% Calculate the optimal positions of PMUs for exclusive PMU measurements:
pmu0 = optimal(1, ones(nbus,1));

% Set up conventional measurement positions:
inj2 = [1];     % Buses with injection measurements
flow2 = [1];    % Branches with flow measurements

% Set up connectivity matrix Tcon, permutation matrix P and vector bcon:
[Tcon, P, bcon] = getTcon(inj2, flow2);

% Calculate the optimal positions of PMUs for the case when PMU
% measurements complement conventional measurements:
pmu0conv = optimal(Tcon*P, bcon);

fprintf('\n');
disp('============================');
disp('COMPLETE OBSERVABILITY CASE');
fprintf('\n');
disp('No conventional measurements:');
fprintf('   PMU locations: '); fprintf('%d ',(find(pmu0 == 1)).'); ...
    fprintf('\n');
fprintf('   Number of PMUs: '); fprintf('%d ', ...
    length((find(pmu0 == 1)).')); fprintf('\n');
disp('With conventional measurements:');
fprintf('   PMU locations: '); fprintf('%d ',(find(pmu0conv == 1)).'); ...
    fprintf('\n');
fprintf('   Number of PMUs: '); fprintf('%d ', ...
    length((find(pmu0conv == 1)).')); fprintf('\n');


% ---------------------------------
% DEPTH-OF-ONE UNOBSERVABILITY CASE
% ---------------------------------
% Set up branch-to-node incident matrix A:
A = getA();

% Calculate the optimal positions of PMUs for Do1 unobservability
% (no injections measurements):
pmu1 = optimal(A, ones(nbra,1));

% Determine the number of unobserved buses:
nuob1 = unobserved(A, pmu1);

% Set up matrix P1 which removes branches associated to zero injection
% measurements:
P1 = getP1(inj2);

% Calculate the optimal positions of PMUs for Do1 unobservability
% (with injections measurements):
pmu1conv = optimal(P1*A, P1*ones(nbra,1));
```

```
% Determine the number of unobserved buses:
nuob1conv = unobserved(A, pmu1conv);

fprintf('\n');
disp('==================================');
disp('DEPTH-OF-ONE UNOBSERVABILITY CASE');
fprintf('\n');
disp('No conventional measurements:');
fprintf('   PMU locations: '); fprintf('%d ',(find(pmu1 == 1)).'); ...
    fprintf('\n');
fprintf('   Number of PMUs: '); fprintf('%d ', ...
    length((find(pmu1 == 1)).')); fprintf('\n');
fprintf('   Number of unobserved buses: '); fprintf('%d ',nuob1); ...
    fprintf('\n');
disp('With conventional measurements:');
fprintf('   PMU locations: '); fprintf('%d ',(find(pmu1conv == 1)).'); ...
    fprintf('\n');
fprintf('   Number of PMUs: '); fprintf('%d ', ...
    length((find(pmu1conv == 1)).')); fprintf('\n');
fprintf('   Number of unobserved buses: '); ...
    fprintf('%d ',nuob1conv); fprintf('\n');


% ----------------------------------
% DEPTH-OF-TWO UNOBSERVABILITY CASE
% ----------------------------------
% Calculate the optimal positions of PMUs for Do2 unobservability
% (no injections measurements):
pmu2 = optimal2(2, 1, ones(nbus,1), pmu1);

% Determine the number of unobserved buses:
nuob2 = unobserved(A, pmu2);


fprintf('\n');
disp('==================================');
disp('DEPTH-OF-TWO UNOBSERVABILITY CASE');
fprintf('\n');
disp('No conventional measurements:');
fprintf('   PMU locations: '); fprintf('%d ',(find(pmu2 == 1)).'); ...
    fprintf('\n');
fprintf('   Number of PMUs: '); fprintf('%d ', ...
    length((find(pmu2 == 1)).')); fprintf('\n');
fprintf('   Number of unobserved buses: '); fprintf('%d ',nuob2); ...
    fprintf('\n');

% ==========================================
```

```
% Function returns branch-to-node incident matrix A.

function A = getA()

global nbus nbra branch;

A = zeros(nbra, nbus);      % Initialize A
fb = branch(:,1);           % From bus
tb = branch(:,2);           % To bus
for i = 1:nbra
    A(i, fb(i)) = 1;
    A(i, tb(i)) = 1;
end

end



% Function returns matrix P1 which removes branches associated to zero
% injection measurements.

function P1 = getP1(inj)

global A nbra;

npin = length(inj);             % Number of power injection measurements

% Determine all branches not associated to zero injection buses:
s = zeros(nbra,1);
for i = 1:npin
    s = s + A(:, inj(i));
end

