Utilization of Phasor Measurement Units in Hybrid Linear State Estimation

Simen Karlsen

Master of Energy and Environmental Engineering
Submission date: Januar 2016
Supervisor: Kjetil Uhlen, ELKRAFT

Norwegian University of Science and Technology
Department of Electric Power Engineering

Abstract

The Phasor Measurement Unit (PMU) introduces new functionality to the measurement spectrum, with accurate time tagging and synchronization capability, and is thus the next step in the technological evolution of state estimation in electrical power systems. This thesis presents the theoretical basis behind a two-pass hybrid linear state estimation model, the aim of which is to utilize PMU measurements in conjunction with classical state estimation. Following this, the construction of a hybrid linear state estimator application is discussed. Numerical simulation has been conducted to evaluate the feasibility of the model as a way of utilizing the increasing availability of synchrophasor measurements to improve state estimation. The results indicate a significant potential improvement of state estimation, especially in aspects of monitoring dynamic behaviour within the system.

Sammendrag

Fasormåleenheter (PMU) introduserer ny funksjonalitet blant målinger, med nøyaktig tidsstempling og synkroniseringsevne, og er dermed det neste steget i den teknologiske utviklingen av tilstandsestimering av elektriske kraftsystemer. Denne avhandlingen presenterer den teoretiske basisen for en to-stegs hybrid lineær tilstandsestimatormodell, med mål om å utnytte PMU-målinger i kombinasjon med tradisjonell tilstandsestimering. Videre diskuteres konstruksjonen av en hybrid lineær tilstandsestimatorapplikasjon. Numeriske simuleringer har blitt utført i den hensikt å evaluere gjennomførbarheten til modellen som en måte å utnytte den økende tilgangen på fasormålinger til å forbedre tilstandsestimering. Resultatene indikerer en betydelig potensiell forbedring av tilstandsestimering, spesielt for overvåkning av dynamisk atferd i systemet.

Acknowledgements

This thesis is submitted to the Department of Electrical Power Engineering at the Norwegian University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science. The thesis is written under the supervision of Professor Kjetil Uhlen.

I wish to extend my sincere gratitude to my supervisor, Professor Kjetil Uhlen, for his valuable insight and guidance throughout the work on this thesis, and for the opportunity to work on this interesting and challenging topic.

I would also like to thank my co-supervisor, Dr. Eilert Bjerkan. Thank you for introducing me to this topic and for your ever-present guidance and encouragement.

My final thanks goes to my friends, for interesting discussions, useful advice and great camaraderie over the course of this past year.

Contents

List of Tables

List of Figures

Abbreviations and Acronyms

CSE	Classical State Estimation
EMS	Energy Management System
GPS	Global Positioning System
HLSE	Hybrid Linear State Estimator
IED	Intelligent Electronic Device
PDC	Phasor Data Concentrator
PMU	Phasor Measurement Unit
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
TSO	Transmission System Operator
TVE	Total Vector Error

1 | Introduction

1.1 Background

Over the course of the 20th century, the presence of electricity became increasingly prevalent, especially in the more developed parts of the world. This development has not seized after the passing of the last century, giving weight to the slogan of the Norwegian TSO, Statnett: "*The future is electric*", and there are no indications suggesting otherwise. Today's modern society has reached a point where the use of electric power has become an integral part of daily life. The importance of the electric power supply is not only present in aspects of individual comfort, and has also time and time again been identified as a critical factor for national security and civil protection. The Norwegian white paper nr. 22 (2007-2008) states the following. *A stable and effective power system is a prerequisite for the societal security in Norway* [1].

The integration of renewable and distributed energy sources will in the time ahead lead to more extremes and uncertainties in the electric power system, making the continuous and reliable operation of the system increasingly challenging.

Given the above, equipping system operators with good tools for monitoring, analyzing, optimizing and forecasting the power system is becoming increasingly important.

1.1.1 History of the PMU

Voltage phasors in the power system have long been known to be a reliable way of calculating active power flow. In the early 1980s, attempts were made to directly measure bus voltage angle differences. The methods involved use of radio transmission to establish synchronization between geographical locations and registering voltage zero-crossing times. These early systems achieved decent measurement accuracies given the available technology, but no filtering of disruptive frequencies were present and only single-phase applications were attempted. [2]

In 1983, the first paper outlining the use of positive-sequence voltage and current phasor measurements was published by A. G. Phadke and J. S. Thorp [3]. The paper presented the utilization of the newly established Global Positioning System (GPS) for highly precise time synchronization between units deployed over vast geographical distances. Along with the paper, a prototype of the modern *phasor measurement unit* (PMU) was constructed. [2]

The PMU has since been further developed as a measurement tool and is now being implemented into power systems world-wide. Over the years, many technologies have been implemented into the state estimator, which today has become an integral part of every power system control center around the globe. The phasor measurement unit is the next step in this technological evolution, bringing new opportunities to the table that aim to increase the quality of system state estimation.

1.1.2 History of State Estimation

Many tools have been developed to aid system operators in maintaining the operational security, efficiency and reliability of the electric power system. The state estimator was introduced in the early 1970s, and has been an effective and important tool for achieving monitoring and continuous security assessments from centralized control rooms. Since its infancy, the fields of computational power, communications technology and power system engineering have seen great advances, and as a result the state estimator has evolved with the access to new means of improving the resulting estimate.

In January 1970, Schweppe and Wildes proposed methods of implementing state estimation in power systems [4]. This made it possible for more accurate and extensive analysis of Supervisory Control and Data Acquisition (SCADA)-collected data from Remote Terminal Units (RTU) and, later, Intelligent Electronic Devices (IED). This led to the introduction of the Energy Management System (EMS), which facilitates the use of numerous analysis and planning applications such as contingency analysis, optimal power flow and automatic generation control [5].

Today, functionality vary between different state estimators, but a common set of applications includes the following.

Topology processor
Gathers the statuses of switches and circuit breakers to continuously create updated one-line diagrams of the system.

Observability analysis
Evaluates the observability of the system given the available information. Identifies unobservable sections of the system as well as observable islands.

State estimation solution
Calculates the estimate of the system state. May also provide estimates for other system parameters, like line currents and load flows.

Bad data processing
Provided there is enough measurement redundancy, the state estimator can identify greater errors in measurements and eliminate these from the state estimation.

Parameter and structural error processing
Provided there is enough measurement redundancy, estimates physical network parameters.

1.2 Objective of the Thesis

The work conducted in this thesis will be twofold. Firstly, the theoretical basis of both the modern PMU and classical state estimation will be outlined, followed by an introduction to the concepts of hybrid linear state estimation. A hybrid estimator will then be constructed based on the aforementioned theoretical aspects.

The second part of the work presented in this thesis will consist of a feasibility study of the hybrid linear state estimator model as a way of utilizing the increasing availability of synchrophasor measurements to improve state estimation. This will be done through numerical simulations, with the aim of identify the possible benefits of introducing PMU technology into state estimation.

The simulations will be conducted on a network model representing parts of the Norwegian transmission grid. In addition to a general assessment of the hybrid estimator model, its effectiveness and potential integration in the Norwegian system will be evaluated.

2 | The Phasor Measurement Unit

This chapter aims to introduce the mathematical basis of the modern phasor measurement unit, as well as a brief summary of its hardware and the communication systems connecting the individual unit to the centralized monitoring system.

2.1 Phasor Representation of Sinusoids

Consider a sinusoidal wave given by the following

$$x(t) = X_m \cos(\omega t + \phi). \tag{2.1}$$

In the above equation $\mathrm{X_m}$ is the peak amplitude of the signal. ω and ϕ are the signal frequency [$\frac{\mathrm{rad}}{\mathrm{s}}$] and the phase angle [rad] to a reference, respectively. The phase angle describes the phase-shifting of the signal compared to a sinusoid with its peak at $t_0 = 0$ seconds. Studying the time at which the signal reaches its peak, t_{peak}, ϕ is defined as positive when the $t_{peak} < t_0$. Equation 2.1 is an expression of the instantaneous value of the signal at any time *t*. The sinusoid can, however, also be represented as a time-independent complex quantity, **X**, as shown in Equation 2.2 below. [2] [6]

$$\mathbf{X} = X_m[\cos(\phi) + j\, sin(\phi)] \tag{2.2}$$

X is known as the *phasor representation* of the signal, and is a vector with magnitude corresponding to the peak value of the signal and angle difference from the real axis equal to the phase angle ϕ outlined above. An illustration depicting the relationship between a sinusoidal signal and its phasor representation is shown in Figure 2.1. [2]

For use in AC circuitry it is convenient to employ the root mean square of the signal, resulting in the following expression. [2]

$$\mathbf{X} = \frac{X_m}{\sqrt{2}}\,[\cos(\phi) + j\, sin(\phi)]. \tag{2.3}$$

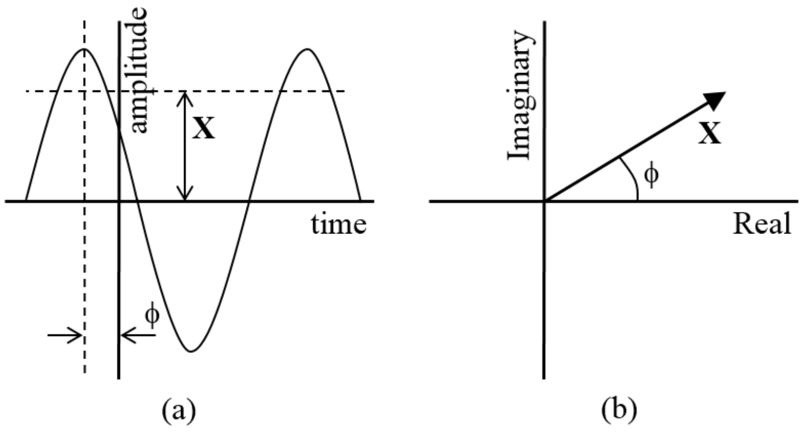


Figure 2.1: *(a) Sinusoidal signal and (b) its phasor representation.*

2.2 Phasor Measurement Units

AC voltages and currents in the power system can be modeled in the time domain using sinusoids as outlined in the above section, and thus as phasors in the form of Equation 2.3. The advantage of this conversion becomes evident when collecting multiple measurements of voltage and current signals throughout the power system in this form. Through utilization of a common time reference, phasor measurements with identical time tags become comparable regardless of geographical origin. This is solely by virtue of accurate synchronization of time tags, and thus these phasor measurements are often referred to as *synchrophasors*. When collected, the synchrophasor measurements provide instantaneous information about the power system. [2]

Phasor measurement units, commonly referred to as PMUs, are devices installed in substations throughout the electrical power grid to estimate the synchrophasors of current and voltage signals, as well as frequency and rate of change of frequency. Both hardware configuration and the software capabilities and applications of PMUs differ between manufacturers. Consequently, the following description of a generic PMU composition may not always be accurate, but describes the most important aspects of the device given its core functionality. See Figure 2.2 as reference for the following paragraphs. [2]

Instrument transformers convert the current and voltage signals to voltages in an appropriate range (commonly around 10 volts) for the analog-to-digital converter. The incoming analog signal is in practice corrupted by various signals. These include harmonics, out-of-band signals and high frequency disturbances. Anti-aliasing filters are utilized to extract a single frequency component from the incoming signal, efficiently limiting corrupting frequencies from affecting the final estimation. [2]

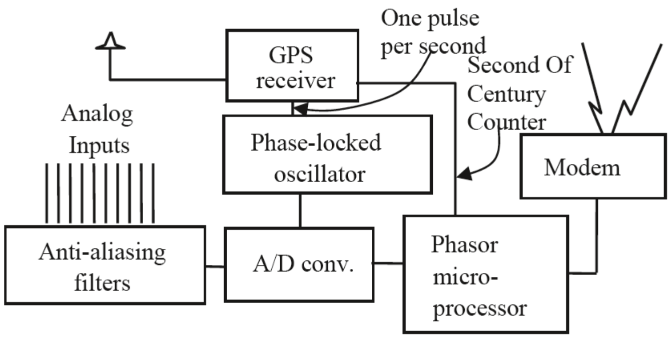


Figure 2.2: *Composition of a generic PMU. From Phadke and Thorp [2].*

Highly accurate time stamps of each signal sample are needed for the collective of PMUs to operate together. To achieve this, the global positioning system (GPS) is used as a time reference, giving a sufficient accuracy in the range of 1 µs. The GPS-receiver is connected to a phase-locked sampling clock, which provides the analog-to-digital signal converter. The microprocessor is then able to calculate the desired estimates and delivers the completed phasors and their accompanying time stamps to the communication transmitter. Synchrophasor measurements from PMUs located at different geographical locations are aggregated in a Phasor Data Concentrator (PDC) for further use in applications concerning monitoring, control and protection of the system. [2] [6]

The configuration of PMUs and PDCs differ according to the desired application, and not very many working cases exist today. The principle idea is to install PMUs in substations throughout the greater power grid, providing information about big portions of the system. Figure 2.3 highlights installation locations of PMUs and PDCs in the North American power grid. The amount of installed devices has multiplied several times over the last decade, greatly improving the information collection coverage in the region. A similar geographical overview of installed and planned PMUs in Norway is given in Figure 2.4.

Standards regarding data file structure for PMUs were established by IEEE in 1991, and later updated in 2005 [8], allowing for units of multiple manufacturing origins to collaborate.

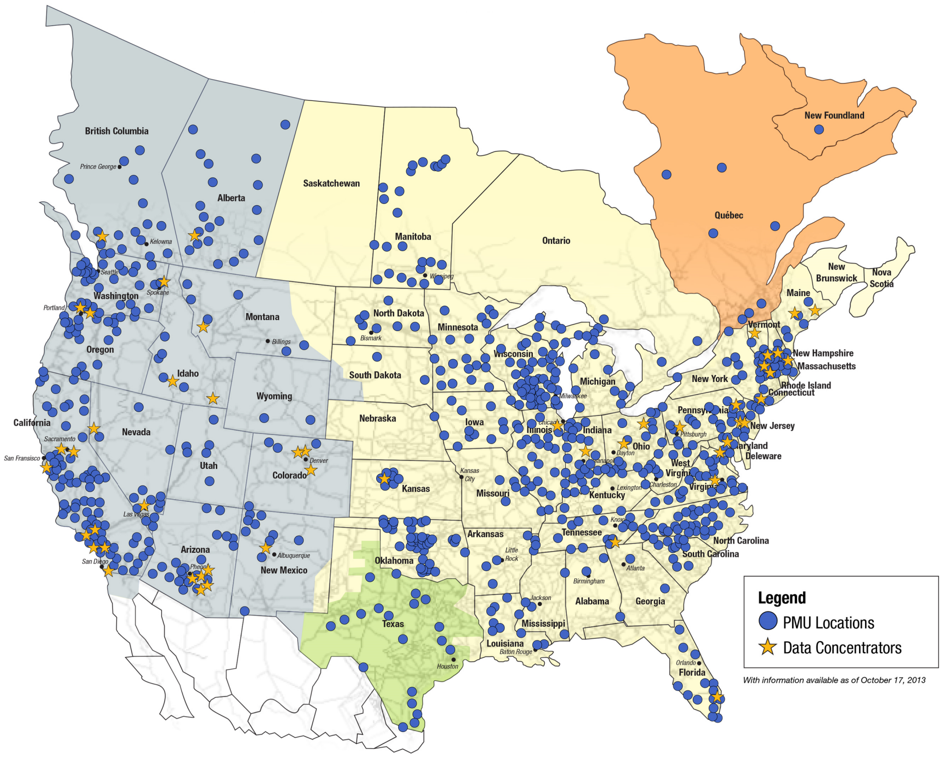


Figure 2.3: *Geographical locations of networked PMUs and PDCs in North America as of October 2013. From the North American SynchroPhasor Initiative [7].*

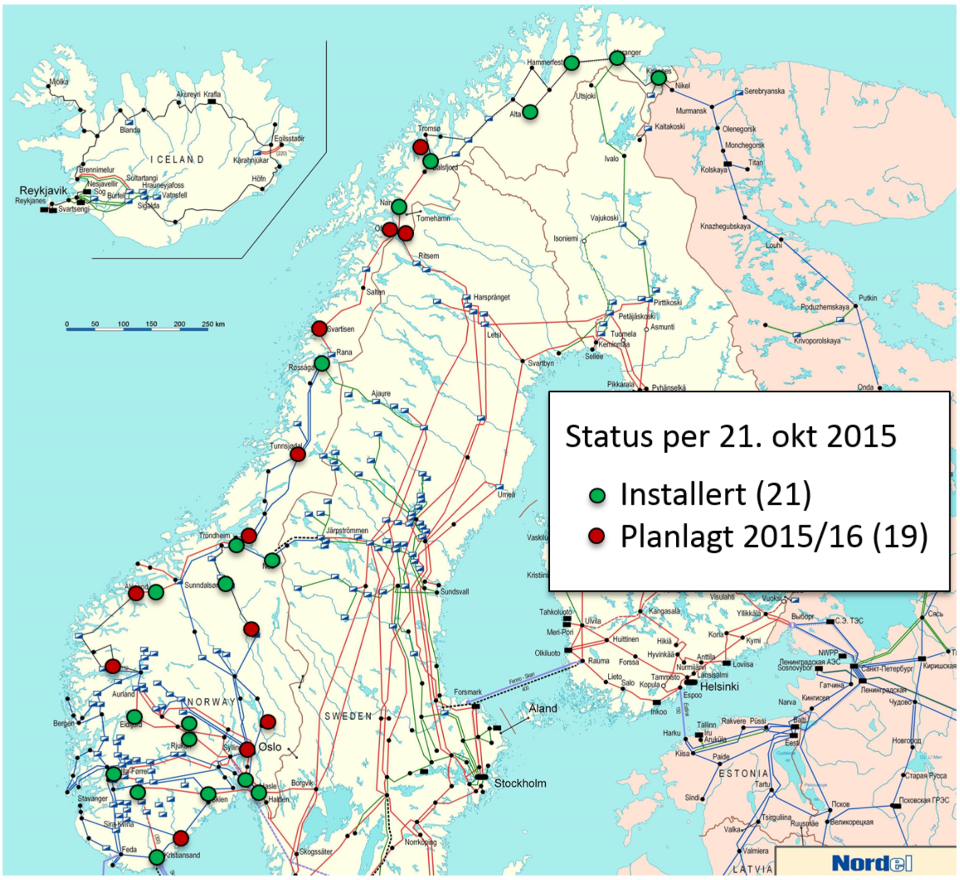


Figure 2.4: *Geographical locations of networked (green) and planned (red) PMUs in Norway as of October 2015. Courtesy of Statnett.*

3 | State Estimation

This chapter will present the theoretical principles behind state estimation for power systems. The use of measurements in the weighted least square method will be outlined, followed by a discussion of the classical state estimator. The chapter will then move on to the subject of utilizing PMU measurements in state estimation, leading up to Chapter 4 and the presentation of hybrid linear state estimation as a way of integrating PMU technology into today's state estimation systems. The construction and testing of such an estimator will be presented in the Chapters 5 and 6, respectively.

3.1 The Principles of State Estimation

The operating conditions of a transmission system can at any time be determined when the voltage phasor of every bus in the system is known along with the complete network model. This is often referred to as the static system state and is assigned to one of three possible states: *normal, emergency* and *restorative* [5]. These states will not be discussed in detail in this report, as it is sufficient to know that they indicate the operational safety of the system as a whole.

The process of identifying the current state of a system, given measurement data collected throughout the geographical extent of the power grid, is called *state estimation.* To avoid vulnerability to measurement errors and the loss or corruption of data, this process is completed with a redundancy of measurements. Throughout this thesis, it is assumed that the investigated system is *observable*, meaning that a sufficient amount of measurements are available for the estimator to arrive at a unique estimate of the system state. The process of determining whether this requirement is satisfied will not be discussed in this work, but an extensive explanation is provided in [5].

