5-2017

A Distributed Dynamic State Estimator Using Cellular Computational Network

Md Ashfaqur Rahman
Clemson University, marahma@clemson.edu

A Distributed Dynamic State Estimator Using Cellular Computational Network

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

---

by
Md Ashfaqur Rahman
May 2017

Accepted by:
Dr. Ganesh Kumar Venayagamoorthy, Committee Chair
Dr. Rajendra Singh
Dr. Johan Enslin
Dr. Taufiquar Khan

Abstract

The proper operation of smart grid largely depends on the proper monitoring of the system. State estimation is a core computation process of the monitoring unit. To keep the privacy of the data and to avoid the unexpected events of the system, it needs to be made fast, distributed, and dynamic. The traditional Weighted Least Squares (WLS) estimator is neither scalable, nor distributed. Increase in the size of the system increases the computation time significantly.

The estimator can be made faster in different ways. One of the major solutions can be its parallel implementation. As the WLS estimator is not completely parallelizable, the dishonest Gauss Newton method is analyzed in this dissertation. It is shown that the method is fully parallelizable that yields a very fast result. However, the convergence of the dishonest method is not analyzed in the literature. Therefore, the nature of convergence is analyzed geometrically for a single variable problem and it is found that the method can converge on a higher range with higher slopes. The effects of the slopes on multi-variable cases are demonstrated through simulation.

On the other hand, a Cellular Computational Network (CCN) based framework is analyzed for making the system distributed and scalable. Through analysis, it is shown that the framework creates an independent method for state estimation. To increase the accuracy, some heuristic methods are tested and a Genetic Algorithm (GA) based solution is incorporated with the CCN based solution to build a

hybrid estimator. However, the heuristic methods are time-consuming and they do not exploit the advantage of the dynamic nature of the states.

With the high data-rate of phasor measurement units, it is possible to extract the dynamic natures of the states. As a result, it is also possible to make efficient predictions about them. Under this situation, a predictor can be incorporated with the estimation process to detect any unwanted changes in the system. Though it is not a part of the power system to date, it can be a tool that can enhance the reliability of the grid. To implement the predictors, a special type of neural network named Elman Recurrent Neural Network (ERNN) is used.

In this dissertation, a distributed dynamic estimator is developed by integrating an ERNN based predictor with a dishonest method based estimator. The ERNN based predictor and the dishonest method based estimator are each implemented at the cell level of a CCN framework. The estimation is a weighted combination of the dishonest module and the predictor module. With this three-stage distributed computation system, it creates an efficient dynamic state estimator.

The proposed distributed method keeps the privacy and speed of the estimation process and enhances the reliability of the system. It fulfills the requirements of the deregulated energy market. It is also expected to meet the future needs of the smart grid.

Dedication

This work is dedicated to my parents who have sacrificed most of their lifetime for my proper education. I am also grateful to my beloved wife for her unwavering patience. Besides, I would like to express my gratitude to my dissertation supervisor, my academic committee members and all the fellow students of my laboratory for their generous support.

Acknowledgments

First, I would like to thank my advisor, Dr. G. Kumar Venayagamoorthy, for his unconditional support to carry out the research and write the dissertation. This work might not have been accomplished without his abundant knowledge, and professional experiences. I would also like to thank him for his patience and dedication of time. Additionally, I would like to thank Dr. Rajendra Singh, Dr. Johan Enslin, and Dr. Taufiquar Khan for serving as my advisory committee.

Second, I would like to thank my family members. None of my achievements would have been accomplished without their love and support. Besides, I would definitely like to thank all the student and non-student members of the Real-Time Power and Intelligent Systems Laboratory, for all the supports and encouragement I received during the critical times of my research.

In addition, I want to thank the Real-Time Power and Intelligent Systems (RTPIS) Laboratory (http://rtpis.org/) for providing a world-class lab facility and an excellent working environment for my research work.

Last but not least, I would like to acknowledge the National Science Foundation, US Department of Energy, and Duke Energy for providing the financial support, directly or indirectly in support of this Ph.D. dissertation research. This work is supported by the National Science Foundation under grant #1312260, US Department of Energy (DOE) under grant DE-0E000060 as well as the Duke Energy Distinguished

Professor Endowment Fund. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the financial supporters.

Table of Contents

List of Tables

List of Figures

Figure **Page**

Chapter 1

Introduction

1.1 Overview

Some of the most important infrastructures of the twenty-first century are the electric power grids. The North American grid is considered to be the single largest machine ever built by man. In order to run the grid in proper physical condition, constant monitoring of the whole grid is needed. To serve the purpose, measurements are taken at strategic points which contain errors of significant amount. State estimator is used to remove these errors from the collected measurements. The output of the state estimator is a basic need for optimal operations of the grid [1].

With the passage of time, existing power system grows larger increasing the number of buses at new areas. Addition of a new bus increases new measurements which increase the computation time for the existing methods like the Weighted Least Squares (WLS) estimator. It has a part of matrix inversion that largely depends on the size of the system. To avoid the inversion, Cholesky decomposition along with back substitution is used. It also depends on the size of the system and it is not parallelizable. With the increase of size, the increased computational complexity can

affect the optimal operation of the system. As a result a good number of research work is done on distributed state estimation [2–4].