% Zero values of 's' indicate branches that are not associated to any of
% zero injection buses:
na_branches = find(s == 0);

% Populate matrix P1:
P1 = zeros(length(na_branches), nbra);
for i = 1:length(na_branches)
    P1(i, na_branches(i)) = 1;
end

end
```

```
% Function returns a connectivity matrix Tpmu.

function Tpmu = getTpmu()

global branch nbra nbus;

% Initialize Tpmu:
Tpmu = ones(nbus,1);
Tpmu = diag(Tpmu);

% Populate non-diagonal elements:
fb = branch(:,1);       % From bus
tb = branch(:,2);       % To bus
for i = 1:nbra
    Tpmu(fb(i),tb(i)) = 1;
    Tpmu(tb(i),fb(i)) = 1;
end

end
```

```
% Function returns a connectivity matrix Tcon.

function [Tcon, P, bcon] = getTcon(inj, flow)

global nbus Tpmu branch;

npf = length(flow);             % Number of power flow measurements
npin = length(inj);             % Number of power injection measurements
fbus = zeros(npf + npin, 1);    % From bus
tbus = fbus;                    % To bus
for i = 1:npf
    fbus(i) = branch(flow(i), 1);
    tbus(i) = branch(flow(i), 2);
end
for i = 1:npin
    fbus(npf + i) = inj(i);
end

% Initialize Tmeas and bmeas:
Tmeas = zeros(npin + npf, nbus);
bmeas = ones(npin + npf, 1);

% Set up matrix Tmeas.
% Power flow measurements bus incidence:
for i = 1:npf
    Tmeas(i, fbus(i)) = 1;
    Tmeas(i, tbus(i)) = 1;
end

% Power injection measurements bus incidence:
for i = 1:npin
    for k = 1:nbus
        Tmeas(npf + i, k) = Tpmu(fbus(npf + i), k);
        bmeas(npf + i) = bmeas(npf + i) + Tmeas(npf + i, k);
    end
    bmeas(npf + i) = bmeas(npf + i) - 2;
end
```

```
% Find indices of buses not associated to conventional measurements
% (zero columns in Tmeas):
na_buses = find(any(Tmeas,1) == 0);
M = length(na_buses);

% Remove zero columns from Tmeas:
Tmeas = Tmeas(:, any(Tmeas,1));

% Set up permutation matrix P:
P = zeros(nbus);
for i = 1:M
    P(i, na_buses(i)) = 1;
end
k = M + 1;
for i = 1:nbus
    if isempty(find(na_buses == i))
        P(k,i) = 1;
        k = k + 1;
    end
end

Tcon = blkdiag(eye(M),Tmeas);
bcon = [ones(M,1); bmeas];

end



% Function returns a vector of optimal PMU positions according to
% requirements specified by matrix G and redundancy requirements specified
% by vector Bg

function pmu = optimal(G, Bg)

global nbus Tpmu;

% Solve mixed-integer linear programming with linear inequalities:
f = ones(nbus,1);       % Objective function;
intcon = 1:nbus;        % Vector of integer variables;
A = -1*G*Tpmu;          % Inequality coefficients in
b = -1*Bg;              %   form A*x <= b;
Aeq = zeros(nbus);
beq = zeros(nbus,1);
lb = zeros(nbus,1);     % Bounds to produce binary
ub = ones(nbus,1);      %   values.
options = optimoptions('intlinprog','Display','off','LPPreprocess','none');

pmu = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,options);

end
```

```
% Function returns a vector of optimal PMU positions with depth N
% unobservability according to linear inequality constraints specified by
% matrix G and vector Bg and optimal placement positions (x0) of N-1 depth.

function pmu = optimal2(N, G, Bg, x0)

global nbus Tpmu;

% Solve mixed-integer linear programming with linear inequalities:
f = ones(nbus,1);       % Objective function;
intcon = 1:nbus;        % Vector of integer variables;
A = -1*G*Tpmu^(N+1);    % Inequality coefficients in
b = -1*Bg;              %   form A*x <= b;
Aeq = (f - x0)';        % Equality constraints
beq = 0;
lb = zeros(nbus,1);     % Bounds to produce binary
ub = ones(nbus,1);      %   values.
options = optimoptions('intlinprog','Display','off');

pmu = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,options);

end
```

```
% Function returns a number of unobserved buses (depth-of-one) in a system
% determined by branch-to-node matrix M and PMU position determined by
% vector 'pmu'.