It is also worth noting that the methods introduced in this chapter represent one of several existing formulations of the state estimation problem. The underlying principles, however, remain the same throughout the various approaches.

3.2 Mathematical Basis

In this section, the mathematical basis used to conduct state estimation will be presented. Both a non-linear and a linear approach will be introduced.

3.2.1 Non-Linear Weighted Least Square

Consider a set of measurements given by the vector $\boldsymbol{z}^T = [z_1, z_2, ..., z_m]$ and the non-linear functions $\boldsymbol{h}^T = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), ..., h_m(\boldsymbol{x})]$ relating each measurement to the state variables $\boldsymbol{x}^T = [x_1, x_2, ..., x_n]$. Additionally, each measurement contains an error ε, such that $\boldsymbol{\varepsilon}^T = [\varepsilon_1, \varepsilon_2, ..., \varepsilon_m]$.

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} h_1(x_1, x_2, ..., x_n) \\ h_2(x_1, x_2, ..., x_n) \\ \vdots \\ h_m(x_1, x_2, ..., x_n) \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix} = \mathbf{h}(\mathbf{x}) + \boldsymbol{\varepsilon} \tag{3.1}$$

The measurement errors are assumed mutually independent and spread within a Gaussian probability distribution with a mean of zero. To be able to weigh different measurements according to their expected errors the matrix $\mathbf{W}$ is introduced as the co-variance matrix for the error variance of each measurement. Because of the independence of the individual errors, this matrix is purely diagonal, as illustrated in Equation 3.2. [5]

$$\mathbf{W} = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \sigma_m^2 \end{bmatrix} \tag{3.2}$$

Equation 3.1 can be turned with respects to the error matrix, giving an expression for the measurement errors.

$$\boldsymbol{\varepsilon} = \mathbf{z} - \mathbf{h}(\mathbf{x}) \tag{3.3}$$

The objective function J($\mathbf{x}$) is defined as the sum of every error squared, each divided by their individual variance.

$$J(\mathbf{x}) = \frac{\varepsilon_1^2}{\sigma_1^2} + \cdots + \frac{\varepsilon_m^2}{\sigma_m^2} = \boldsymbol{\varepsilon}^{\boldsymbol{T}} \cdot \mathbf{W}^{-1} \cdot \boldsymbol{\varepsilon} \tag{3.4}$$

Equation 3.3 can be substituted into Equation 3.4 to get

$$J(\mathbf{x}) = \frac{1}{2} \cdot [\mathbf{z} - \mathbf{h}(\mathbf{x})]^T \cdot \mathbf{W}^{-1} \cdot [\mathbf{z} - \mathbf{h}(\mathbf{x})]. \tag{3.5}$$

The goal of the estimation is to minimize J($\mathbf{x}$). To achieve this, the first-order optimality conditions must be satisfied. This can be expressed as

$$g(\mathbf{x}) = \frac{\delta J(\mathbf{x})}{\delta \mathbf{x}} = -\mathbf{H}^T(\mathbf{x}) \cdot \mathbf{W}^{-1} \cdot [\mathbf{z} - \mathbf{h}(\mathbf{x})] = 0, \tag{3.6}$$

where $\mathbf{H}$ is the measurement Jacobian coefficient matrix

$$\mathbf{H}(\mathbf{x}) = \frac{\delta \mathbf{h}(\mathbf{x})}{\delta \mathbf{x}}. \tag{3.7}$$

The gain matrix of this system is

$$\mathbf{G}(\mathbf{x}^k) = \frac{\delta g(\mathbf{x}^k)}{\delta \mathbf{x}} = \mathbf{H}^T(\mathbf{x}^k) \cdot \mathbf{W}^{-1} \cdot \mathbf{H}(\mathbf{x}^k), \tag{3.8}$$

where $\boldsymbol{x}^k$ is the solution vector at iteration k. $g(\boldsymbol{x})$ is non-linear and can be expanded into its Taylor series around the point $\boldsymbol{x} = \boldsymbol{x}^k$. By neglecting all higher order terms the resulting expression can be rearranged to arrive at Equation 3.9. From here, the iterative Gauss-Newton method can be used to find a solution vector satisfying a desired accuracy. [5]

$$\mathbf{x}^{k+1} = \mathbf{x}^k - [\mathbf{G}(\mathbf{x}^k)]^{-1} \cdot g(\mathbf{x}^k) \tag{3.9}$$

3.2.2 Linear Weighted Least Square

An estimation problem where all the measurements relate linearly to the state variables can be written on the form of Equation 3.10 below.

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} \frac{\delta h_1}{\delta x_1} & \frac{\delta h_1}{\delta x_2} & \cdots & \frac{\delta h_1}{\delta x_n} \\ \frac{\delta h_2}{\delta x_1} & & & \vdots \\ \vdots & & \ddots & \\ \frac{\delta h_m}{\delta x_1} & \cdots & & \frac{\delta h_m}{\delta x_n} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{bmatrix} = \mathbf{H} \cdot \mathbf{x} + \boldsymbol{\varepsilon} \tag{3.10}$$

In the above equation, $\mathbf{H}$ is the measurement Jacobian coefficient matrix. The vectors $\mathbf{z}$, $\mathbf{x}$ and $\boldsymbol{\varepsilon}$ are the sets of measurements, state variables and measurement errors, respectively. Similar to the approach described in Subsection 3.2.1, the diagonal co-variance matrix $\mathbf{W}$ is introduced, and the objective function becomes

$$J(\mathbf{x}) = [\mathbf{z} - \mathbf{H}\mathbf{x}]^T \cdot \mathbf{W}^{-1} \cdot [\mathbf{z} - \mathbf{H}\mathbf{x}]. \tag{3.11}$$

Following the first-order optimality condition gives Equation 3.12, which can be rearranged into Equation 3.13.

$$\left.\frac{\delta J(\mathbf{x})}{\delta \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\hat{\mathbf{x}} - \mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} = 0 \tag{3.12}$$

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{3.13}$$

As opposed to the method presented in Subsection 3.2.1, this estimation is linear and direct, and does not require iterations. [2]

3.3 Power System Application

3.3.1 Classical State Estimation

The mathematical principles outlined in the previous section can be applied to the electric power system. Traditional state estimators use measurements of various types. The most common measurements are line power flows, bus power injections, bus voltage magnitudes and line current flow magnitudes. Magnitude and angle of bus voltages are used as the system's state variables, giving the following state vector for a system with n buses. [5]

$$\mathbf{x}^T = [\theta_1, \theta_2, \ldots, \theta_n, V_1, V_2, \ldots, V_n] \tag{3.14}$$

One of the angles are selected as the reference bus and set to an arbitrary value, generally zero. Assuming all constant network parameters are known, the measurement function for each of the measurement types can be found with the help of a standard π equivalent transmission line model, as illustrated in Figure 3.1. V_i and V_j represent the voltage magnitudes of the buses i and j. The parameters $g_{ij} + jb_{ij}$ and $g_{s,ij} + jb_{s,ij}$ are the branch series and shunt admittances, respectively.

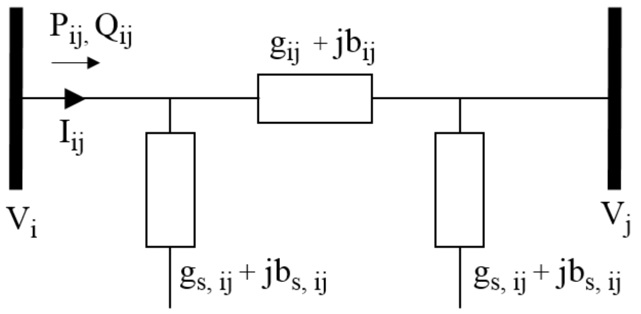


Figure 3.1: *π equivalent model of transmission line.*

The real and reactive line power flow measurements from bus i to bus j are given by the Equations 3.15 and 3.16 below. [5]

$$P_{ij} = V_i^2(g_{s,ij} + g_{ij}) - V_iV_j(g_{ij}\cos(\theta_i - \theta_j) + b_{ij}\sin(\theta_i - \theta_j)) \tag{3.15}$$

$$Q_{ij} = -V_i^2(b_{s,ij} + b_{ij}) - V_iV_j(g_{ij}\sin(\theta_i - \theta_j) - b_{ij}\cos(\theta_i - \theta_j)) \tag{3.16}$$

Given $G_{ij} + jB_{ij}$ as the ij-th element of the bus admittance matrix and $\aleph_i$ as the set of buses connected to bus i, the real and reactive bus power injections are

$$P_i = V_i \sum_{j \epsilon \aleph_i} V_j(G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)) \tag{3.17}$$

$$Q_i = V_i \sum_{j \epsilon \aleph_i} V_j(G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j)) \tag{3.18}$$

The line current flow magnitude measurement function is

$$I_{ij} = \frac{\sqrt{P_{ij}^2 + Q_{ij}^2}}{V_i} \tag{3.19}$$

This results in the measurement Jacobian coefficient matrix, **H**, given i Equation 3.20. Every row in the matrices **z**, **H** and $\boldsymbol{\varepsilon}$ represents the entirety of a single measurement type. Similarly, the vectors in the $\boldsymbol{x}$-matrix contain all bus voltage angles and magnitudes. [5]

$$\mathbf{z} = \begin{bmatrix} \mathbf{P_{flow}} \\ \mathbf{Q_{flow}} \\ \mathbf{P_{inj}} \\ \mathbf{Q_{inj}} \\ \mathbf{I_{mag}} \\ \mathbf{V_{mag}} \end{bmatrix} = \begin{bmatrix} \frac{\delta \mathbf{P_{flow}}}{\delta \boldsymbol{\theta}} & \frac{\delta \mathbf{P_{flow}}}{\delta \mathbf{V}} \\ \frac{\delta \mathbf{Q_{flow}}}{\delta \boldsymbol{\theta}} & \frac{\delta \mathbf{Q_{flow}}}{\delta \mathbf{V}} \\ \frac{\delta \mathbf{P_{inj}}}{\delta \boldsymbol{\theta}} & \frac{\delta \mathbf{P_{inj}}}{\delta \mathbf{V}} \\ \frac{\delta \mathbf{Q_{inj}}}{\delta \boldsymbol{\theta}} & \frac{\delta \mathbf{Q_{inj}}}{\delta \mathbf{V}} \\ \frac{\delta \mathbf{I_{mag}}}{\delta \boldsymbol{\theta}} & \frac{\delta \mathbf{I_{mag}}}{\delta \mathbf{V}} \\ \frac{\delta \mathbf{V_{mag}}}{\delta \boldsymbol{\theta}} & \frac{\delta \mathbf{V_{mag}}}{\delta \mathbf{V}} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\theta} \\ \mathbf{V} \end{bmatrix} + \begin{bmatrix} \varepsilon_{\boldsymbol{P_{flow}}} \\ \varepsilon_{\boldsymbol{Q_{flow}}} \\ \varepsilon_{\boldsymbol{P_{inj}}} \\ \varepsilon_{\boldsymbol{Q_{inj}}} \\ \varepsilon_{\boldsymbol{I_{mag}}} \\ \varepsilon_{\boldsymbol{V_{mag}}} \end{bmatrix} = \mathbf{H} \cdot \mathbf{x} + \boldsymbol{\varepsilon} \tag{3.20}$$

An exhaustive list of the complete expressions for each **H**-matrix segment can be found in Appendix A.

Given the non-linear relation between most of the measurements and the state variables, the estimator is forced to use the non-linear approach presented in Subsection 3.2.1. This type of state estimator, utilizing SCADA-collected measurements and applying the non-linear weighted least square method, is referred to as the *classical state estimator* (CSE). One of the main problems with this type of estimation is linked to the SCADA's data scan duration and the lack of time stamping. It normally takes the SCADA system between 5 and 10 seconds to request and receive measurements from the system's RTUs, in which time the system state might change. This means that an assumption central to classical state estimation is that the system remains static throughout the process of measurement collection.

This can have a big impact on the quality of the final estimate, depending on the current operating conditions of the system. As an example, dynamic behaviours like unwanted waveforms might be difficult to identify because of this. [2] [5]

3.3.2 State Estimation with Phasor Measurements

Imagine a system where PMU measurements alone could supply information enough to form complete observability. This would have several effects on the estimate of the system's state. Firstly, it would allow the estimator to apply the linear and non-iterative method outlined in Subsection 3.2.2. Both bus voltage and line current measurements conducted by PMUs are given in rectangular coordinates. Because of this, the state vector needs to be changed from polar to rectangular coordinates for the shift to the linear approach to be achieved. This results in a state vector on the form given in Equation 3.21, where n is the number of buses in the system. The subscripts R and I denote the real and imaginary rectangular components of the bus voltages, respectively.

$$\mathbf{x} = \begin{bmatrix} \mathbf{V_R} \\ \mathbf{V_I} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} V_{R1} \\ V_{R2} \\ \vdots \\ V_{Rn} \end{bmatrix} \\ \begin{bmatrix} V_{I1} \\ V_{I2} \\ \vdots \\ V_{In} \end{bmatrix} \end{bmatrix} \tag{3.21}$$

The second benefit of using only PMU measurements is rooted in their linked time-tags. Since all measurements will have a very accurate time stamp, the static assumption could be removed and the estimate would be a snapshot of the dynamic system at the point in time where the synchrophasor measurements were collected. A slight time skew is however unavoidable because of the communication delay. Another benefit of the PMU-monitored system worth mentioning is the PMU's inherent low measurement error, which in general is smaller than that of the generic RTU.

There are, however, a few issues that merits a mention. Conducting a state estimation using purely PMU measurements will require a substantial amount of the system's buses to have a PMU installed. Redundancy is also a requirement to be able to eliminate bad data caused by poor measurements and corruption during communication. As per today, the aforementioned proposed system is unlikely due to the costs of PMUs, but might become a reality in the years ahead, as PMU technology gains a bigger foothold in both academic and commercial aspects. [2] [9]

The complete measurement model for this system is given in Equation 3.22 below. [9]

$$\mathbf{z} = \begin{bmatrix} \mathbf{V_R} \\ \mathbf{V_I} \\ \mathbf{I_R} \\ \mathbf{I_I} \end{bmatrix} = \begin{bmatrix} \frac{\delta \mathbf{V_R}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{V_R}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{V_I}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{V_I}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{I_R}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{I_R}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{I_I}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{I_I}}{\delta \mathbf{V_I}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{V_R} \\ \mathbf{V_I} \end{bmatrix} + \begin{bmatrix} \varepsilon_{V_R} \\ \varepsilon_{V_I} \\ \varepsilon_{I_R} \\ \varepsilon_{I_I} \end{bmatrix} = \mathbf{H} \cdot \mathbf{x} + \varepsilon \qquad (3.22)$$

The expressions $\frac{\delta \mathbf{V_R}}{\delta \mathbf{V_I}}$ and $\frac{\delta \mathbf{V_I}}{\delta \mathbf{V_R}}$ are both vectors of zeros, while both $\frac{\delta \mathbf{V_R}}{\delta \mathbf{V_R}}$ and $\frac{\delta \mathbf{V_I}}{\delta \mathbf{V_I}}$ are sets of vectors, each representing a single bus voltage measurement, with 1 in columns corresponding to that specific measurement's state variable, and zeros as the remaining values. [9]

Consider a nominal π equivalent model of a transmission line illustrated in Figure 3.2 below.

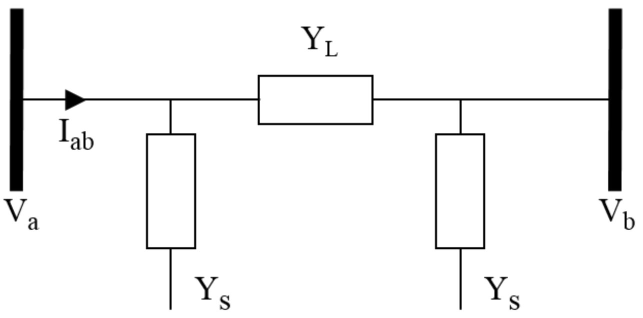


Figure 3.2: *π equivalent model of transmission line.*

Each of the variables depicted in the π model can be decomposed into rectangular coordinates as shown in the following equations. Y_L represents the total series line admittance.

$$Y_L = \frac{1}{Z_L} = \frac{1}{R_L + jX_L} \qquad (3.23)$$

Y_S is half of the total shunt admittance of the transmission line. Note that the shunt conductance, G_S, is often neglected in practical applications.

$$Y_S = G_S + jB_S \qquad (3.24)$$

The bus voltages and line currents can be expressed in their real and imaginary components.

$$V_a = V_{R,a} + jV_{I,a} \tag{3.25}$$

$$I_{ab} = I_{R,ab} + jI_{I,ab} \tag{3.26}$$

The PMU measurements come in four variants, namely the real and imaginary components of bus voltages and line currents. The measurement functions, expressed by the constant network parameters and the state variables, are presented in Equations 3.27 through 3.30. [10]

$$V_{R,a} = V_{R,a} \tag{3.27}$$

$$V_{I,a} = V_{I,a} \tag{3.28}$$

$$I_{R,ab} = V_{R,a} \cdot (G_L + G_S) - V_{I,a} \cdot (B_L + B_S) - V_{R,b} \cdot G_L + V_{I,b} \cdot B_L \tag{3.29}$$

$$I_{I,ab} = V_{R,a} \cdot (B_L + B_S) + V_{I,a} \cdot (G_L + G_S) - V_{R,b} \cdot B_L - V_{I,b} \cdot G_L \tag{3.30}$$

This gives the following general measurement Jacobian coefficient matrix, **H**, where the measuring PMU is located in the substation of bus a with a line current measurement conducted on the branch leading from bus a to bus b. [10]

$$\mathbf{H} = \begin{bmatrix} \frac{\delta V_{R,a}}{\delta V_{R,a}} & \frac{\delta V_{R,a}}{\delta V_{R,b}} & \frac{\delta V_{R,a}}{\delta V_{I,a}} & \frac{\delta V_{R,a}}{\delta V_{I,b}} \\ \frac{\delta V_{I,a}}{\delta V_{R,a}} & \frac{\delta V_{I,a}}{\delta V_{R,b}} & \frac{\delta V_{I,a}}{\delta V_{I,a}} & \frac{\delta V_{I,a}}{\delta V_{I,b}} \\ \frac{\delta I_{R,ab}}{\delta V_{R,a}} & \frac{\delta I_{R,ab}}{\delta V_{R,b}} & \frac{\delta I_{R,ab}}{\delta V_{I,a}} & \frac{\delta I_{R,ab}}{\delta V_{I,b}} \\ \frac{\delta I_{I,ab}}{\delta V_{R,a}} & \frac{\delta I_{I,ab}}{\delta V_{R,b}} & \frac{\delta I_{I,ab}}{\delta V_{I,a}} & \frac{\delta I_{I,ab}}{\delta V_{I,b}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ G_L + G_S & -G_L & -(B_L + B_S) & B_L \\ B_L + B_S & -B_L & G_L + G_S & -G_L \end{bmatrix} \tag{3.31}$$

Given the above information, the linear weighted least square method discussed in Subsection 3.2.2 can now be applied to the measurement model from Equation 3.22 to achieve the non-iterative solution given below [2].

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{W}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{W}^{-1} \mathbf{z} = \mathbf{M} \cdot \mathbf{z} \tag{3.32}$$

Note that the matrix **M** is constant as long as the network topology does not change. This means that it can be pre-calculated offline and stored between estimations, thus increasing speed of the process.