1.1.1 Dynamic Estimation

Situational awareness in control center operations is undergoing a dramatic change with the advent of phasor measurement units (PMUs). Where the traditional estimator runs at the SCADA scan rate of one sample per 2-4 seconds, PMUs are collecting data at the rate of 30, 60, 120 and 240 samples per second [5]. The future state estimator has to run faster than the PMU rate. On top of that, to increase the reliability, a state predictor should also be included with the estimator and the whole operation needs to be completed within the time-frame of the PMU. With the PMU rate of data, it is possible to extract the dynamic nature of the states. As a result, the importance of dynamic estimator is increasing with the implementation of the PMUs.

1.1.2 Distributed Estimation

In order to solve the problem of distributability of any networked system, a new framework referred as Cellular Computational Network (CCN) is proposed in [6]. In this framework, every cell includes a moderate powerful computational unit. These units communicate with each other to exchange information related to their tasks. The output comes directly from these cells.

In Chapter 5 and 6 of the dissertation, a new state estimation approach based on CCN is investigated. Every bus contains a cell that collects the local measurements and runs its own estimation separately. The estimated states are passed to those neighboring cells which are in need of them. Thus the estimation of the full system

will conclude.

To improve the accuracy of the CCN based framework, a hybrid estimator consists of CCN and Genetic Algorithm (GA) is developed. It is found to be the most accurate estimator among the hybrids with Particle Swarm Optimization (PSO), and its two variants, Comprehensive Learning PSO (CLPSO), and Orthogonal Learning PSO (OLPSO).

Two main features of the distributed estimation, parallelism and privacy are discussed below.

1.1.2.1 Parallelism

To make the estimation process fast, it needs to be made parallel. Due to a large portion of non-parallelizable part, the WLS estimator cannot be made very fast. To solve the problem, a version of the WLS estimator known as the dishonest Gauss Newton method is shown to be completely parallelizable in this study. The method is well known in stability analysis, and it is also used for state estimation with fast decoupled method [7]. One of the main concerns is the convergence of the estimator which is not analyzed so far. In this dissertation, the effect of the slope on the range of convergence is analyzed for a single variable case and it is shown for three functions that the range increases with increased slope.

The parallel implementation of the dishonest method requires parallel processing units like the Graphics Processing Units (GPUs). GPUs are very suitable for short operations [8]. They separate the processors in multidimensional threads and blocks [9]. The structure of the dishonest method is very suitable for GPU operations that make it the fastest estimator. In this study, it is found that an equivalent accuracy requires around one tenth time for this method on GPU. This equivalence ensures a relative accuracy of more than 99%. It will become necessary tool for smart

grid operations.

1.1.2.2 Privacy

Smart grid enables a two way communication to optimize the operations of the power system. It involves strong and fast computation and communication units. In addition to the speed, the emerging smart grid technology will support deregulated energy market. This requires the privacy of the data. The advantage of the distributed estimation is that it keeps the privacy with the speed.

CCN based hybrid estimator serves the purpose of distributability. But, it has some basic differences with the dishonest method. The differences are summarized in Table 1.1.

Table 1.1: Comparison of Dishonest and Cellular Hybrid Method

Qualities	Dishonest Estimator	CCN-GA Estimator
Accuracy	High	Low
Distributable	No	Yes
Parallelizable	Yes	Yes
Speed	Very fast	Moderate
Ill-conditioned case	Fails	Works normally
Local Observability	Not needed	Needed

The two methods are developed separately and they will be presented in Chapter 4, 5, and 6. The smart grid requires a single estimator combining the qualities stated above.

1.2 Objectives of the Dissertation

The objective of the dissertation is to develop a state estimator that will run faster than the PMU rate, save the privacy of the energy market participants, and detect any large changes in the system. To serve the purposes, a predictive dynamic state estimation is developed with the help of the CCN framework and the dishonest method. It is fast, distributed, and dynamic. Using a low cost prediction model with CCN, it is able to run faster than the maximum PMU rate.

1.3 Contributions of the Dissertation

The contributions of this dissertation can be divided in two parts, major contributions, and minor contributions. Both of them are described below.

1.3.1 Major Contributions

(i) The cellular computational network framework is applied for developing a distributed estimator. It is shown that the framework can work independently and does not depend on any underlying computation method [10, 11].

(ii) The inherent parallelism of the dishonest Gauss Newton method is revealed and its effectiveness is shown on a GPU. It requires the least time among the most promising implementations of state estimation in recent times [12].

(iii) The nature of convergence of the dishonest method is analyzed and a reliable operation strategy is proposed. Under the strategy, the method is shown to converge over a high range. The analysis of convergence shows that a better point can be found than the traditional flat start for calculating the constant

Jacobian [13].

(iv) A new dynamic estimator is built incorporating the qualities of the dishonest method and a state predictor. Both of them are framed under the cellular network. The dynamism is inherited from the integration of the prediction. The final estimator is fast, and distributed.