function nuob = unobserved(M, pmu)

global nbus;

p = find(pmu == 1);               % Indices of buses with PMUs
s = zeros(1, nbus);
for i = 1:length(p)
    br = find(M(:,p(i)) == 1);    % Indices of branches connected to
    br = br.';                    %    i-th PMU
    s = s + sum(M(br, :));
end

uob = find(s == 0);     % Determine indices of unobserved buses
nuob = length(uob);     % Determine the total number of unobserved buses

end
```

```
% 4. PMU MEASUREMENTS SET-UP
% ==========================================
% This section creates the measurements file 'PMUdatas.m' of PMU
% measurements from the bus matrix, according to specified PMU positions
% and specified measurement errors.

global pmu0 pmu0conv pmu1 pmu1conv pmu2 pmudata nbus;

pmu = pmu0;       % Set PMU incidence vector from section 3
% pmu0, pmu0conv - for complete observability
% pmu1, pmu1conv - for Do1 unobservability
% pmu2           - for Do2 unobservability
% ...or use the following to install PMUs to all buses (redundancy):
%   pmu = ones(1, nbus);

pmubus = find(pmu == 1);                % Indices of buses with PMUs
Rpmu = 1e-6 * ones(length(pmubus),1);   % Set PMU variances

% Get PMU measurements from specified PMU placement positions:
createPMUdatas(pmubus, Rpmu);
pmudata = PMUdatas();

% ==========================================



% Function creates and fills the file 'PMUdatas.m' with PMU voltage and
% current measurements.

function [] = createPMUdatas(pmubus, Rpmu)

global bus Tpmu ybus;

% Create a file 'PMUdatas.m':
fid = fopen('4 PMU Measurements Set-up\PMUdatas.m','w');
fprintf(fid, '%s\n%s\n%s\n%s\n%s\n\n%s\n\n%s\n%s\n',...
'% PMU measurement data', ...
'% ============================', ...
'% Type of measurement:', ...
'% 1 - Voltage measurement;', ...
'% 2 - Current measurement;', ...
'function pmudata = PMUdatas()', ...
'pmudata   = [',...
'%   |Msnt |Type |          Value          | From | To | Rii |');

count = 1;


% Print voltage measurements:
fprintf(fid, '%s\n',...
'%--------------------- Voltage ---------------------------------%');
type = 1;
to = 0;
rng;       % Randomize...
for i = 1:length(pmubus)
    from = pmubus(i);
    value = bus(from,3).* exp(1j*deg2rad(bus(from,4)));
    % Generate a random value from a normal distribution with actual
    % voltage as a mean value and R as variance:
    value = value + (1 + 1j)*sqrt(Rpmu(i)).*randn(1);
    R = Rpmu(i);
    fprintf(fid, '%8d %5d %14.4g%+1.4fi %7d %4d %7g%s\n',...
```

```
        count, type, [real(value) imag(value)], from, to, R,';');
    count = count + 1;
end
fprintf(fid, '%s\n',...
'%------------------------------------------------------------------%');

% Print current measurements:
fprintf(fid, '%s\n',...
'%--------------------- Current ---------------------------------%');
type = 2;
rng;       % Randomize...
for i = 1:length(pmubus)
    from = pmubus(i);
    temp = find(Tpmu(from,:) == 1);     % Find all buses connected to 'from'
    temp = setdiff(temp, from);         % Remove bus 'from'
    if isempty(temp) == 0
        for k = 1:length(temp)
            to = temp(k);
            vs = bus(from,3).* exp(1j*deg2rad(bus(from,4)));
            vr = bus(to,3).* exp(1j*deg2rad(bus(to,4)));
            value = ybus(from,to) * (vs - vr);
            value = value + (1 + 1j)*sqrt(Rpmu(i)).*randn(1);
            R = Rpmu(i);
            fprintf(fid, '%8d %5d %14.4g%+1.4fi %7d %4d %7g%s\n',...
                count, type, [real(value) imag(value)], from, to, R,';');
            count = count + 1;
        end
    end
end

fprintf(fid, '%s\n%s\n',...
'%------------------------------------------------------------------%',...
'];');

fclose(fid);
disp('File "PMUdatas.m" was updated!');

end
```

```
% 5. LINEAR STATE ESTIMATION
% ==========================================
% This section produces the state vector using the direct PMU measurements.
% The PMU measurements can be used exclusively or complemented with WLS-
% estimated state vector.

global nbus;

% Set up current measurement bus incedence matrix:
Apmu = getApmu();

% Set up voltage measurement bus incedence matrix:
II = getII();