3.4 Bad Data Detection and Elimination

The presence of bad measurements, measurement values that are not coinciding with the actual parameter, is an inevitable occurrence in big measurement configurations. These can be caused by an inaccurate measurement from the measurement unit itself or because of a corrupted communication packet. The former of these normally result in relatively small errors, whereas the latter can cause huge discrepancies. In either case these measurements can to a large degree affect the outcome of an estimation. Some measurements should therefore be eliminated before a final estimation solution is reached. [2]

This process is referred to as *bad data detection and elimination* and can be done in a number of different ways. The method highlighted here is completed in two main steps. After an initial estimate has been completed, the measurements are evaluated using a chi-squared test. Let $\tilde{\mathbf{z}}$ be a vector containing the measurement residuals, the differences between the measurements and their respective estimated values so that

$$\tilde{\mathbf{z}} = \mathbf{z} - \hat{\mathbf{z}} = \mathbf{z} - \mathbf{H}\hat{\mathbf{x}}, \tag{3.33}$$

where $\hat{\mathbf{x}}$ is the resulting state vector after the initial estimation. Next, $\mathbf{R}$ is defined as the co-variance of $\tilde{\mathbf{z}}$.

$$\mathbf{R} = Cov(\tilde{\mathbf{z}}) = \mathbf{H}(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H})^{-1}\mathbf{H}^T \tag{3.34}$$

Finally, the vector residuals are normalized by their co-variance matrix, resulting in c, a χ^2-distributed variable with degrees of freedom corresponding to the amount of measurements.

$$c = \tilde{\mathbf{z}}^T\mathbf{R}^{-1}\tilde{\mathbf{z}} \tag{3.35}$$

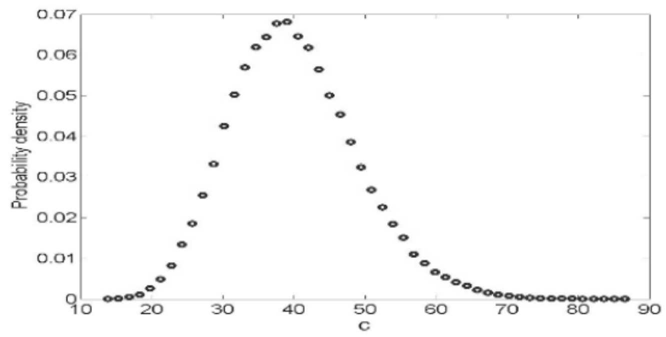


Figure 3.3: *Chi-squared distribution of c with 40 degrees of freedom. From Phadke and Thorp [2].*

The variable c is a representation of the total measurement residual of the estimation, and an off-center value (e.g. above 60 or below 20 in the case of Figure

3.3) indicates discrepancies in the measurements. For every specific case, an upper and a lower bond is chosen as an acceptance limit for the measurement set. If c is found to be outside the acceptable area the process moves on to its second step: bad data rejection. Here, the measurement with the highest normalized residual is found and removed from the data set before a new state estimation is conducted with the updated measurement set. The result is again evaluated in the same way, and the process is repeated until c is within the allowed band. [2]

Note that in cases with high amounts of measurements it can be pertinent to remove several measurements for every iteration to reduce the total amount of state estimations needed. Also, measurements with grossly big residuals are often automatically removed after the first state estimation is conducted. [2]

This method is consistently used in classical static state estimation, which can easily be done because of the sheer amount of measurements providing a high degree of redundancy. This means that eliminating a few of the measurements will not affect the complete observability of the system. With PMU measurements, however, measurement redundancy is rarely present. This implies that bad data detection and elimination on measurement sets consisting solely of synchrophasors is in many cases impossible without reducing observability through removal of measurements. It can, however, be applied in conjunction with static state estimation results as described in the following chapter. Being able to eliminate any corrupting measurements despite lacking complete observability from PMUs alone is an important advantage of the Hybrid Linear State Estimation model, which will be discussed in the following chapter. [2]

4 | Hybrid Linear State Estimation

4.1 Background and Principle

PMU coverage is rarely extensive enough to single-handedly provide complete observability, let alone redundancy enough to give a satisfactory estimate of the system state. Despite this, introducing synchrophasor measurements into state estimation can potentially bring great benefits in terms of both accuracy and confidence. Several methods of utilizing PMU measurements in state estimation has been suggested. These include, but are not limited to, introducing PMU measurements directly into classical state estimation by transforming them to polar coordinates, and use of pseudo measurements of power flows in tandem with PMU measurements to achieve complete observability. [2]

The approach discussed and investigated in this thesis was proposed by Nuqui and Phadke in 2007 [9]. The proposed model attempts to tackle the problem of incomplete PMU coverage through a two-pass verification method. The first pass utilizes classical static state estimation through WLS using the system's SCADA-collected measurements, as discussed in Subsection 3.3.1. The second pass is a linear state estimation using PMU measurements as well as the state vector resulting from the first pass. This can be viewed two ways; the static state estimation is used to counter the lack of complete PMU obervability, or the PMU measurements are used to improve the original static state estimation. Either way one chooses to look at it, this model aims to improve on the final estimation of the system state. Figure 4.1 presents a visual illustration of the hybrid linear estimation process. [9]

Note that the state vector must prior to the second pass be transformed from polar to Cartesian coordinates for the linearity requirement to be satisfied.

A common angle reference between the state vector values and the PMU measurements must also be established. Since most control centers today already employ a classical state estimator, this method would in practice be performed as a post-processing operation of the initial state estimate. For this reason, the first pass of the method is not discussed in-depth in this chapter. Since the two passes are completed separately, the method is completely non-invasive and can therefore be implemented alongside existing state estimators without compromising these. [2] [9]

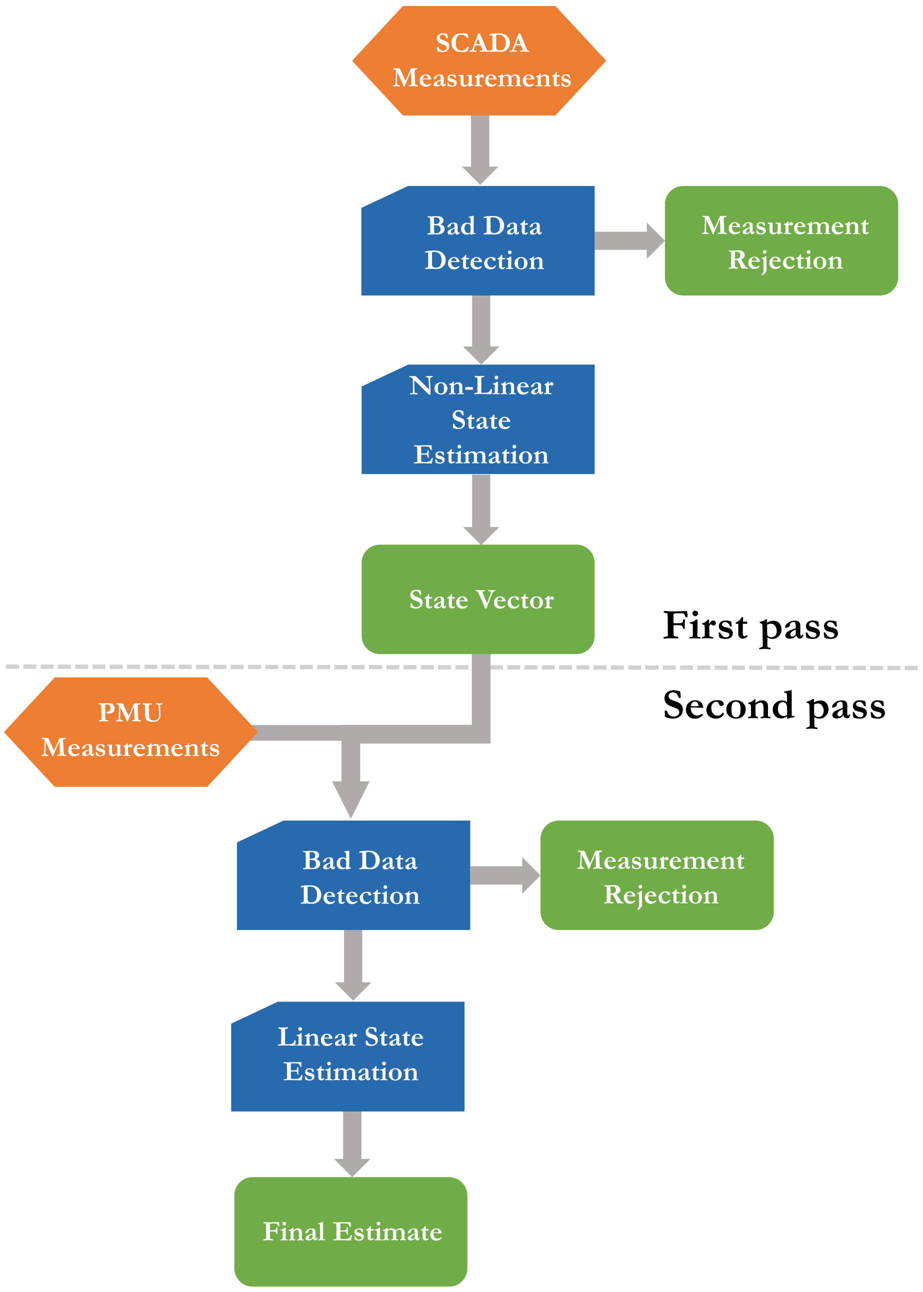


Figure 4.1: *The two-pass hybrid linear state estimator model structure.*

4.2 Model Formulation

The second pass linear estimation is performed mathematically in a way similar to that discussed in Section 3.3.2, where the state vector from the first pass, from here referred to as the *CSE state vector*, is treated as a set of measurements. The complete measurement model presented in Equation 3.22 is expanded to facilitate the incorporation of the classically estimated state vector as shown in Equation 4.1 below. The subscripts *CSE* and *PMU* refer to values obtained from the classical state estimator's state vector and PMU measurements, respectively.

$$\mathbf{z} = \begin{bmatrix} \begin{bmatrix} \mathbf{V}_R \\ \mathbf{V}_I \end{bmatrix}_{CSE} \\ \begin{bmatrix} \mathbf{V}_R \\ \mathbf{V}_I \end{bmatrix}_{PMU} \\ \begin{bmatrix} \mathbf{I}_R \\ \mathbf{I}_I \end{bmatrix}_{PMU} \end{bmatrix} = \begin{bmatrix} \frac{\delta \mathbf{V_{R,CSE}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{V_{R,CSE}}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{V_{I,CSE}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{V_{I,CSE}}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{V_{R,PMU}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{V_{R,PMU}}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{V_{I,PMU}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{V_{I,PMU}}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{I_{R,PMU}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{I_{R,PMU}}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{I_{I,PMU}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{I_{I,PMU}}}{\delta \mathbf{V_I}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{V_R} \\ \mathbf{V_I} \end{bmatrix} + \begin{bmatrix} \varepsilon_{\boldsymbol{V_{R,CSE}}} \\ \varepsilon_{\boldsymbol{V_{I,CSE}}} \\ \varepsilon_{\boldsymbol{V_{R,PMU}}} \\ \varepsilon_{\boldsymbol{V_{I,PMU}}} \\ \varepsilon_{\boldsymbol{I_{R,PMU}}} \\ \varepsilon_{\boldsymbol{I_{I,PMU}}} \end{bmatrix} = \mathbf{H} \cdot \mathbf{x} + \boldsymbol{\varepsilon} \tag{4.1}$$

The measurement Jacobian coefficient matrix, $\mathbf{H}$, can be rewritten in the following manner

$$\mathbf{H} = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{I} \\ \mathbf{II} & 0 \\ 0 & \mathbf{II} \\ \frac{\delta \mathbf{I_{R,PMU}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{I_{R,PMU}}}{\delta \mathbf{V_I}} \\ \frac{\delta \mathbf{I_{I,PMU}}}{\delta \mathbf{V_R}} & \frac{\delta \mathbf{I_{I,PMU}}}{\delta \mathbf{V_I}} \end{bmatrix}, \tag{4.2}$$

where $\mathbf{I}$ is an identity matrix and $\mathbf{II}$ is a set of vectors, each representing a single bus voltage measurement, with 1 in columns corresponding to that specific measurement's state variable, and zeros as the remaining values. Additionally, the co-variance matrix, $\mathbf{W}$, has been expanded to include the variances of the CSE state vector values as illustrated in Equation 4.3.

$$\mathbf{W} = \begin{bmatrix} \boldsymbol{\sigma}^2_{V_{R,CSE}} & & & \cdots & & 0 \\ & \boldsymbol{\sigma}^2_{V_{I,CSE}} & & & & \\ & & \boldsymbol{\sigma}^2_{V_{R,PMU}} & & & \vdots \\ \vdots & & & \boldsymbol{\sigma}^2_{V_{I,PMU}} & & \\ & & & & \boldsymbol{\sigma}^2_{I_{R,PMU}} & \\ 0 & & \cdots & & & \boldsymbol{\sigma}^2_{I_{I,PMU}} \end{bmatrix} \tag{4.3}$$

In $\mathbf{W}$, each element, $\boldsymbol{\sigma^2_{type}}$, is a diagonal matrix containing the individual measurement variances within the distinct measurement type. [9]

The weighted least square solution of the model is non-iterative and on the same form as in Subsection 3.2.2:

$$\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}^{-1}\mathbf{z} \tag{4.4}$$

4.3 A Brief Discussion of the Model

No approach to processing big amounts of data is perfect. This section aims to briefly outline the most important benefits and flaws of the hybrid linear state estimator model, on a strictly conceptual basis.

4.3.1 Benefits

The most obvious benefit provided by the above presented model is the potential to increase estimate accuracy and thus increase the confidence of the result. This will allow for better security margins and more sound economical decision making for system operators.

Among present state estimators, an update to the estimate is normally calculated with time intervals in the range of 10 seconds, while PMU systems often report measurements at 50 Hz. This means that the hybrid estimator can deliver an updated estimate with a much higher time-frequency than classical systems. This, combined with the more meticulous time stamping of PMU measurements, allows for more reliable detection of unwanted waveforms in the system, for example in the case of faults in the system.

The use of the first pass state vector as measurements increases the redundancy of measurements and allows for bad data detection and elimination among PMU measurements, despite having a low number of units deployed in the system.

Lastly, it should again be pointed out that the hybrid estimator can be used as post processing of existing classical state estimator systems, and can be implemented in a non-invasive manner completely separate from the SCADA-system and EMS.

4.3.2 Problems

A requirement for the steady operation of the hybrid estimator is the successful convergence of the classical state estimator. Although this is rarely a problem, it would halt the hybrid model until a CSE state vector is provided.

Another prerequisite for the model is a common angle reference with the CSE. The obvious solution to this is monitoring the reference bus of the classical estimator with a PMU. In the case of unit breakdown or telemetry faults at this substation, however, there should be a protocol for selecting a new common reference in its place.

Finding a good value for the variance of the elements in the first pass state vector is difficult. One approach to this is to let the relevant components of the co-variance matrix, $\mathbf{W}$, be a representation of the confidence and expected variance of the classic state estimator results. Unfortunately, this is only valid for the first iteration of the hybrid estimator after it receives a state vector from the CSE. As the hybrid estimator will run several times between each CSE iteration, it is clear that the older the CSE estimate is, the less reliable it becomes. One way of dealing with this problem is to gradually change the weight of the measurements depending on their age. This approach can help in compensating for the time skew, but is still not a perfect solution.

Last but not least it should be mentioned that, despite introducing PMU measurements into the estimation process, the static assumption can not be removed, due to the continued extensive use of SCADA-collected measurements.

5 | Model Construction

5.1 Purpose and Objective

As part of this work, a Hybrid Linear State Estimator (HLSE) has been constructed and tested within different simulation environments. This chapter describes the approach taken in the construction process. The aim of the model was to conduct a proof-of-concept study to investigate the method's feasibility as a way of utilizing the increasing availability of synchrophasor measurements to improve classical state estimation. The functionality objective of the model was to use the methodology introduced in the previous chapters to arrive at a system state estimate, given the following simulated input data:

Network data
Topology layout and parameter data for each branch in the system.

PMU measurements
Synchrophasor measurements of bus voltages and line currents in rectangular coordinates, along with their physical locations in the network.

SCADA measurements
Measurements of line power flows, bus power injections, bus voltage magnitudes and line current flow magnitudes, conducted by RTUs and collected by a SCADA-system.

5.2 Model Structure

The resulting hybrid linear state estimator application was constructed as a set of Matlab scripts. Matlab version 8.4.0 (R2014b) [11] was used in both the construction of the application and the subsequent numeric simulation analysis.

The estimator application is run from a Main script, acting as a hub for the second pass of the estimator model. Several functions are connected to the Main script, each performing different tasks related to the state estimation process. The Main script is in turn called by a Run_simulation script which controls the testing environment of the numerical simulations. Figure 5.1 illustrates the complete application structure and the information exchanged between the Main script and each individual function. Each of the model's components and their functionality will be discussed in the following subsections. Most of the produced scripts are supplied in Appendix B.

5.2.1 Use of Downloaded Scripts

In this work, the focus point of the construction and testing processes revolved around the second pass of the hybrid estimator. In the later testing of the model, supplying a realistic state vector to the second pass of the hybrid estimator was important. The first pass of the HLSE can be conducted with a standard classical state estimator. This is a tried and tested algorithm of which scripts are readily available on file exchange communities. A power system state estimator utilizing the non-linear weighted least square method was downloaded from the Matlab Central File Exchange. This script was used as the first pass of the estimator. The name of the downloaded document is *Power System State Estimation using WLS* and can be found by following this hyperlink [12].

This script was modified to work as a Matlab function, enabling other scripts to call upon it to receive a classically estimated state vector from a supplied set of measurements. The network model used for testing the final application was added to the downloaded script. Additionally, a measurement error generator was implemented in the script. The latter is further discussed later in this chapter.

5.2.2 Run_simulation Script

The numerical simulations applied to the hybrid estimator application, as described in Chapter 6, involve the execution of several state estimation runs. These extended tests are governed by the Run_simulation script. It is not technically a part of the estimator application itself, but is rather what prompts every new estimation to start throughout the testing process. The main task of this component is to manage the simulation by supplying the Main script with the input data relevant for the desired test and by collecting the results from the estimator so that they can later be accessed. For the purpose of the numerical simulations, it also runs the classical state estimator, enabling it to supply the second pass with a realistic classically estimated state vector, and error generation scripts to add measurement errors to all measurements.

The data provided by Run_simulation to the Main script is given in two matrices: *measurements* and *branches*. The *measurements* matrix is a list of measurements and values that are viewed by the linear state estimator as measurements. This means that it is comprised of PMU measurements of bus voltages and line currents, together with the classically estimated state vector, which in the second pass is considered as measurements. Each row of the matrix represents a single measurement, whilst the range of information provided with each measurement is defined by the matrix's columns. An overview of information contained within the *measurements* matrix is given in Table 5.1.

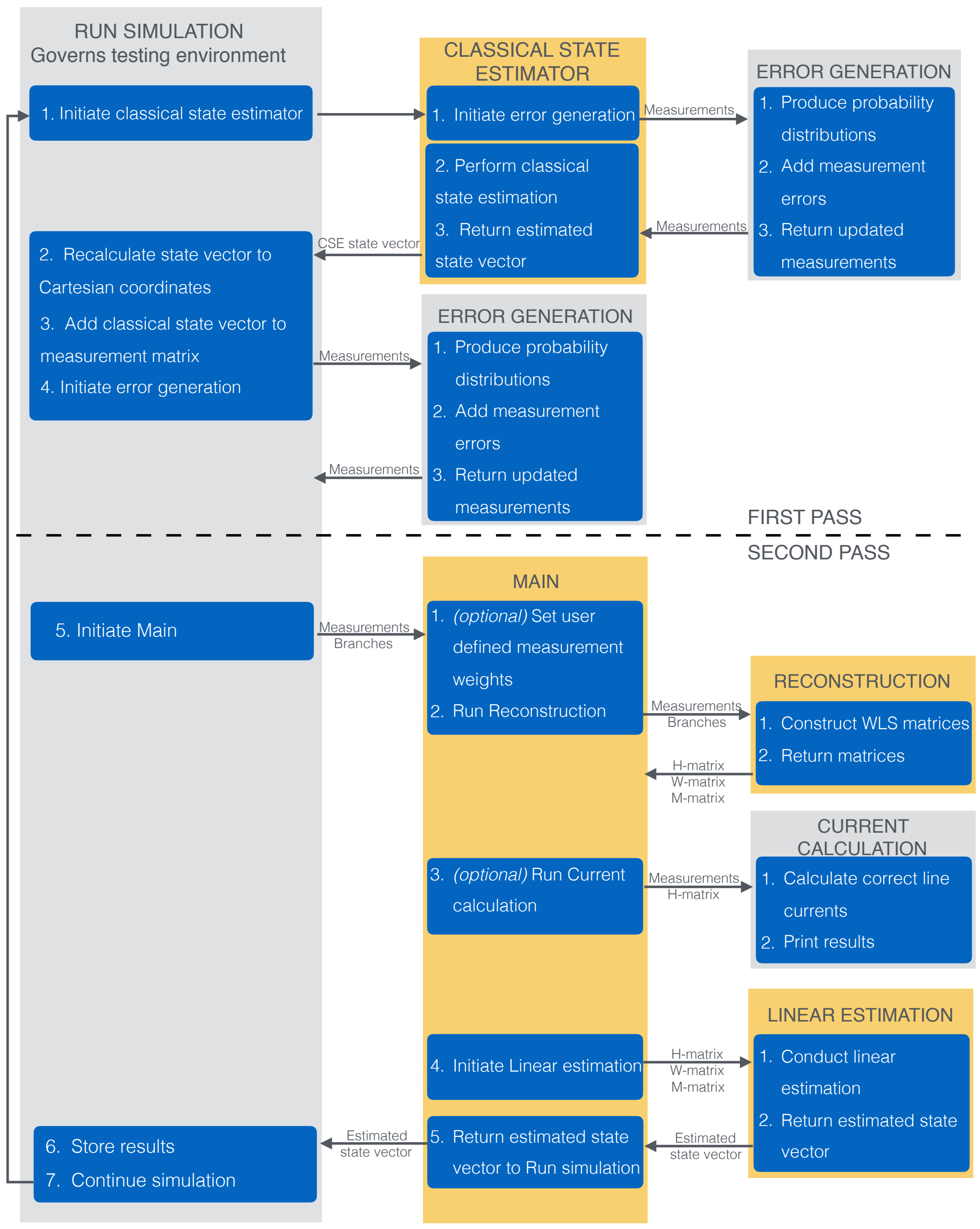

RUN SIMULATION
Governs testing environment
1. Initiate classical state estimator
2. Recalculate state vector to Cartesian coordinates
3. Add classical state vector to measurement matrix
4. Initiate error generation
5. Initiate Main
6. Store results
7. Continue simulation
CLASSICAL STATE ESTIMATOR
1. Initiate error generation
2. Perform classical state estimation
3. Return estimated state vector
ERROR GENERATION
1. Produce probability distributions
2. Add measurement errors
3. Return updated measurements
Measurements
CSE state vector
FIRST PASS
SECOND PASS
MAIN
1. (optional) Set user defined measurement weights
2. Run Reconstruction
3. (optional) Run Current calculation
4. Initiate Linear estimation
5. Return estimated state vector to Run simulation
Branches
RECONSTRUCTION
1. Construct WLS matrices
2. Return matrices
H-matrix
W-matrix
M-matrix
CURRENT CALCULATION
1. Calculate correct line currents
2. Print results
LINEAR ESTIMATION
1. Conduct linear estimation
2. Return estimated state vector
Estimated state vector

Figure 5.1: *A graphical representation of the Hybrid Linear State Estimator application structure. The yellow blocks represent the scripts that constitute the two-pass hybrid estimator algorithm. The grey blocks are scripts needed to conduct the numerical simulations.*

Table 5.1: *Information contained within the measurements matrix.*

Unit number	The arbitrary number given to each specific value
Unit type	The type specification of each value Type 1: PMU bus voltage measurements Type 2: PMU line current measurements Type 3: Classic state estimation bus voltage values
Bus from	Type 1 and 3: the bus at which the value corresponds to Type 2: the substation in which the PMU is located
Bus to	Type 1 and 3: not used Type 2: the bus at the opposite end of the measured branch
Value	The measured or estimated value as a complex number given in rectangular coordinates and per unit
Confidence	The variance of the value's expected measurement or estimation error in per unit

The *branches* matrix provides the estimator application with information regarding the network topology through a complete list of the branches in the system. The build-up of the matrix is similar to the *measurements* matrix described above, with individual branches represented in each row and accompanying information given in the different columns. An overview of information contained within the *branches* matrix is given in Table 5.2. This matrix provides the state estimator with enough information to form a complete model of the network topology.

Table 5.2: *Information contained within the branches matrix.*

Branch number	The arbitrary number given to each branch
Bus from Bus to	The buses connected by the branch It is insignificant which is selected as *from* and *to*
Series impedance	The series impedance of the branch as a complex number given in per unit
Shunt admittance	The total shunt admittance of the branch as a complex number given in per unit

It should be mentioned that the Run_simulation script changes slightly between the Monte Carlo steady-state based tests conducted to investigate the effects of PMU coverage and measurement weighting, and the time based dynamic response simulations. The purpose of the script, however, remains the same throughout the numerical simulations.

5.2.3 Main Script

The second pass of the hybrid linear state estimator application is run from the Main script, which is acting as a hub for the estimation process. Its task is to handle the exchange of information between the estimator's function scripts and deliver the final estimate back to the Run_simulation script. After receiving the *measurements* and *branches* matrices from the Run_simulation script, it passes them both on to the Reconstruction script, which returns the set of matrices required to complete the linear state estimation. The Main script then hands these matrices over to the Linear_estimation script, which completes the final linear state estimate. After receiving the resulting state vector, the Main script returns this to the Run_simulation script, thus completing its run.

In addition to this, the script has two optional operations, both essential to the testing process. First of these is the ability to manually set the weights of the three types of measurements, influencing the degree of impact the individual types have on the final estimate. The second optional action is to initiate the Current_calculation script, which returns the actual line currents of the system.

5.2.4 Reconstruction Script

This script constructs the matrices required to complete the linear state estimation based on the *measurements* and *branches* matrices. The constructed matrices are called **M**, **W** and **H**, and correspond to **z**, **W** and **H** from Equation 4.4. After these are constructed, they are returned to the Main script.

5.2.5 Linear_estimation Script

Supplied with the matrices outlined in the above subsection, this short script conducts the non-iterative linear state estimation using Equation 4.4. After this calculation, the estimated system state is returned to the Main script as the state vector.

5.2.6 Error_generator Scripts

The purpose of this script is to add measurement errors to an existing set of measurements, which are used in the numerical simulation to mimic real-world measurement errors. It does this by creating probability distributions based on the measurement types, within which it calculates an error for each individual measurement. This error is added to the measurement, and then returned to the Main script.

The probability distributions are all Gaussian with zero mean, and have standard deviations given by the user. The script also contains an optional operation which allows the user to print the measurement errors in comparison to the original set of measurement.

Appendix B contains the Error_generator script used to handle PMU measurements. The measurements given to the classical state estimator are also given errors according to the same principles, through the script referred to as Error_generator_cl in the attached Matlab code.

5.2.7 Current_calculation Script

The Current_calculation script uses the measurement Jacobian coefficient matrix, $\mathbf{H}$, together with the correct state vector of the system, calculated using PSS®E [13], to calculate the correct line currents in all measurement points of line currents. This includes line current measurements from both PMUs and RTUs.

This operation is optional, and did not actively take part in any of the tests conducted. It was, however, needed for the simulation of the above mentioned measurements. The script was used prior to the testing processes to complete the *measurements* matrix with correct measurements for the following simulations.

6 | Numerical Simulation and Testing

6.1 Purpose and Objective

To investigate the feasibility of the hyrid linear state estimator model, several numerical simulations were conducted using the Matlab model presented in Chapter 5, with parallel simulations run in PSS®E [13] for validation. The numerical simulations are discussed in detail throughout this chapter, with the aim of mapping both the model's fulfillment of its functionality objectives and its effectiveness compared to a classical state estimator.

6.2 Simulation Network Model

The constructed estimator application can accommodate any grid configuration, provided it is given the needed input data. The network model employed throughout the tests conducted with the estimator is part of the Statnett Tunglast Norgesmodell, a PSS®E model containing the Norwegian transmission system in a heavy load scenario. Due to its origin, the use of this model does not only serve as an assessment of the hybrid estimator as a concept, but also to investigate the viability of importing system data directly from Statnett's existing models. Specifically, the tested grid fragment is the span between the substations Alta and Kirkenes. A single-line diagram of the grid is provided in Figure 6.1. This is a 10-bus system, and is hence given the name *Finnmark 10-Bus Model.* The substations Alta, Adamselv, Varangerbotn and Kirkenes each contain an installed and networked PMU, capable of line current measurements of all connected branches.

6.3 Core Functionality Validation

Before the numerical simulations were initiated, the basic functionality of the estimator application was subjected to a short validation study. By running PSS®E's decoupled Newton-Raphson loadflow simulation, the steady-state solution of the Statnett Tunglast Norgesmodell was found. This solution was used as a baseline

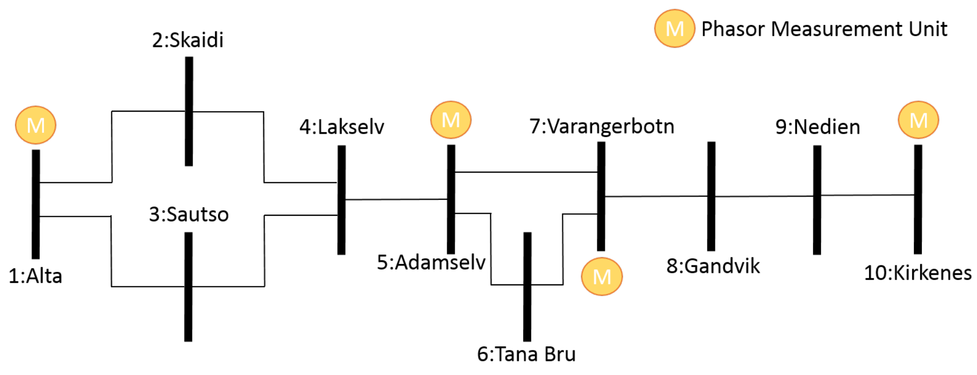


Figure 6.1: *Single-line diagram of the Finnmark 10-Bus Model.*

for the tests and considered to be the *correct*, or target, values in the following validation study. These values, including bus voltages, line currents, power injections and line power flows, were then given to the estimator application in the form of measurements. No measurement errors were added to the input date. Since the network topology and the set of constant parameters comprising the Finnmark 10-Bus Model is a fragment of the greater Statnett Tunglast Norgesmodell run in PSS®E, the expected result of running the estimator with the *correct* measurements would be a correct state vector. This was achieved.

Following this, an additional test was conducted to validate whether the second pass of the estimator application handled all types of measurements correctly. The three types of measurements given to the second pass are the following.

PMU voltage
Bus voltage measurements from buses with an installed PMU.

PMU current
Line current measurements from branches connected to buses with an installed PMU.

CSE voltage
The classically estimated state vector resulting from the first pass of the estimator. These comprise a complete list of the bus voltages.

Measurement errors were introduced to two of these sets at a time, whilst keeping the remaining set of measurements as the correct values found in PSS®E. The measurement errors were simulated and added to the measurement sets using the Error_generator script. This test was not meant to mimic a real estimation scenario, and thus the standard deviation of the probability distributions used to produce the measurement errors were arbitrarily set to 2 percent of the measured value. The estimation weight of the unaltered (correct) measurement set was gradually increased, and a Monte Carlo simulation of 200 runs were conducted for each step. Both remaining measurement sets, now affected by the simulated measurement errors, were kept at weights of 1.

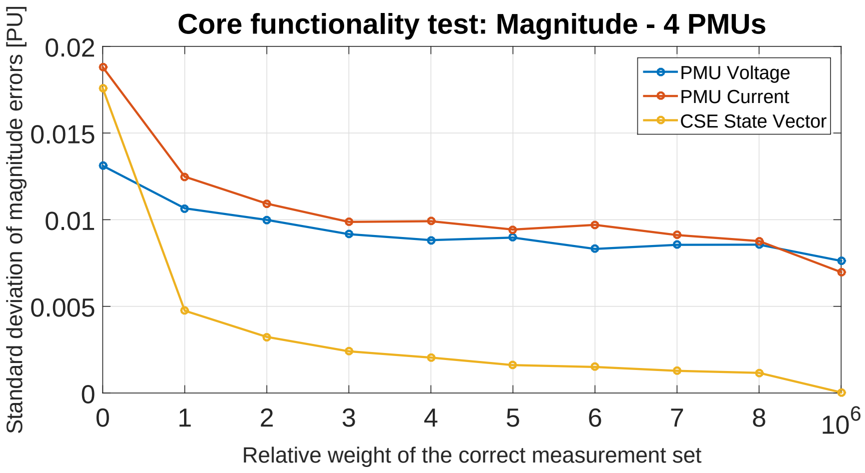
It should be noted that in this and the following simulations, the linear estimation solution as presented in Equation 4.4 is changed so that the co-variance matrix, **W**, is not inverted. This is done for the practical aspects of letting higher values in the **W** matrix result in a more predominant use of the corresponding measurement, and not the inverse, which is true for the previously presented equation.

For each Monte Carlo run, the standard deviation of the estimation error was recorded. The results of the simulations are shown in the Figures 6.2a and 6.2b below. The graphs illustrate the development of the standard deviation of magnitude and angle errors in the estimated state vector as the relative weighting of the correct measurement type increases. Each value is the mean of standard deviations across all 200 Monte Carlo runs.

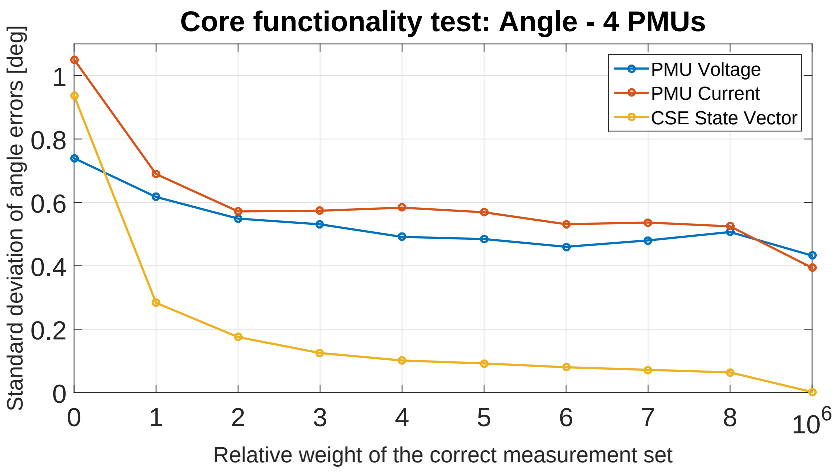
It is clear that heavier weighting of correct measurements contribute to a lower standard deviation among errors in the estimated state vector. The limited coverage of PMUs, however, restricts the achievable lower bound of the errors. This is true even at a relative weighting of one million, at which point the correct measurement set is virtually the only one deciding the state variables it is directly related to. The reason for this is the fact that there are only 4 PMUs in the Finnmark 10-Bus System, giving the estimator PMU measurements of only 4 bus voltages and 9 line currents to work with. This leaves many buses not directly connected to any of the available values without errors, thus making it impossible to reach a perfect estimate in this way.

To illustrate this effect, a similar set of Monte Carlo simulations were conducted, this time with a PMU in every single bus in the system. The results are shown in the Figures 6.3a and 6.3b below. Here, the potential lower limit of the error is evidently reduced. It should be pointed out that tests where the PMU current measurements are error free give worse results than the others. This is assumed to occur because the other two measurement types, PMU bus voltage measurements and the CSE state vector, are direct measurements of the monitored values, whilst the PMU current measurements need an additional calculation operation to reach bus voltages. This is evident considering the measurement Jacobian coefficient matrix, **H**, presented in Equation 4.2.

The simulations above conclude the validation of the estimator application's target functionality. The simulations gave the expected results when given perfect input data and demonstrated its ability to utilize the measurement weighting to its advantage in a desired manner.

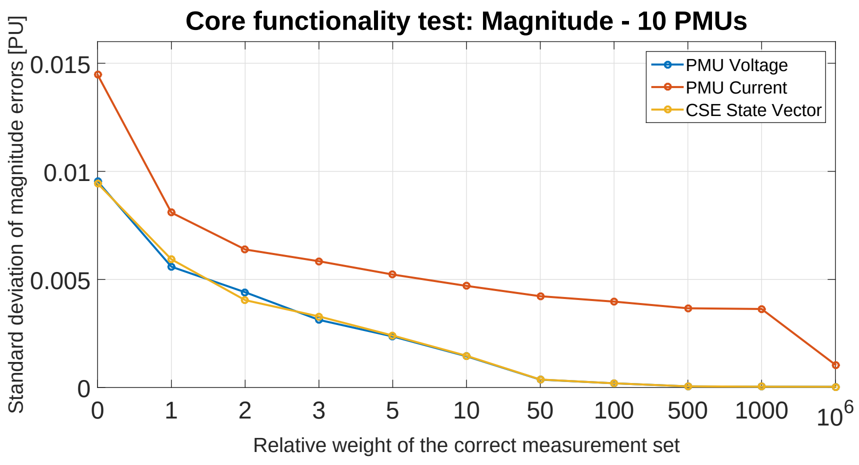


(a)

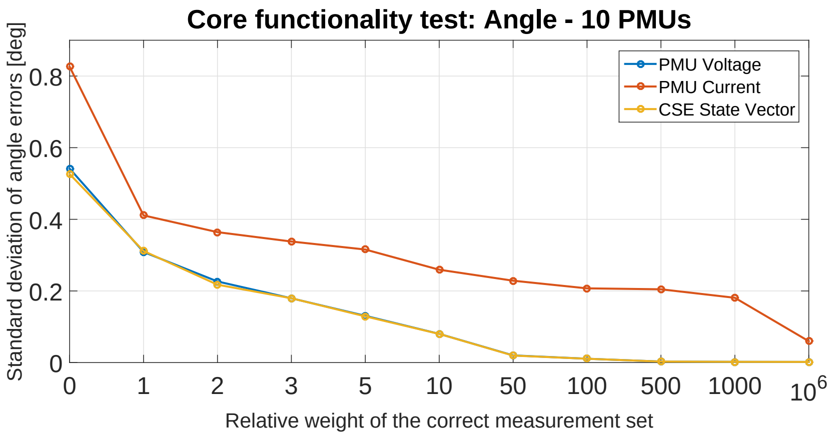


(b)

Figure 6.2: *Standard deviations of state vector errors of magnitude (a) and angle (b). Each curve represents the results of a simulation where the indicated measurement sets are kept at perfect values, whilst the remaining sets are given measurement errors. The correct measurement sets are gradually given increased weights in the estimation process. 4 PMUs were included in the model during this simulation.*



(a)



(b)

Figure 6.3: *Standard deviations of state vector errors of magnitude (a) and angle (b). Each curve represents the results of a simulation where the indicated measurement sets are kept at perfect values, whilst the remaining sets are given measurement errors. The correct measurement sets are gradually given increased weights in the estimation process. 10 PMUs were included in the model during this simulation.*

6.4 Numerical Simulations

Following the confirmation of the estimator application's functionality, the testing moved on to simulating real measurement scenarios. Three numerical simulations were performed during the testing process, divided into two different testing environments. The first two of these were steady-state Monte Carlo simulations, during which the impact of measurement weights and the number of deployed PMUs were investigated, independently. The third and last simulation was that of a dynamic response of a fault scenario. In this test, the hybrid linear state estimation model's ability to monitor the dynamic behaviour of the system was examined.