% Set up matrix of series admittances:
y = gety();

% Set up matrix of shunt admittances:
ys = getys();

% LINEAR STATE ESTIMATION UTILIZING PMU MEASUREMENTS EXCLUSIVELY:
% Set up system matrix:
B1 = [II; y * Apmu + ys];

% Estimate the state vector:
[th_lse, V_lse] = LSE(B1);

disp('Linear State Estimation utilizing PMU measurements exclusively:');
disp('  Bus | V (pu) |  Angle (Deg)');
disp('--------------------------');
for i = 1:nbus
    fprintf('%4g', i); fprintf('  %8.4f', V_lse(i)); fprintf('   %8.4f', ...
        th_lse(i)); fprintf('\n');
end
fprintf('\n');
clear i;

% LINEAR STATE ESTIMATION UTILIZING BOTH PMU AND CONVENTIONAL MEASUREMENTS:
% Set up system matrix
B2 = [diag(ones(nbus,1)); II; y * Apmu + ys];

% Estimate the state vector:
[th_hyb, V_hyb] = LSEconv(B2);

disp('Linear State Estimation with added conventional measurements:');
disp('  Bus | V (pu) |  Angle (Deg)');
disp('--------------------------');
for i = 1:nbus
    fprintf('%4g', i); fprintf('  %8.4f', V_hyb(i)); fprintf('   %8.4f', ...
        th_hyb(i)); fprintf('\n');
end
fprintf('\n');
clear i;

% ==========================================
```

```
% Function returns the current measurement bus incidence matrix Apmu.

function Apmu = getApmu()

global pmudata nbus;

im = find(pmudata(:,2) == 2);    % Indices of current measurements
nim = length(im);                % Number of current measurements
fbus = pmudata(:,4);             % From bus
tbus = pmudata(:,5);             % To bus

Apmu = zeros(nim, nbus);
for i = 1:nim
    Apmu(i, fbus(im(i))) = 1;
    Apmu(i, tbus(im(i))) = -1;
end

end


% Function returns the voltage measurement bus incidence matrix II.

function II = getII()

global pmudata nbus;

vm = find(pmudata(:,2) == 1);    % Indices of voltage measurements
nvm = length(vm);                % Number of voltage measurements
fbus = pmudata(:,4);             % Measurement bus

II = zeros(nvm, nbus);
for i = 1:nvm
    II(i, fbus(vm(i))) = 1;
end

end




% Function returns the diagonal matrix 'y' of series admittances of
% measured branches.

function y = gety()

global pmudata ybus;

im = find(pmudata(:,2) == 2);    % Indices of current measurements
nim = length(im);                % Number of current measurements
fbus = pmudata(:,4);             % From bus
tbus = pmudata(:,5);             % To bus

y = zeros(nim);
for i = 1:nim
    y(i,i) = ybus(fbus(im(i)), tbus(im(i)));
end

end
```

```
% Function returns the matrix 'ys' of shunt admittances.

function ys = getys()

global pmudata nbus bbus;

im = find(pmudata(:,2) == 2);    % Indices of current measurements
nim = length(im);                % Number of current measurements
fbus = pmudata(:,4);             % From bus
tbus = pmudata(:,5);             % To bus

ys = zeros(nim, nbus);
for i = 1:nim
    ys(i,fbus(im(i))) = bbus(fbus(im(i)), tbus(im(i)));
end

end



% Function returns state vector in the form of complex voltages after
% performing linear state estimation utilizing PMU measurements
% exclusively.

function [Th, V] = LSE(B)

global pmudata;

% Make diagonal matrix of covariances:
W = diag(pmudata(:,6));

% Make the measurements vector:
Z = pmudata(:,3);

% Linear State Estimation:
X = (B'*(W\B))\B'*(W\Z);

Th = rad2deg(angle(X));
V = abs(X);

end
```

```
% Function returns state vector in the form of complex voltages after
% performing linear state estimation utilizing both PMU and conventional
% measurements.

function [Th, V] = LSEconv(B)

global pmudata nbus Xest;

% Make diagonal matrix of covariances:
W1 = diag(1e-4 * ones(nbus,1));    % Covariance of WLS estimated vector
W2 = diag(pmudata(:,6));           % Covariance of PMU measurements
W = blkdiag(W1, W2);

% Make the measurements vector:
Z1 = Xest;
Z2 = pmudata(:,3);
Z = [Z1; Z2];

% Linear State Estimation:
X = (B'*(W\B))\B'*(W\Z);

Th = rad2deg(angle(X));
V = abs(X);

end
```