Table 6.1: *Simulation environments and the tests conducted within each of them.*

Testing environments		
Monte Carlo steady-state simulation		Time spanning dynamic simulation
Measurement weighting test	PMU coverage test	Dynamic behaviour handling test

6.4.1 Measurement Errors

Throughout the following simulations, for each Monte Carlo run and relevant time step, the Main script would run the Error_generator script for both PMU measurements and the classical state estimator with its integrated measurement error simulation. This meant that the measurement set supplied to the estimator application's second pass contained measurement errors as would be expected in a real estimation scenario. As discussed in Chapter 5, the measurement errors were simulated using a Gaussian probability distribution with a mean of zero, where a user defined standard deviation determined its span of possible outcomes.

The measurements given as input to the classical state estimator script, meaning all SCADA-collected RTU measurements, were given a standard deviation of 2 percent of the measured value. Despite these errors in the classical estimator input, the resulting estimated state vector consistently ended up with far lower errors in the steady-state environment. The classical state estimator's results in the dynamic scenario were much weaker. This was, however, in great part due to the uncertainty in measurement time more than the individual measuring errors, as will be discussed in more detail in Subsection 6.4.4.

The magnitudes of the measurement errors applied to the PMU measurements through the Error_generator script were determined by the *IEEE Standards for Synchrophasors for Power Systems* [8]. This standard states that a PMU measurement must have a *total vector error* (TVE) of less than 1 percent. For the following simulations, a standard deviation of PMU measurement errors were selected so that 1 percent of all measurements would fall outside of this limit. The calculation conducted to reach the value of this standard deviation can be found in Appendix C.

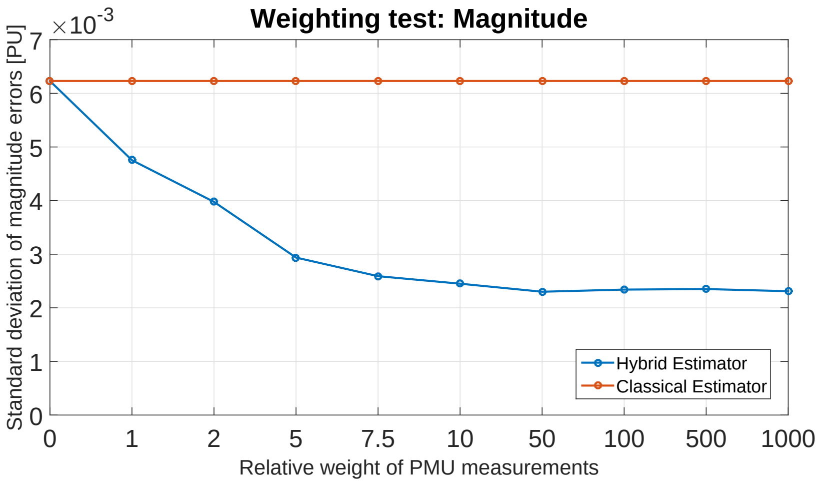
6.4.2 Measurement Weighting Test

The first of the two Monte Carlo steady-state simulations aimed to investigate the impact of PMU measurement weighting relative to those of the classically estimated state vector values. For each step of a gradual increase of the PMU measurement weights, a 1000-run Monte Carlo simulation was performed with measurement error generation as described in Subsection 6.4.1. The simulation was run on the Finnmark 10-Bus System, where 4 PMUs are installed.

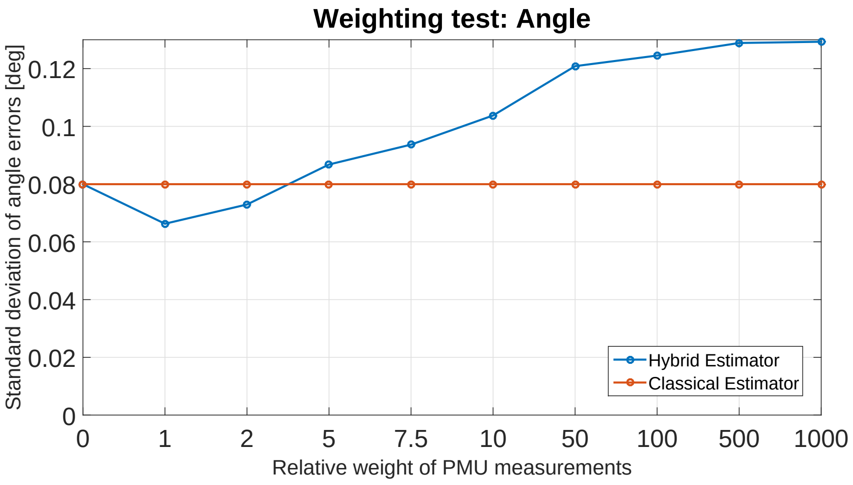
For each completed estimation, the standard deviations of the state vector error for both magnitude and angle were calculated and stored. The Figures 6.4a and 6.4b below show the mean of the 1000 standard deviations for each relative weighting preset.

Magnitude errors for the state vector components clearly improve with increasing weighting of the PMU data. At a relative weight of 5, the hybrid estimator's magnitude error standard deviation is reduced to half of that of the classically estimated state vector.

The estimation accuracy of bus voltage angles, on the other hand, ends up suffering at higher measurement weights. This is probably due to the fact that the classical state estimator results are in polar form and thus give a direct link to the angle. In comparison, the PMU line current measurement needs to go through several calculations to end up with an expression of the bus voltage angle, increasing the potential final error.



(a)



(b)

Figure 6.4: *Standard deviations of state vector errors of magnitude (a) and angle (b). The graphs illustrate the results from simulations conducted in the hybrid linear state estimator compared to the classical state estimator. The PMU measurements are gradually given increased weights in the estimation process.*

6.4.3 PMU Coverage Test

The second investigation conducted within the steady-state simulation environment studied the impact of the number of phasor measurement units installed in the tested system. The network parameters, except for the number of PMUs, was kept identical to the Finnmark 10-Bus System. Similarly, the measurement error generation remained as outlined in Subsection 6.4.1. The number of PMUs ranged from zero to 10, at which point every bus in the system would have an installed unit. For each test case, a 1000-run Monte Carlo simulation was conducted, and the standard deviations of the state vector error for both magnitude and angle were calculated and stored. The Figures 6.5a and 6.5b below show the mean of the 1000 standard deviations for each measurement unit coverage preset.

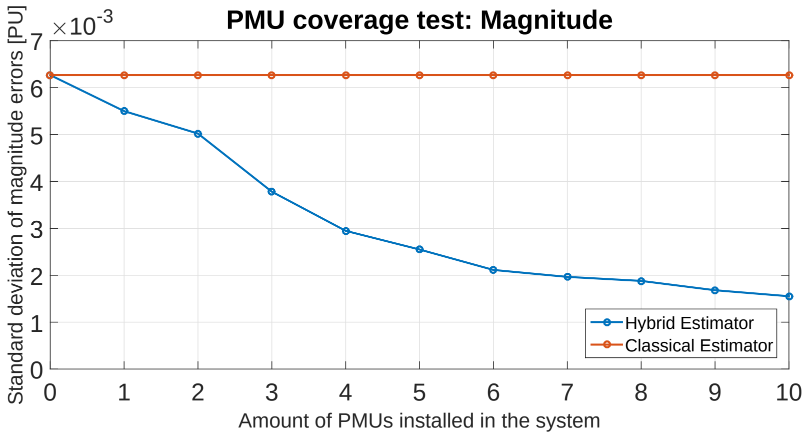
It should be mentioned that all simulated PMUs were given access to all branches connected to their substation. Due to the promising results presented in 6.4.2, the relative weighting of PMU measurements were set to five times that of the CSE state vector.

From Figure 6.5b, it is evident that the estimated bus voltage angles are not affected greatly by the number of phasor measurement units, although complete or close to complete coverage of the system generates a slightly better result than the classical state estimator.

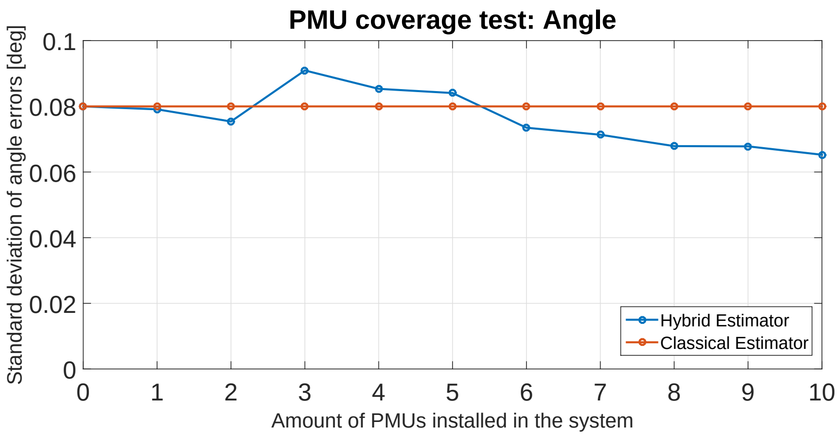
The bus voltage vector magnitude, on the other hand, experiences a clear enhancement of estimation results as the number of PMUs is increased. At 4 PMUs, equating to a coverage of 40 % of all buses, the standard deviation of magnitude errors is more than halved compared to a classical state estimator operating alone.

When reaching a high degree of PMU coverage in the system, the incremental improvement of the estimate stagnates somewhat. The introduction of the four last PMUs (from 6 to 10 installed units) each yield an average of 2,2 % better estimations compared to the CSE. Comparatively, the first five units installed (from zero to 5) enhance the magnitude estimation by an average of 11,8 % for every additional PMU.

This set of simulations concludes the steady-state Monte Carlo testing environment.



(a)

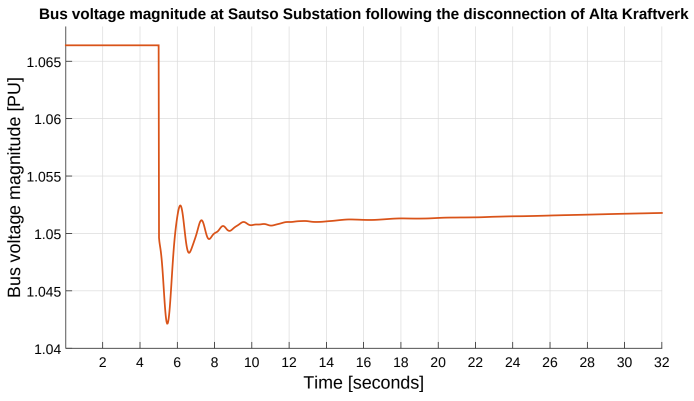


(b)

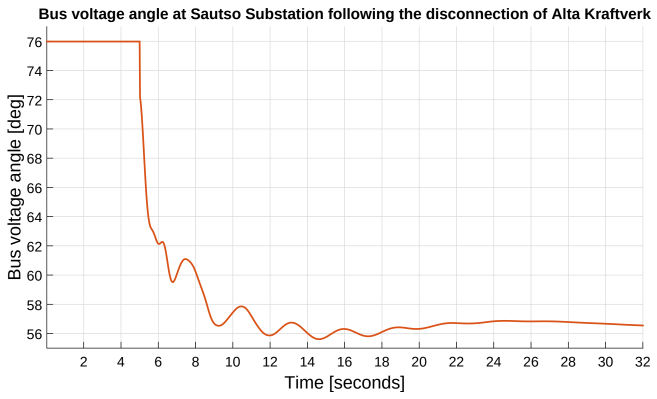
Figure 6.5: *Standard deviations of state vector errors of magnitude (a) and angle (b). The graphs illustrate the results from simulations conducted in the hybrid linear state estimator compared to the classical state estimator. The number of PMUs installed in the system are gradually increased throughout the simulations.*

6.4.4 Dynamic Behaviour Handling Test

The following simulations were conducted in a slightly different manner from the ones discussed above. Instead of the steady-state loadflow simulation previously used as the data basis, the Statnett Tunglast Norgesmodell was instead subjected to a dynamic analysis in PSS®E. The model was preparatory run under stationary conditions to reach a stable operation point. Following this introductory phase, the hydroelectric power plant, *Alta Kraftverk*, located at bus 3:Sautso, was disconnected from the system. This lead to a dynamic response in the system, causing waveforms that gradually stabilized at a new equilibrium point after approximately 15 seconds. The Figures 6.6a and 6.6b depict the dynamic behaviour of the Sautso Substation after the disconnection of Alta Kraftverk. The data collected from this PSS®E dynamic simulation was used as input for the dynamic testing environment.



(a)



(b)

Figure 6.6: *Bus voltage magnitude (a) and angle (b) at Sautso Substation following the disconnection of Alta Kraftverk. The disconnection occurs after 5 seconds.*

Under the assumption of a steady-state system and the absence of telemetry failure, both PMU and RTU measurement errors are singularly dependent on the inaccuracy of the measuring unit itself. When the measured signal becomes time dependent, another layer of measurement uncertainties is added. As mentioned in Chapter 3, it normally takes the SCADA system between 5 and 10 seconds to request and receive measurements from the system's RTUs. RTUs employ no precise time stamping system for their measurements, as opposed to PMUs, and thus the measured value might be taken from any point within the 5-10 second data collection period, in which time the measured value might change. Figure 6.7 illustrates this point, showing how a value measured within the SCADA collection time period, T_{scan}, could range from A_{min} to A_{max}.

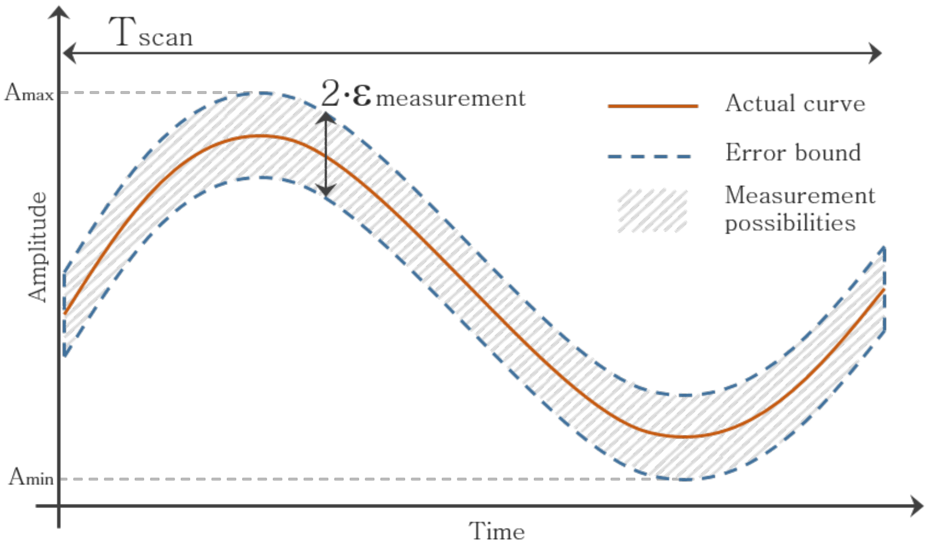


Figure 6.7: *Representation of the possible outcomes of an RTU measurement, given its time and amplitude uncertainties, here illustrated by ε and T_{scan}, respectively. Measurements may range from A_{min} to A_{max}.*

To recreate this effect, the simulated RTU measurements given as input to the hybrid estimator first pass (the classical state estimator) were chosen randomly from the previous five seconds of actual system values. As an example, consider a measurement used in a classical state estimation conducted at second 10 into the simulation. This value would be randomly picked from all the measured variable's states within the time interval spanning from T = 5 seconds to T = 10 seconds. The measurements would subsequently be subjected to the previously introduced error generation, applying the unit's inherent measurement uncertainty to the final value.

Phasor measurements are meticulously time tagged, and therefore do not possess this weakness. Because of this, the PMU measurements used in these simulations were only given errors based on the measurement unit uncertainty. All inherent unit uncertainty errors were calculated in the same manner as described in Subsection 6.4.1.

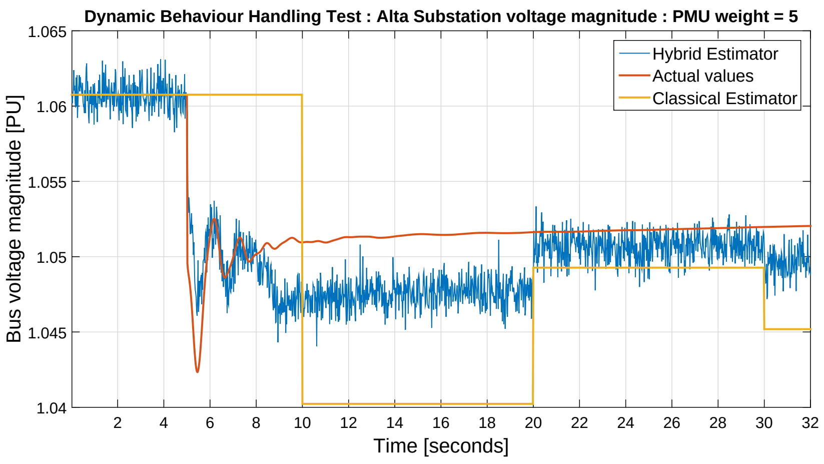
Throughout the dynamic behaviour simulation, the classical state estimator was run every 10 seconds, mirroring an ordinary control center system. Meanwhile, the second pass of the hybrid estimator application was run 50 times every second, corresponding to the reporting time of the phasor measurement units. As a result of this, the first pass results, the classically estimated state vector, would be reused as input to the second pass several hundred times before being updated by a new classical estimation process. Due to the promising results presented in 6.4.2, the relative weighting of PMU measurements were set to five times that of the CSE state vector.

The aim of this simulation was to compare the hybrid linear state estimator's ability to monitor the dynamic behaviour of a system to that of a classical state estimator. For this reason, the tests recorded the results from both the classical and hybrid estimators, as well as the actual system data as calculated by PSS®E.

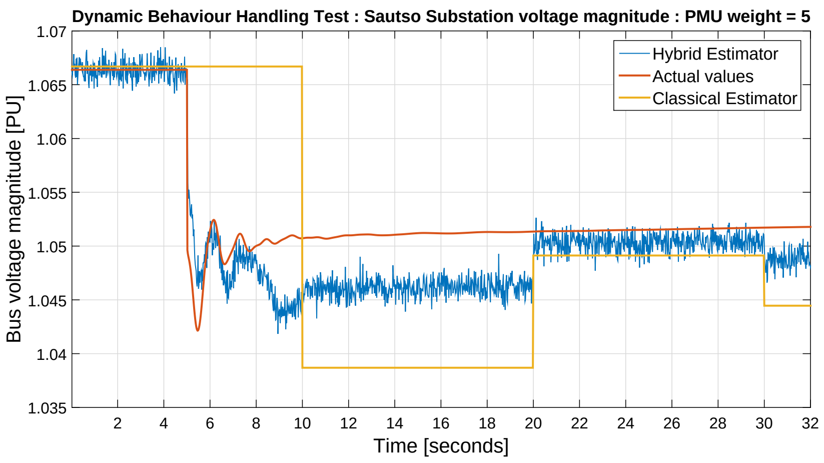
The Figures 6.8 and 6.9 illustrate the estimator results recorded in the substations Alta and Sautso. The system dynamics were monitored with near equal accuracy throughout the system, hence only these two buses are illustrated. It is immediately clear that the classical estimates (conducted at second zero, 10, 20 and 30) provide a far inferior representation of the system's dynamic behaviour than that produced by the hybrid estimator. Despite this, the hybrid estimate is undoubtedly heavily influenced by the first pass state vector. This is evident by the manner of which the hybrid estimate is frequently drawn towards the classical estimate and away from the actual values. Considering the construction of the hybrid estimator and its method of handling PMU measurements in conjunction with the first pass state vector, this effect is expected.

There are, however, exceptions to this, one of which can be observed in the bus voltage magnitude plots in Figure 6.8a and 6.8b. The hybrid estimates in the time window between second 7 and 10 pull in the opposite direction of the classically estimated values. This is probably caused by the big voltage angle discrepancy in this period, which can be seen in Figure 6.9.

The final estimates contain a substantial amount of noise due to the simulated measurement errors. This is especially true for bus voltage magnitudes, and less prevalent among the angle estimates. For practical applications, a low-pass filter should be included in the algorithm to reduce this effect.

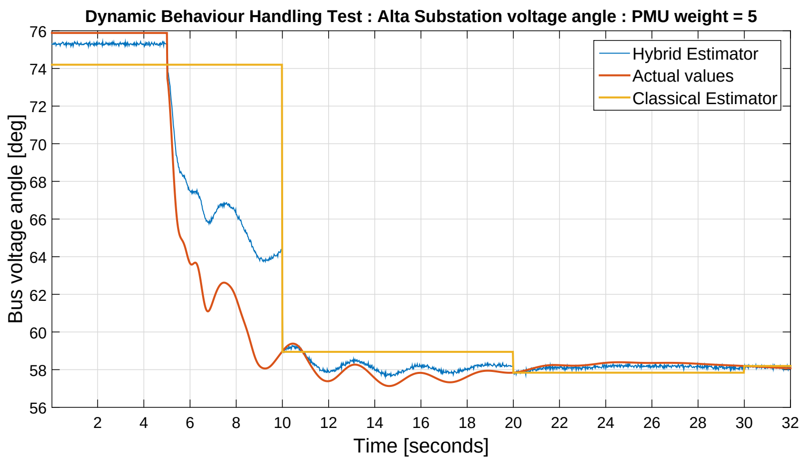


(a)

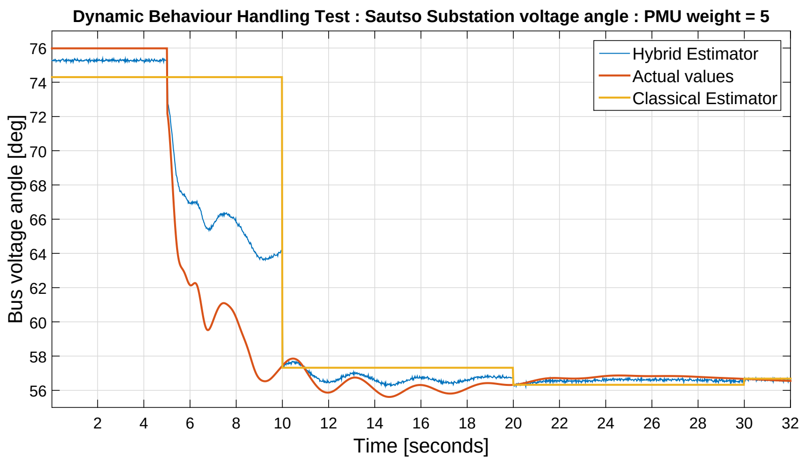


(b)

Figure 6.8: *Bus voltage magnitude at Alta (a) and Sautso (b) substations following the disconnection of Alta Kraftverk. The disconnection occurs after 5 seconds.*



(a)



(b)

Figure 6.9: *Bus voltage angle at Alta (a) and Sautso (b) substations following the disconnection of Alta Kraftverk. The disconnection occurs after 5 seconds.*

The problems outlined above often become larger as the age of the first pass state vector increases. In an attempt to combat the disruptive effects of old state vector values, a new weighting system was introduced into the simulation. The introduced concept is based on gradually decreasing the weight of state vector as it ages. When a new classical state estimation would be completed, the weight of its state vector would be restored, only to again be incrementally reduced over the subsequent 10 seconds. The following equation was introduced to the Matlab scripts.

$$Weight_{statevector} = \frac{10.000 - Age[ms]}{10.000} \tag{6.1}$$

Figure 6.10 depicts the development of the first pass state vector weight over time based on Equation 6.1.

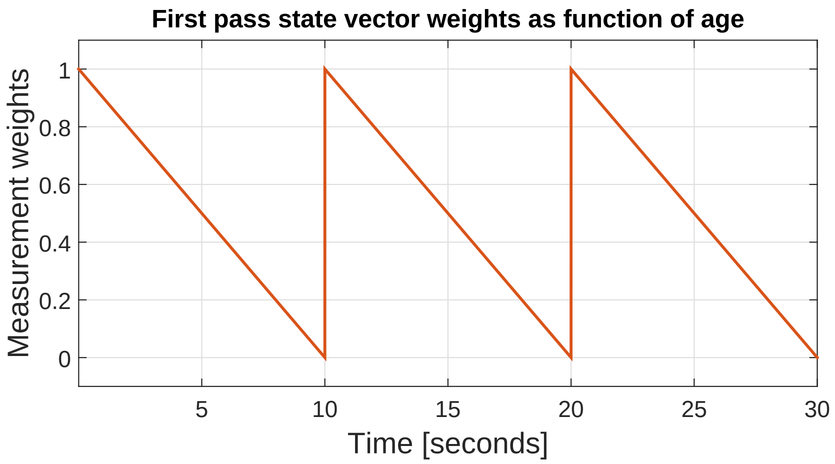
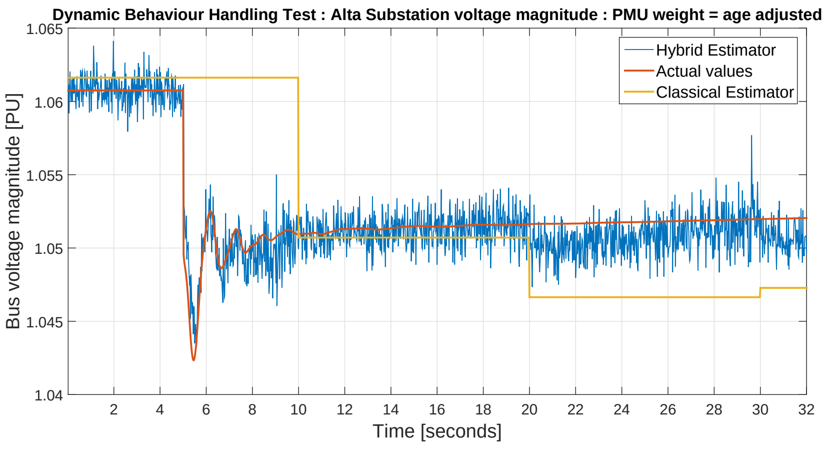
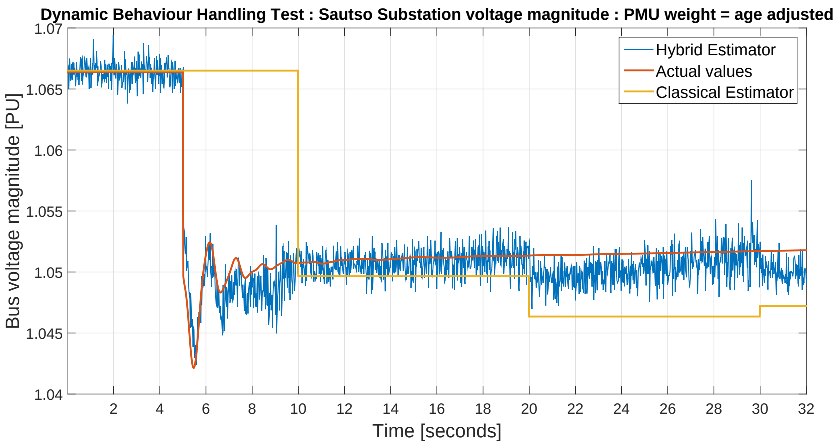


Figure 6.10: *Measurement weights of the first pass state vector as a function of its own age. A new estimated state vector is produced every 10 seconds, renewing the credibility, and thus the weights, of the values.*

The results obtained using the new weighting system are portrayed in the Figures 6.11 and 6.12. As can be observed, the angle estimation errors are greatly reduced through the process of reducing the weight of the classically estimated state vector. This, in turn, prohibits the magnitude estimation error seen in Figure 6.8.

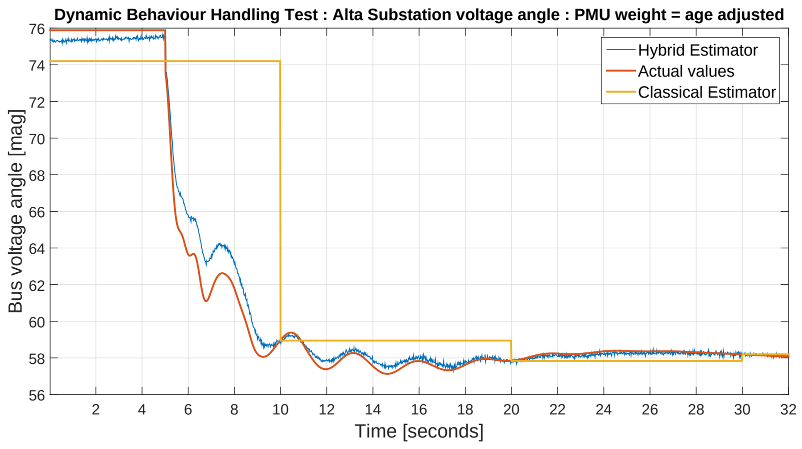


(a)

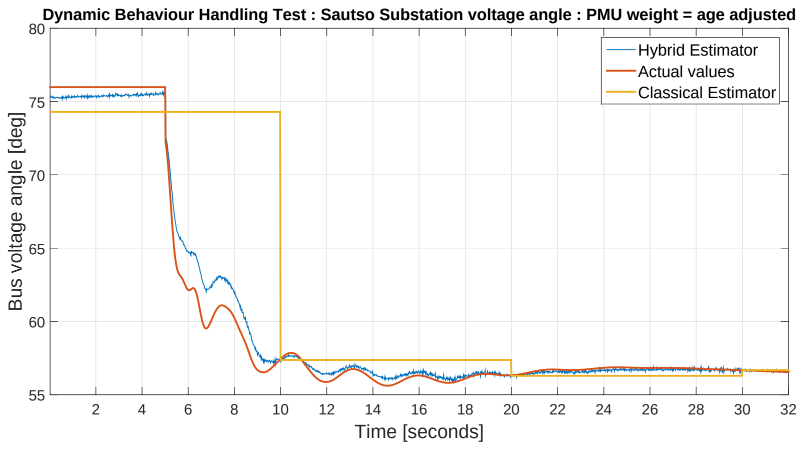


(b)

Figure 6.11: *Bus voltage magnitude at Alta (a) and Sautso (b) substations following the disconnection of Alta Kraftverk with age adjusted first pass state vector weight. The disconnection occurs after 5 seconds.*



(a)



(b)

Figure 6.12: *Bus voltage angle at Alta (a) and Sautso (b) substations following the disconnection of Alta Kraftverk with age adjusted first pass state vector weight. The disconnection occurs after 5 seconds.*

7 | Discussion and Concluding Remarks

7.1 Assumptions and Simulation Weaknesses

Throughout the processes of model construction and numerical simulation, a collection of assumptions were made. This section will highlight these assumptions and briefly discuss their impact on the simulation results.

The constructed Matlab estimator application does not have an integrated observability analysis tool. In all simulations presented in this thesis, complete observability was at all times present by virtue of the great amount of measurements used in the Finnmark 10-Bus Model, hence observability was never a relevant issue. In a real-world application of any estimator, however, observability should be considered to ensure the validity of the estimation. Observability analysis of measurement topologies has not been discussed in this work, but an extensive explanation is provided in [5].

In Subsection 6.4.3, a stepwise activation of phasor measurement units was conducted, from zero units to complete coverage. The order of activation was determined on a basis of intuition and geographical spread. There exists, however, analytical methods of optimizing placements of PMUs. The aim of these methods is to acquire the best possible range of measurement information given an available number of units. Several sources of published material discuss this issue, including the Power System Engineering Research Center report *Optimal Placement of Phasor Measurement Units for State Estimation* [14]. In the case of the numerical simulations conducted in this work, the radial nature of the test system makes the aforementioned methods virtually redundant, due to the low complexity of power flow directions within the network. It is likely that only small improvements could have been achieved by applying a more meticulous approach.

Several assumptions were made in regards to measurement errors. Firstly, the inherent measurement inaccuracy of the units themselves were simulated to be spread over a Gaussian distribution with a mean of zero. The use of the Gaussian probability distribution was a reasonable approach, assuming that all metering units discussed in this work generally adhere to this norm. The zero-mean assumption addresses any constant offset in the measurements, which in this case is considered non-existent. In practice, measurement units can develop such offsets over prolonged periods of time if not regularly adjusted, which in turn can impact state estimator results.

Furthermore, all measurement value errors were considered independent. This is also a sound assumption given the fact that most units are not directly linked in any way, and thus any discrepancy occurring in one unit should not affect the recorded measurements in the others.

The final assumption related to measurement error was that of ignoring telemetry failure. This includes both corruption of communication packets as well as any potential downtime of a unit or communications link. In other words, the simulations conducted assumed all measurement units in the system to be operational and able to transmit their measurements to the control center without complications. Both of these issues are inevitable components of a real power system and should be expected and managed accordingly when implementing a state estimator. The aim of this study has been to investigate the feasibility of the hybrid linear estimator methodology. As the prospects of unit downtime is more an issue of observability, it does not impact the feasibility analysis. The problem of communication failure introduces problems that can be handled with relative ease through the utilization of the bad data detection and elimination process discussed in Chapter 3. The lack of such corrupted measurements in the simulations meant a bad data detection and elimination module was not needed in the constructed Matlab application.

The numerical simulations heavily relied on the network analysis results provided by PSS®E. Given the coherent and consistent results produced by the Matlab estimator application, it is safe to say that the imported data was valid. It should still be noted that a loss of data accuracy could have occurred in the process of data transfer, e.g. in the form of lost decimals.

As was explained in Chapter 3, the co-variance matrix of the linear estimator model is used as a means to weigh the different types of measurements. In all of the numerical simulations performed in this work, the values selected for the measurement weights were based on the results from the measurement weighting test described in Subsection 6.4.2. In a real-world implementation of a hybrid state estimator model, however, a more rigorous approach of utilizing the individual measurement unit's accuracy should be applied. Despite straying from this principle, the performed simulations give a good indication of the capabilities of the hybrid estimator model compared to its classical counterpart.

Owing to very extensive durations of the dynamic behaviour simulations, each case was simulated only once. This reduces the ability to reach a generalized conviction with basis in the collected results. Despite this, the simulations still provided a good indication of the performance of the hyrid estimator model. It should also be noted that, although every case was simulated once, every single 32-second simulation conducted 1600 hybrid linear state estimations and four classical state estimations. The high number of estimations is a result of the chosen second pass estimation time step of 20 ms.

7.2 Evaluation of Conducted Simulations

This section will summarize the results found throughout the numerical simulations. Each separate simulation aimed to investigate one or several aspects of the hybrid linear state estimator model in comparison to the classical state estimator and identify possible benefits of the model.

The measurement weighting test illustrated the importance of measurement weights in the estimation process. The results indicate a strong potential decrease in estimated bus voltage magnitude errors by increasing the relative PMU measurement weights. The incremental impact was by far at its highest when the PMU weights were within the first few multiples of the first pass values' weights. At exceedingly high relative weights, the incremental benefit of further increase approached zero. The bus voltage angle estimates ended up suffering from higher weightings of the PMU measurements. This suggests that, under steady-state conditions, the classical approach estimates bus voltage angles as proficiently as the hybrid estimator.

The PMU coverage test studied the impact of the number of installed phasor measurement units in the monitored system. Despite the meager incremental improvements of higher degrees of coverage, it is clear that increasing the system coverage of PMUs positively influence estimation accuracy as well as reliability. This is especially true if factors like telemetry faults and unit down-time are considered, which indirectly decrease the number of available PMU measurements.

Lastly, the dynamic behaviour simulation investigated the hybrid estimator model's ability to monitor the system dynamic response, in this case caused by the disconnection of a power plant. Despite some slight errors, the hybrid linear state estimator's capabilities of detecting and monitoring waveforms in the system far exceeds that of the classical state estimator. The improvement comes in three forms. Firstly, the estimation error is decreased compared to that of the classical estimation. Secondly, the frequency of the estimations greatly improves the tracking proficiency of dynamic behaviour and unwanted waveforms. The third and last factor of improvement is the time of detection of unwanted dynamics in the system. Depending on when during the classical estimation cycle a dynamic event occurs, it could take up to 10 seconds for the control center to become aware of it. The hybrid estimator, on the other hand, registers the fault in a matter of 50-100 milliseconds, plus any communication delay.

7.3 Concluding Remarks

The introductory chapter stated that the aim of this study was to investigate the feasibility and effectiveness of the two-pass hybrid linear state estimation model as a way of utilizing the increasing availability of synchrophasor measurements to improve state estimation. The findings obtained through the numerical simulations show great improvements in state estimation results, both in a steady-state and dynamic environment, even with a relatively small number of PMUs in the system. The improvement compared to the classical state estimator was especially prevalent in the aspect of detecting dynamic behaviour in the system, due to the time-synchronization capabilities of the PMUs.

A careful selection of weighting methodology is still advised for any practical application, and the inclusion of age-adjusted measurement weights is recommended. The constructed estimator application illustrated the fact that the method is completely non-invasive to existing state estimation systems, adding to the practical appeal of the model as a way of introducing synchrophasors to state estimation. All in all, the feasibility of the two-pass hybrid linear state estimation model, as proposed by Nuqui and Phadke [9], has been confirmed.

Additionally, all testing was conducted using existing PSS®E network models owned by Statnett. This demonstrates that existing network models can be employed in the estimator model with ease, and simultaneously proves the feasibility of the model in the Norwegian transmission system. The simulations indicate that even a small number of PMUs can greatly improve system monitoring.

Given the increasing number of phasor measurement units in the Norwegian transmission grid and the growing initiative of PMU integration within Statnett, it is recommended that a pilot project is launched, employing the two-pass hybrid estimator model principles in the existing wide area monitoring system.

Bibliography

[1] The Norwegian department of Justice and Police. St.meld. nr. 22 (2007-2008), 2008.

[2] A. G. Phadke and J. S. Thorp. *Synchronized Phasor Measurements and Their Applications*. Springer, 2008.

[3] A. G. Phadke and J. S. Thorp. A new measurement technique for tracking voltage phasors, local system frequency, and rate of change of frequency. *IEEE Transactions on on PAS. Vol. No. 5*, 1983.

[4] F. C. Schweppe and J. Wildes. Power System Static-State Estimation, parts i, ii and iii. *IEEE Transactions on Power Apperatus and Systems, Vol.PAS-89*, 1970.

[5] Ali Abur and Antonio Gomez Exposito. *Power System State Estimation - Theory and Implementation*. Marcel Dekker Inc., 2004.

[6] Machowski, Bialek and Bumby. *Power System Dynamics: Stability and Control*. Wiley, 2008.

[7] The North American SynchroPhasor Initiative. North American Synchrophasor Initiative (NASPI) Program Information, Accessed 15. May 2015: http://www.http://energy.gov/oe/downloads/north-american-synchrophasor-initiative-naspi-program-information.

[8] IEEE Power Engineering Society. IEEE Standards for Synchrophasors for Power Systems. *IEEE Std C37.118-2005*, 2005.

[9] R. F. Nuqui and A. G. Phadke. Hybrid Linear State Estimation Utilizing Synchronized Phasor Measurements. *Power Tech, 2007 IEEE Lausanne*, 2007.

[10] Hadi Saadat. *Power System Analysis, third edition*. PSA Publishing, 2010.

[11] MATLAB. *version 8.4.0 (R2014b)*. The MathWorks Inc., Natick, Massachusetts, 2014.

[12] PRAVI. Power System State Estimation using WLS, Accessed 10. November 2015: http://www.mathworks.com/matlabcentral/fileexchange/23052-power-system-state-estimation-using-wls.

[13] PSS®E. *Version 33*. Siemens Industry Inc., Munich, Germany, 2012.

[14] Bei Xu and Ali Abur. Optimal Placement of Phasor Measurement Units for State Estimation, 2005.

Appendices

A | Measurement Jacobian Coefficient Equations

In this appendix, an exhaustive list of the complete expressions for each **H**-matrix segment from Equation 3.20 is given. [5]

Line power flow:

$$\frac{\delta P_{ij}}{\delta \theta_i} = V_i V_j (g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)) \tag{A.1}$$

$$\frac{\delta P_{ij}}{\delta \theta_j} = -V_i V_j (g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)) \tag{A.2}$$

$$\frac{\delta P_{ij}}{\delta V_i} = -V_j (g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)) + 2V_i (g_{ij} + g_{s,ij}) \tag{A.3}$$

$$\frac{\delta P_{ij}}{\delta V_j} == -V_i (g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)) \tag{A.4}$$

$$\frac{\delta Q_{ij}}{\delta \theta_i} = -V_i V_j (g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)) \tag{A.5}$$

$$\frac{\delta Q_{ij}}{\delta \theta_j} = V_i V_j (g_{ij} \cos(\theta_i - \theta_j) + b_{ij} \sin(\theta_i - \theta_j)) \tag{A.6}$$

$$\frac{\delta Q_{ij}}{\delta V_i} = -V_j (g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)) - 2V_i (b_{ij} + b_{s,ij}) \tag{A.7}$$

$$\frac{\delta Q_{ij}}{\delta V_j} = -V_i (g_{ij} \sin(\theta_i - \theta_j) - b_{ij} \cos(\theta_i - \theta_j)) \tag{A.8}$$

Bus power injection:

$$\frac{\delta P_i}{\delta \theta_i} = \sum_{j=1}^{n} V_i V_j(-G_{ij}\sin(\theta_i - \theta_j) + B_{ij}\cos(\theta_i - \theta_j)) - V_i^2 B_{ii} \tag{A.9}$$

$$\frac{\delta P_i}{\delta \theta_j} = V_i V_j(G_{ij}\sin(\theta_i - \theta_j) + B_{ij}\cos(\theta_i - \theta_j)) \tag{A.10}$$

$$\frac{\delta P_i}{\delta V_i} = \sum_{j=1}^{n} V_j(G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)) + V_i G_{ii} \tag{A.11}$$

$$\frac{\delta P_i}{\delta V_j} = V_j(G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)) \tag{A.12}$$

$$\frac{\delta Q_i}{\delta \theta_i} = \sum_{j=1}^{n} V_i V_j(G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j)) - V_i^2 G_{ii} \tag{A.13}$$

$$\frac{\delta Q_i}{\delta \theta_j} = V_j(-G_{ij}\cos(\theta_i - \theta_j) - B_{ij}\sin(\theta_i - \theta_j)) \tag{A.14}$$

$$\frac{\delta Q_i}{\delta V_i} = \sum_{j=1}^{n} V_j(G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j)) - V_i B_{ii} \tag{A.15}$$

$$\frac{\delta Q_i}{\delta V_j} = V_i(G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j)) \tag{A.16}$$

Line current magnitude:

$$\frac{\delta I_{ij}}{\delta \theta_i} = \frac{g_{ij}^2 + b_{ij}^2}{I_{ij}} \cdot V_i V_j \sin(\theta_i - \theta_j) \tag{A.17}$$

$$\frac{\delta I_{ij}}{\delta \theta_j} = -\frac{g_{ij}^2 + b_{ij}^2}{I_{ij}} \cdot V_i V_j \sin(\theta_i - \theta_j) \tag{A.18}$$

$$\frac{\delta I_{ij}}{\delta V_i} = \frac{g_{ij}^2 + b_{ij}^2}{I_{ij}} \cdot (V_i - V_j \cos(\theta_i - \theta_j)) \tag{A.19}$$

$$\frac{\delta I_{ij}}{\delta V_j} = \frac{g_{ij}^2 + b_{ij}^2}{I_{ij}} \cdot (V_j - V_i \cos(\theta_i - \theta_j)) \tag{A.20}$$

Bus voltage magnitude:

$$\frac{\delta V_i}{\delta \theta_i} = 0 \tag{A.21}$$

$$\frac{\delta V_i}{\delta \theta_j} = 0 \tag{A.22}$$

$$\frac{\delta V_i}{\delta V_i} = 1 \tag{A.23}$$

$$\frac{\delta V_i}{\delta V_j} = 0 \tag{A.24}$$

B | Estimator Application Matlab Scripts

This appendix provides the produced Matlab scripts that constitute the constructed two-stage hybrid linear estimator application.

B.1 Run_simulation script

B.1.1 Monte Carlo Simulation

```
%-------------------------------------------------------------%
%--------------- MONTE CARLO SIMULATION ----------------%
%-------------------------------------------------------------%

%-------------------------------------------------------------%
%----- User input for amount of Monte Carlo runs -----%
%-------------------------------------------------------------%

runs = 1000;

%-------------------------------------------------------------%
%------ Creating vectors for result collection -------%
%-------------------------------------------------------------%

% Vector for collecting standard deviation of vector
   magnitude
error_sd_mag = zeros(runs,1);

% Vector for collecting standard deviation of vector
   angle
error_sd_ang = zeros(runs,1);

```

```

%-------------------------------------------------------%
% Creating Branches and Measurements matrices with
   actual values
%-------------------------------------------------------%

%---------------- Branches ------------------------%
% [Branch_nr , Bus_f , Bus_t , Series_impedance(Z=R+jX)
   , Shunt_admittance(the total shunt admittance)]
branches = [
    1 1 2 0.22669+1.3055i 0.00526i;
    2 1 3 0.19224+0.7807i 0.00257i;
    3 2 4 0.25195+1.30183i 0.00272i;
    4 3 4 0.37478+1.52958i 0.00408i;
    5 4 5 0.49047+2.01225i 0.00404i;
    6 5 7 0.56704+1.9329i 0.00375i;
    7 5 6 0.39337+1.61331i 0.00324i;
    8 6 7 0.0871+0.35686i 0.00071i;
    9 7 8 0.300256+0.786273i 0.001505i;
    10 8 9 0.353396+0.925427i 0.001771i;
    11 9 10 0.316168+0.82794i 0.001584i
    ];

%------------------- Measurements -------------------%
% Type 1: PMU bus voltage measurement
% Type 2: PMU branch current measurement
% Type 3: Classic state estimation bus voltage value
% Commented out rows are used in the PMU coverage test
% [Unit_nr , Type , Bus_f , Bus_t , Measurement_value ,
   confidence]
measurements = [
    % Type 1: PMU bus voltage measurements
    1 1 1 0 0.2865+1.0123i 10;
    2 1 5 0 0.2111+1.0286i 10;
    3 1 7 0 0.2128+1.0161i 10;
    4 1 10 0 0.1372+1.0418i 10;
    % 0 1 2 0 0.3224+0.9981i 10;
    % 0 1 3 0 0.2742+1.0206i 10;
    % 0 1 4 0 0.2894+1.0046i 10;
    % 0 1 6 0 0.2154+1.0168i 10;
    % 0 1 8 0 0.1909+1.0249i 10;
    % 0 1 9 0 0.1634+1.0343i 10;
    % Type 2: PMU line current measurement
    5 2 1 2 0.0032749+0.029299i 10;
    6 2 1 3 -0.0076258-0.016859i 10;
    7 2 5 4 0.00023574+0.039876i 10;
    8 2 5 7 0.0037916+0.002947i 10;
    9 2 5 6 0.0045808+0.0045069i 10;
    10 2 7 5 -0.0076252-0.0021521i 10;
    11 2 7 6 -0.0041018+0.0063759i 10;
    12 2 7 8 -0.0012598-0.027938i 10;
    13 2 10 9 -0.0034527+0.030852i 10;
    % 0 2 4 2 -0.0013405+0.025753i 10;
    % 0 2 4 3 -0.0096586-0.011176i 10;
    % 0 2 4 5 -0.0043426-0.038865i 10;
    % 0 2 8 7 -0.000276+0.028242i 10;
    % 0 2 8 9 0.00011814-0.029083i 10;
    % 0 2 2 1 -0.0085624-0.027698i 10;
    % 0 2 2 4 -0.0013831-0.024921i 10;
    % 0 2 6 5 -0.0078943-0.0038159i 10;
    % 0 2 6 7 0.0033801-0.0062239i 10;
    % 0 2 3 1 0.0050135+0.01758i 10;
    % 0 2 3 4 0.0055273+0.012326i 10;
    % 0 2 9 8 -0.0019415+0.029397i 10;
    % 0 2 9 10 0.0018084-0.030614i 10;
    % Type 3: Classic state estimation bus voltage value
    14 3 1 0 0.2865+1.0123i 10;
    15 3 2 0 0.3224+0.9981i 10;
    16 3 3 0 0.2742+1.0206i 10;
    17 3 4 0 0.2894+1.0046i 10;
    18 3 5 0 0.2111+1.0286i 10;
    19 3 6 0 0.2154+1.0168i 10;
    20 3 7 0 0.2128+1.0161i 10;
    21 3 8 0 0.1909+1.0249i 10;
    22 3 9 0 0.1634+1.0343i 10;
    23 3 10 0 0.1372+1.0418i 10;
    ];


%---------------------------------------------------%
%----------- PSSE calculated bus voltages ------------%
%---------------------------------------------------%

% [Bus_nr , V_Amplitude , V_Angle]
PSSE = [
    1 1.0521 74.2;
    2 1.0489 72.1;
    3 1.0568 74.96;
    4 1.0454 73.93;
    5 1.05 78.4;
    6 1.0394 78.04;
    7 1.0381 78.17;
    8 1.0425 79.45;
    9 1.0471 81.02;
    10 1.0508 82.5
    ];

% Recalculation to rectangular coordinates
for i = 1:length(PSSE)
    PSSE_rec(i,1) = i;
    [PSSE_rec(i,2),PSSE_rec(i,3)] = pol2cart(PSSE(i,3)
       *2*pi/360 , PSSE(i,2));
end


%-----------------------------------------------------%
% Runs the estimator application the selected amount of
   times
%-----------------------------------------------------%

for t = 1:runs

% Runs error_generation functions to add measurement
   errors to measurements
    measurements = error_generator(measurements);
    error_cl = wls_function(  );

    for n = 1:num_t_3
        [X Y] = pol2cart(error_cl(n,3)*2*pi/360,error_cl
           (n,2));
        measurements(n + num_t_1 + num_t_2, 5) = X + j*Y
           ;
    end

% Initiating estimator application
    V = main( measurements , branches );
    [a, b] = size(V);

% Creating placeholder vectors for storing vector errors
    after each estimation
    errors_magnitude = zeros(a/2,1);
    errors_angle = zeros(a/2,1);

% For-loop for extraction of estimation errors
    for n = 1:a/2
```

```matlab

% Display printing of errors
        % st = ['Bus ', num2str(n), ': ', num2str(V(2*n-1)), ' ', num2str(V(2*n)) 9 '| ' num2str(PSSE_rec(n,2)) 9 num2str(PSSE_rec(n,3)) 9 9 '|  Differanse: ' num2str(V(2*n-1) - PSSE_rec(n,2)) 9 num2str(V(2*n) - PSSE_rec(n,3))];
        % disp(st);

% Calculation of estimation errors
        magnitude_estimated = sqrt ( (V(2*n-1))^2 + (V(2*n))^2 );
        magnitude_actual = sqrt ( (PSSE_rec(n,2))^2 + (PSSE_rec(n,3))^2 );
        errors_magnitude(n,1) = magnitude_estimated - magnitude_actual;

        angle_estimated = atan( V(2*n) / V(2*n-1) )*360/(2*pi);
        angle_actual = atan( PSSE_rec(n,3) / PSSE_rec(n,2) )*360/(2*pi);
        errors_angle(n,1) = angle_estimated - angle_actual;
    end

% Calculaton of the error set's standard deviation
    counter_m = 0;
    counter_a = 0;
    for n = 1:a/2
        counter_m = counter_m + (errors_magnitude(n,1))^2;
        counter_a = counter_a + (errors_angle(n,1))^2;
    end
    error_sd_mag(t,1) = sqrt ( counter_m / (a/2) );
    error_sd_ang(t,1) = sqrt ( counter_a / (a/2) );
end


%-----------------------------------------------------%
% Calculation and optional plotting of mean estimation error standard deviation across all estimation runs
%-----------------------------------------------------%
 mean_mag_error = sum(error_sd_mag) / runs
 mean_ang_error = sum(error_sd_ang) / runs
 error_sd_mag = sort (error_sd_mag);
 error_sd_ang = sort (error_sd_ang);
% plot(error_sd_mag)
% plot(error_sd_ang)
```

B.1.2 Dynamic Response Simulation

```
%-----------------------------------------------------%
%------------ DYNAMIC RESPONSE SIMULATION ------------%
%-----------------------------------------------------%

%-----------------------------------------------------%
% Creating matrices with actual pre-fault initial values
%-----------------------------------------------------%

%---------------- Branches ------------------------%
% [Branch_nr , Bus_f , Bus_t , Series_impedance(Z=R+jX)
    , Shunt_admittance(the total shunt admittance)]
branches = [
    1 1 2 0.22669+1.3055i 0.00526i;
    2 1 3 0.19224+0.7807i 0.00257i;
    3 2 4 0.25195+1.30183i 0.00272i;
    4 3 4 0.37478+1.52958i 0.00408i;
    5 4 5 0.49047+2.01225i 0.00404i;
    6 5 7 0.56704+1.9329i 0.00375i;
    7 5 6 0.39337+1.61331i 0.00324i;
    8 6 7 0.0871+0.35686i 0.00071i;
    9 7 8 0.300256+0.786273i 0.001505i;
    10 8 9 0.353396+0.925427i 0.001771i;
    11 9 10 0.316168+0.82794i 0.001584i
    ];

%----------------- Initial CSE values -----------------%
% [Unit_nr , Type , Bus_f , Bus_t , Measurement_value ,
    confidence]
measurements_se = [
    % Type 1: PMU bus voltage measurements
    1 1 1 0 0.2865+1.0123i 10;
    2 1 5 0 0.2111+1.0286i 10;
    3 1 7 0 0.2128+1.0161i 10;
    4 1 10 0 0.1372+1.0418i 10;
    % Type 2: PMU branch current measurement
    5 2 1 2 0.0032749+0.029299i 10;
    6 2 1 3 -0.0076258-0.016859i 10;
    7 2 5 4 0.00023574+0.039876i 10;
    8 2 5 7 0.0037916+0.002947i 10;
    9 2 5 6 0.0045808+0.0045069i 10;
    10 2 7 5 -0.0076252-0.0021521i 10;
    11 2 7 6 -0.0041018+0.0063759i 10;
    12 2 7 8 -0.0012598-0.027938i 10;
    13 2 10 9 -0.0034527+0.030852i 10;
    % Type 3: Classic state estimation bus voltage value
    14 3 1 0 0.2865+1.0123i 10;
    15 3 2 0 0.3224+0.9981i 10;
    16 3 3 0 0.2742+1.0206i 10;
    17 3 4 0 0.2894+1.0046i 10;
    18 3 5 0 0.2111+1.0286i 10;
    19 3 6 0 0.2154+1.0168i 10;
    20 3 7 0 0.2128+1.0161i 10;
    21 3 8 0 0.1909+1.0249i 10;
    22 3 9 0 0.1634+1.0343i 10;
    23 3 10 0 0.1372+1.0418i 10;
    ];

%--------------- Initial hybrid values ---------------%
measurements_hybrid = [
    % Type 1: PMU bus voltage measurements
    1 1 1 0 0.2865+1.0123i 10;
    2 1 5 0 0.2111+1.0286i 10;
    3 1 7 0 0.2128+1.0161i 10;
    4 1 10 0 0.1372+1.0418i 10;
    % Type 2: PMU branch current measurement
    5 2 1 2 0.0032749+0.029299i 10;
    6 2 1 3 -0.0076258-0.016859i 10;
    7 2 5 4 0.00023574+0.039876i 10;
    8 2 5 7 0.0037916+0.002947i 10;
    9 2 5 6 0.0045808+0.0045069i 10;
    10 2 7 5 -0.0076252-0.0021521i 10;
    11 2 7 6 -0.0041018+0.0063759i 10;
    12 2 7 8 -0.0012598-0.027938i 10;
    13 2 10 9 -0.0034527+0.030852i 10;
    % Type 3: Classic state estimation bus voltage value
    14 3 1 0 0.2865+1.0123i 10;
    15 3 2 0 0.3224+0.9981i 10;
    16 3 3 0 0.2742+1.0206i 10;
    17 3 4 0 0.2894+1.0046i 10;
    18 3 5 0 0.2111+1.0286i 10;
    19 3 6 0 0.2154+1.0168i 10;
    20 3 7 0 0.2128+1.0161i 10;
    21 3 8 0 0.1909+1.0249i 10;
    22 3 9 0 0.1634+1.0343i 10;
    23 3 10 0 0.1372+1.0418i 10;
    ];


%-----------------------------------------------------%
%------- Creates matrices for storing results --------%
%-----------------------------------------------------%
result_hybrid = zeros(1601,20);
result_se = zeros(4,20);


%-----------------------------------------------------%
%------- For-loop covering 32000 milli-seconds -------%
%-----------------------------------------------------%
for t = 0:32000


%-----------------------------------------------------%
% Activation of classical state estimator every 10
   seconds
%-----------------------------------------------------%
    if t == 0 || t == 10000 || t == 20000 || t == 30000
        if t == 0
            num = 1;
        elseif t == 10000
            num = 2;
        elseif t == 20000
            num = 3;
        elseif t == 30000
            num = 4;
        end

% Correction of angle reference
        [angle_reference c] = cart2pol(real(
           measurements_hybrid(1,5)) , imag(
           measurements_hybrid(1,5)));
        angle_reference = angle_reference * 360 / (2*pi)

% Creating a new set of SCADA measurements and running
   CSE
        error_cl = wls_function_2( t + 10000 ,
           angle_reference);
        for n = 1:10
```

```
            [X Y] = pol2cart(error_cl(n,3)*2*pi/360, error_cl(n,2));
            measurements_se(n + 13, 5) = X + j*Y;
            result_se(num, n * 2 - 1) = sqrt ( X^2 + Y^2 );
            result_se(num, n * 2) = atan ( Y / X ) * 360 / (2*pi);
        end


    end



%-----------------------------------------------------%
% Activation of hybrid linear state estimator every 50 milliseconds
%-----------------------------------------------------%
    if mod(t,20) == 0
        measurements_hybrid = get_values( t + 15000 , t + 15001 , 1);
        for n = 14:23
            measurements_hybrid(n,5) = measurements_se(n,5);
        end
        measurement_units = error_generator(measurement_units);
        V = estimator(measurements_hybrid , branches , t);
        [a, b] = size(V);

        for bus_n = 1:10
            result_hybrid((t/20) + 1, bus_n * 2 - 1) = sqrt ( (V(2*bus_n-1))^2 + (V(2*bus_n))^2 );
            result_hybrid((t/20) + 1, bus_n * 2) = atan( V(2*bus_n) / V(2*bus_n-1) )*360/(2*pi);
        end
    end
end

%-----------------------------------------------------%
%------------ Optional printing of results -----------%
%-----------------------------------------------------%
result_hybrid
```

```
result_se
```

B.2 Main script

```
%-----------------------------------------------------%
%-------------------- MAIN SCRIPT --------------------%
%--------------- Estimator Second Pass ---------------%
%-----------------------------------------------------%


%-----------------------------------------------------%
%--------------- Function declaration ----------------%
%- The script returns the completed state vector: V --%
%-----------------------------------------------------%

function [ V ] = main( measurements , branches )


%-----------------------------------------------------%
%---------- Measurement confidence override ----------%
% For testing purposes, the measureent weights can be overridden here%
%-----------------------------------------------------%

num_t_1 = 0; % Amount of type 1 measurements
num_t_2 = 0; % Amount of type 2 measurements
num_t_3 = 0; % Amount of type 3 measurements
for n = 1:length(measurements)
    if measurements(n,2) == 1
        num_t_1 = num_t_1 + 1;
    elseif measurements(n,2) == 2
        num_t_2 = num_t_2 + 1;
    elseif measurements(n,2) == 3
        num_t_3 = num_t_3 + 1;
    end
end
for n = 1:num_t_1                       % Type 1 weights
    measurements(n,6) =                    1;
end
for n = 1:num_t_2                       % Type 2 weights
    measurements(num_t_1+n,6) =            1;
end
```

```
for n = 1:num_t_3                           % Type 3 weights
    measurements(num_t_1+num_t_2+n,6) =    1;
end


%-----------------------------------------------------%
%-------- Initiates the reconstruction script --------%
%-----------------------------------------------------%

[ H , W , M ] = reconstruction ( branches , measurements
    );


%-----------------------------------------------------%
%------- Initiates the linear estimation script ------%
%-----------------------------------------------------%

V = linear_estimation( M, H, W );


%-----------------------------------------------------%
%-------- Optional current calculation script --------%
%-----------------------------------------------------%

%The current calculation script requires the correct
   state vector, PSSE_rec
% [Bus_nr , V_Amplitude , V_Angle]
PSSE = [
    1 1.0521 74.2;
    2 1.0489 72.1;
    3 1.0568 74.96;
    4 1.0454 73.93;
    5 1.05 78.4;
    6 1.0394 78.04;
    7 1.0381 78.17;
    8 1.0425 79.45;
    9 1.0471 81.02;
    10 1.0508 82.5
    ];
% Recalculation to rectangular coordinates
for i = 1:length(PSSE)
    PSSE_rec(i,1) = i;
    [PSSE_rec(i,2),PSSE_rec(i,3)] = pol2cart(PSSE(i,3)
       *2*pi/360 , PSSE(i,2));
end
```

```
%current_calculation ( measurements , H , PSSE_rec);

end
```

B.3 Reconstruction script

```
%--------------------------------------------------------------------%
%--------------- RECONSTRUCTION SCRIPT ---------------%
%--------------------------------------------------------------------%

%--------------------------------------------------------------------%
%--------------- Function declaration ----------------%
%----- The script returns the matrices H, W and M ----%
%--------------------------------------------------------------------%
function [ H , W , M ] = reconstruction( branches ,
    measurements )


%--------------------------------------------------------------------%
%----------- Counting measurements by types ----------%
%--------------------------------------------------------------------%
num_t_1 = 0; % Amount of type 1 measurements
num_t_2 = 0; % Amount of type 2 measurements
num_t_3 = 0; % Amount of type 3 measurements
for i = 1:length(measurements)
    if measurements(i,2) == 1
        num_t_1 = num_t_1 + 1;
    elseif measurements(i,2) == 2
        num_t_2 = num_t_2 + 1;
    elseif measurements(i,2) == 3
        num_t_3 = num_t_3 + 1;
    end
end


%--------------------------------------------------------------------%
%----------- Counting the amount of branches ---------%
%--------------------------------------------------------------------%
num_branches = length(branches);



```

```
%-----------------------------------------------------------------------%
% Changing branch-matrix series values from impedance to
    admittance
%-----------------------------------------------------------------------%
for i = 1:num_branches
    branches(i,4) = 1/branches(i,4);
end


%-----------------------------------------------------------------------%
%Changing branch/matrix shunt values from the total
   admittance to half of the total admittance
%-----------------------------------------------------------------------%
for i = 1:num_branches
    branches(i,5) = 0.5*branches(i,5);
end


%-----------------------------------------------------------------------%
%------------ Counting the amount of buses ------------%
%-----------------------------------------------------------------------%
high = max(branches);
num_buses = max(high(2),high(3));


%-----------------------------------------------------------------------%
%--------------- Creating empty H-matrix -------------%
%-----------------------------------------------------------------------%
H = zeros(2 * (num_t_1 + num_t_2 + num_t_3) , 2 *
   num_buses);


%-----------------------------------------------------------------------%
%------ Filling state estimator part of H-matrix -----%
%-----------------------------------------------------------------------%
for i = 1:(2 * num_t_3)
    H(i,i) = 1;
end


%-----------------------------------------------------------------------%
%-- Filling PMU voltage measurement part of H-matrix -%
%-----------------------------------------------------------------------%
for i = 1:num_t_1
```

```
    H(2 * num_t_3 + 2 * i - 1, 2 * measurements(i,3) - 1) = 1;
    H(2 * num_t_3 + 2 * i, 2 * measurements(i,3)) = 1;
end


%-----------------------------------------------------------%
% Pairing PMU current measurement units with their relevant branchs
%-----------------------------------------------------------%
pairing = zeros(num_t_2,1);
for i = 1:num_t_2
    for n = 1:num_branches
        if( (measurements(num_t_1 + i,3) == branches(n,2)) && (measurements(num_t_1 + i,4) == branches(n,3)) || (measurements(num_t_1 + i,3) == branches(n,3)) && (measurements(num_t_1 + i,4) == branches(n,2)) )
            pairing(i) = branches(n,1);
        end
    end
end


%-----------------------------------------------------------%
%-- Filling PMU current measurement part of H-matrix -%
%-----------------------------------------------------------%
for i = 1:num_t_2
    H(2 * (num_t_1 + num_t_3) + 2 * i - 1, 2 * measurements(num_t_1 + i,3) - 1) = real(branches(pairing(i),4) + branches(pairing(i),5));

    H(2 * (num_t_1 + num_t_3) + 2 * i - 1, 2 * measurements(num_t_1 + i,3)) = - imag(branches(pairing(i),4) + branches(pairing(i),5));

    H(2 * (num_t_1 + num_t_3) + 2 * i - 1, 2 * measurements(num_t_1 + i,4) - 1) = - real(branches(pairing(i),4));

    H(2 * (num_t_1 + num_t_3) + 2 * i - 1, 2 * measurements(num_t_1 + i,4)) = imag(branches(pairing(i),4));

    H(2 * (num_t_1 + num_t_3) + 2 * i, 2 * measurements(
        num_t_1 + i,3) - 1) = imag(branches(pairing(i),4)
         + branches(pairing(i),5));

    H(2 * (num_t_1 + num_t_3) + 2 * i, 2 * measurements(
        num_t_1 + i,3)) = real(branches(pairing(i),4) +
        branches(pairing(i),5));

    H(2 * (num_t_1 + num_t_3) + 2 * i, 2 * measurements(
        num_t_1 + i,4) - 1) = - imag(branches(pairing(i)
        ,4));

    H(2 * (num_t_1 + num_t_3) + 2 * i, 2 * measurements(
        num_t_1 + i,4)) = - real(branches(pairing(i),4));

end


%-----------------------------------------------------%
%-------------- Creating empty W-matrix --------------%
%-----------------------------------------------------%
W = zeros(2 * ( num_t_1 + num_t_2 + num_t_3 ));


%-----------------------------------------------------%
%------------------ Filling W-matrix -----------------%
%-----------------------------------------------------%
for i = 1:num_t_3 %Weights for SE voltage values
    W(2*i-1,2*i-1) = measurements(num_t_1 + num_t_2 + i
        ,6);
    W(2*i,2*i) = measurements(num_t_1 + num_t_2 + i,6);
end
for i = 1:num_t_1 %Weights for PMU voltage values
    W(2*(num_t_3+i)-1,2*(num_t_3+i)-1) = measurements(i
        ,6);
    W(2*(num_t_3+i),2*(num_t_3+i)) = measurements(i,6);
end
for i = 1:num_t_2 %Weights for PMU current values
    W(2*(num_t_3+num_t_1+i)-1,2*(num_t_3+num_t_1+i)-1) =
         measurements(num_t_1 + i,6);
    W(2*(num_t_3+num_t_1+i),2*(num_t_3+num_t_1+i)) =
        measurements(num_t_1 + i,6);
end


```

```
%----------------------------------------------------%
%-------------- Creating empty M-matrix --------------%
%----------------------------------------------------%
M = zeros(2 * ( num_t_1 + num_t_2 + num_t_3 ),1);


%----------------------------------------------------%
%------- Filling M-matrix with measurement data ------%
%----------------------------------------------------%
for i = 1:num_t_3 %SE voltage values
    M(2 * i - 1) = real(measurements(num_t_1+ num_t_2 +
        i,5));
    M(2 * i) = imag(measurements(num_t_1+ num_t_2 + i,5)
        );
end
for i = 1:num_t_1 %PMU voltage values
    M(2 * (num_t_3 + i) - 1) = real(measurements(i,5));
    M(2 * (num_t_3 + i)) = imag(measurements(i,5));
end
for i = 1:num_t_2 %PMU current values
    M(2 * (num_t_3 + num_t_1 + i) - 1) = real(
        measurements(num_t_1 + i,5));
    M(2 * (num_t_3 + num_t_1 + i)) = imag(measurements(
        num_t_1 + i,5));
end

end
```

B.4 Linear_estimation script

```
%----------------------------------------------------%
%------------- LINEAR ESTIMATION SCRIPT --------------%
%----------------------------------------------------%


%----------------------------------------------------%
%--------------- Function declaration ----------------%
%--- The script returns the final state estimate: V --%
%----------------------------------------------------%
function [ V ] = linear_estimation( M, H, W )


```

```
%-----------------------------------------------------%
%----------- Direct linear state estimation ----------%
%-----------------------------------------------------%
V = inv( transpose(H) * inv(W) * H) * transpose(H) * inv
   (W) * M;


%-----------------------------------------------------%
% Alternative form used for an inverted relation between
    measurement predominance and its weighting
%-----------------------------------------------------%
% V = inv( transpose(H) * W * H) * transpose(H) * W * M;


end
```

B.5 Error_generator script

```
%-----------------------------------------------------%
%------------- ERROR GENERATOR SCRIPT ---------------%
%-----------------------------------------------------%


%-----------------------------------------------------%
%--------------- Function declaration ----------------%
% The script returns a measurement matrix with added
   errors
%-----------------------------------------------------%
function [ measurements_updated ] = error_generator(
   measurements )


%-----------------------------------------------------%
% Storing the original measurements for optional
   printing later in the script
%-----------------------------------------------------%
measurements_unchanged = measurements;


%-----------------------------------------------------%
%------- Measurement standard deviation input --------%
%-----------------------------------------------------%
```

```matlab
sd_type_1 = 0.0027451;    % Type 1 : PMU Voltage
sd_type_2 = 0.0027451;    % Type 2 : PMU Current
sd_type_3 = 0.01;         % Type 3 : CSE State Vector


%-----------------------------------------------------%
%--- Creating probability distributions for errors ---%
%-----------------------------------------------------%
pd1 = makedist('Normal','mu',0,'sigma',sd_type_1);
pd2 = makedist('Normal','mu',0,'sigma',sd_type_2);
pd3 = makedist('Normal','mu',0,'sigma',sd_type_3);


%-----------------------------------------------------%
% Calculation and addition of errors for each measurement
%-----------------------------------------------------%
num_t_1 = 0; % Amount of type 1 measurements
num_t_2 = 0; % Amount of type 2 measurements
num_t_3 = 0; % Amount of type 3 measurements
for n = 1:length(measurements)
    if measurements(n,2) == 1
        num_t_1 = num_t_1 + 1;
    elseif measurements(n,2) == 2
        num_t_2 = num_t_2 + 1;
    elseif measurements(n,2) == 3
        num_t_3 = num_t_3 + 1;
    end
end

for n = 1:num_t_1                                   % Type 1
    pd1 = makedist('Normal','mu',0,'sigma',sd_type_1 * sqrt((real(measurements(n,5)))^2+(imag(measurements(n,5)))^2));
    measurements(n,5) =                         measurements(n,5) + random(pd1) + random(pd1)*1i;
end
for n = 1:num_t_2                                   % Type 2
    pd2 = makedist('Normal','mu',0,'sigma',sd_type_2 * sqrt( (real(measurements(num_t_1+n,5)))^2 + (imag(measurements(num_t_1+n,5)))^2));
    measurements(num_t_1+n,5) =             measurements(num_t_1+n,5) + random(pd2) + random(pd2)*1i;
end

```

```
%The error generation of type 3 measurements was removed after the addition of the classical state estimator. SCADA measurements are given measurement errors within the CSE.
%for n = 1:num_t_3                              % Type 3
%    pd3 = makedist('Normal','mu',0,'sigma',sd_type_3 * sqrt((real(measurements(num_t_1+num_t_2+n,5)))^2+(imag(measurements(num_t_1+num_t_2+n,5)))^2));
%    measurements(num_t_1+num_t_2+n,5) =    measurements(num_t_1+num_t_2+n,5) + random(pd3) + random(pd3)*1i;
%end


%-----------------------------------------------------%
% Returning the updated measurements matrix, now containing errors
%-----------------------------------------------------%
measurements_updated = measurements;


%-----------------------------------------------------%
%- Optional: printing of relative measurement errors -%
%-----------------------------------------------------%

%-----------------------------------------------------%
%------------------ Magnitude errors -----------------%
%-----------------------------------------------------%
for n = 1:(num_t_1+num_t_2)
    if n == 1
        fprintf('Type 1: PMU Voltage \n');
    end
    actual = sqrt( (real(measurements_unchanged(n,5)))^2 + (imag(measurements_unchanged(n,5)))^2 );
    measurement = sqrt( (real(measurements(n,5)))^2 + (imag(measurements(n,5)))^2 );
    error = 100 * (measurement - actual) / actual;
    st = ['Measurement vector ', num2str(n), ': ', num2str( measurement ) , 9 , 'Real vector: ', num2str( actual ) 9 'Diff: ' , num2str( error )];
    disp(st);

    if n == num_t_1
        fprintf('Type 2: PMU Current \n');
    end
end
```

```


%----------------------------------------------------------%
%--------------------- Angle errors --------------------%
%----------------------------------------------------------%
for n = 1:(num_t_1+num_t_2)
    if n == 1
        fprintf('Type 1: PMU Voltage \n');
    end
    actual = atan(imag(measurements_unchanged(n,5))/real
        (measurements_unchanged(n,5)))*360/(2*pi);
    measurement = atan(imag(measurements(n,5))/real(
        measurements(n,5)))*360/(2*pi);
    error = 100 * (measurement - actual) / actual;
    st = ['Measurement vector ', num2str(n), ': ',
        num2str( measurement ) , 9 , 'Real vector: ',
        num2str( actual ) 9 'Diff: ' , num2str( error )];
    disp(st);

    if n == num_t_1
        fprintf('Type 2: PMU Current \n');
    end
end


end
```

B.6 Current_calculationn script

```
%----------------------------------------------------------%
%------------- CURRENT CALCULATION SCRIPT --------------%
%----------------------------------------------------------%


%----------------------------------------------------------%
%---------------- Function declaration -----------------%
% The script returns nothing, but rather prints its
    results
%----------------------------------------------------------%
function [  ] = current_calculation( measurements, H ,
    PSSE_rec)

```

```

%-----------------------------------------------------------%
%----------- Counting measurements by types -----------%
%-----------------------------------------------------------%
num_t_1 = 0; % Amount of type 1 measurements
num_t_2 = 0; % Amount of type 2 measurements
num_t_3 = 0; % Amount of type 3 measurements
for i = 1:length(measurements)
    if measurements(i,2) == 1
        num_t_1 = num_t_1 + 1;
    elseif measurements(i,2) == 2
        num_t_2 = num_t_2 + 1;
    elseif measurements(i,2) == 3
        num_t_3 = num_t_3 + 1;
    end
end


%-----------------------------------------------------------%
%----------- Stores the actual state vector -----------%
%-----------------------------------------------------------%
voltage_fasit = zeros(2 * length(PSSE_rec),1);
for i = 1:length(PSSE_rec)
    voltage_fasit(2 * i - 1) = PSSE_rec(i,2);
    voltage_fasit(2 * i) = PSSE_rec(i,3);
end


%-----------------------------------------------------------%
% Calculates the correct line currents of all PMU line
   current measurement points, then prints them to the
   console
%-----------------------------------------------------------%
for i = 1:(2 * num_t_2)
    a = mod(i,2);
    if a == 1
        ifstr = ',r';
    else
        ifstr = ',i';
    end

    teller = 0;
    for n = 1:length(voltage_fasit)
        teller = teller + H(2 * (num_t_1 + num_t_3) + i,
           n) * voltage_fasit(n);
    end

    st = ['I', num2str(floor(i/2) + a), ifstr, ' = ',
       num2str(teller)] ;
    disp(st);
end

end

end
```

C | Calculation of PMU Measurement Error Standard Deviation

The requirement from [8] states that the *total vector error* (TVE) shall be less than 1 percent for any given PMU measurement. Consider a measurement $\mathbf{X}_{measured} = X_{measured,real} + i \cdot X_{measured,imag}$ opposed to the actual system value $\mathbf{X}_{actual} = X_{actual,real} + i \cdot X_{actual,imag}$. The TVE of this measurement discrepancy is given by the following.

$$TVE = \frac{|\mathbf{X}_{measured} - \mathbf{X}_{actual}|}{\mathbf{X}_{actual}} < 0.01 \tag{C.1}$$

Deconstructed into rectangular coordinates, this becomes

$$TVE = \sqrt{\frac{(X_{measured,real} - X_{actual,real})^2 + (X_{measured,imag} - X_{actual,imag})^2}{X^2_{actual,real} + X^2_{actual,real}}} < 0.01 \tag{C.2}$$

Now, the errors of the real and imaginary components of the measurements can be defined as $\varepsilon_{real} = X_{measured,real} - X_{actual,real}$ and $\varepsilon_{imag} = X_{measured,imag} - X_{actual,imag}$.

$$TVE = \sqrt{\frac{\varepsilon^2_{real} + \varepsilon^2_{imag}}{X^2_{actual,real} + X^2_{actual,real}}} < 0.01 \tag{C.3}$$

$$TVE = \varepsilon^2_{real} + \varepsilon^2_{imag} < 10^{-4} \cdot (X^2_{actual,real} + X^2_{actual,real}) \tag{C.4}$$

An assumption is made, stating that the errors of the real and imaginary components are similarly sized, so that the measurement error for both components of a PMU measurement can be scaled according to ε, where $\varepsilon_{real} \approx \varepsilon_{imag} \approx \varepsilon$.

$$TVE = 2 \cdot \varepsilon^2 < 10^{-4} \cdot (X^2_{actual,real} + X^2_{actual,real}) \tag{C.5}$$

$$TVE = |\varepsilon| < \sqrt{\frac{10^{-4}}{2}} \cdot \sqrt{X^2_{actual,real} + X^2_{actual,real}} \tag{C.6}$$

$$TVE = |\varepsilon| < \sqrt{\frac{10^{-4}}{2}} \cdot \mathbf{X}_{actual} \approx 0.007071 \cdot \mathbf{X}_{actual} \tag{C.7}$$

In a Gaussian distribution 1 percent of the possibility set lies outside of 2.575829 times the standard deviation. To achieve a distribution where 99 percent of the PMU measurements satisfy the requirements stated above, the standard deviation of the individual components, real and imaginary, is set to be the following:

$$\sigma = \frac{\sqrt{\frac{10^{-2}}{2}}}{2.575829} \cdot \mathbf{X}_{actual} = 0.00274516 \cdot \mathbf{X}_{actual} \tag{C.8}$$