1.3.2 Minor Contributions

(i) The general solutions for high dimension like the CLPSO, and the OLPSO are applied for state estimation for the first time. Though OLPSO improves the performance, they do not completely solve the problems of the basic PSO [11].

(ii) The CCN based method is applied in two different ways. A static estimator is developed using a layer-based architecture and its performance is shown through simulation [10].

(iii) The second CCN based approach implements a semi-dynamic hybrid estimator and it is shown to overcome most of the issues related to the PSO and its variants. It integrates GA to improve the output of the CCN [11].

(iv) A distributed state predictor is developed using CCN. It predicts the states in near future to detect any unwanted changes in the system.

1.4 Contribution to Real Power Systems

With the advancement of the power system, the nature of the operation is changing and it is creating new requirements for state estimation. There was a time when implementing power meters at different buses over large geographical area and

collecting the data from them were big challenges. Getting the most accurate estimation with the minimum number of measurements was the sole objective. Nowadays, it became easier to implement the devices and communicate with them. The increased number of measurements increased the computational load. Moreover, deregulation of the market created the necessity of the privacy of data. Additionally, with the advancement of the PMU technology, large disturbances can be detected with predictors.

Accommodating all requirements may not be possible for a single estimation method, but some requirements can be integrated based on their priority. In the proposed distributed dynamic method, the requirements of the privacy of data, fast processing and the detection of large changes are integrated. It will be able to serve any real-time application that uses the PMU rate of data.

1.5 Organization of the Dissertation

The rest of the dissertation is organized as follows. The history of power system state estimation and the background of computational intelligence and platforms are given in Chapter 2 and 3. The convergence and the speedup of the dishonest method is shown in Chapter 4. A distributed static estimator is developed using only CCN in Chapter 5. To make the cells of the static estimator simultaneous, a semi-dynamic hybrid estimator is investigated in Chapter 6. The prototype of a state predictor based on Elman Recurrent Neural Network (ERNN) is proposed in Chapter 7. Integrating the qualities of the CCN framework, the dishonest method, and the state predictor, a new estimator is proposed in Chapter 8. The dissertation is concluded with suggestions for future work in Chapter 9.

1.6 Summary

State estimation is a mandatory part of power system operation. It needs to be perfected for the operation of smart grid. Out of many qualities, two important of them, distributability and reliability, are focused in this dissertation. The final objective of this dissertation is to build a distributed dynamic estimator. It is expected to meet the requirements of the future grid.

Chapter 2

Background of State Estimation

2.1 Introduction

State estimation is the process of removing errors from the collected measurements. Measurements can be taken in the form of power flows in the transmission lines, power injections and voltage magnitudes of the buses, phase differences of connected buses, current flows etc. In power systems, the states are directly derived from the measurements of different types.

The state vector forms the set of variables with minimum cardinality which can describe the whole system. The voltage magnitudes and phase angles are taken as the state vector in nonlinear estimation. In DC estimation, only the angles are estimated. All other variables can be derived from these state variables directly.

2.2 History of State Estimation

2.2.1 Weighted Least Square Estimator

Least Square estimator was first developed by the famous scientist Carl Friedrich Gauss in 1795 [14]. Fred Schweppe proposed it for power systems in 1970 [15–17]. Most of the important properties are revealed in these three papers. The concept of dynamic state estimator is introduced in the same year in [18].

2.2.2 Dishonest Gauss Newton Method

Though the concept of an overdetermined system for power system measurements is introduced in 1970, the techniques of power flow solution existed in the literature before that. In 1967 [19], the authors mentioned that the Jacobian matrix can be kept constant which can reduce the computational complexity. This leads to the idea of dishonest method. But, this method is not well investigated in the literature [7]. This method saves a big part of the computation of WLS estimator, but takes more number of iterations to converge at current rate of estimation. As the plan is to run the estimator at a very high rate, the system should not change much in the meantime; the estimator can start with the values obtained from the previous estimation. Thus the problem of large number of iterations can be removed. In case of big changes, the fast convergence of WLS estimator with update of Jacobian matrix is comparable with the slow convergence of the dishonest method.

2.2.3 Distributed State Estimation

The parallelization of state estimation is first mentioned in 1971 [20]. A decentralized Kalman filter based dynamic estimation method is proposed in 1978 [21]. A

topology based parallelization technique is shown in [22] which is equally applicable in vectors of computers. However, the primary works on parallelized or distributed estimation considered a full scale computer for each local unit. Over time, the distributed and parallel estimator got separated based on their requirements [23]. While the distributed estimators assume a strong processor for a number of buses with limited communication between the areas, the parallel estimators focus on fast centralized estimation. The distributed one keeps the privacy of individual parts, while the parallel one remains vulnerable to large cyber-attacks.

One of the most used parallel computation device of recent times is the Graphics Processing Unit (GPU). It works in groups of threads and blocks. Though it was not anticipated in the 80s, the parallel computing devices like the GPUs are much suitable for short operations, instead of big chunks of code. So, the parallelization of state estimation has got new challenges and a few works are already done on the use of High Performance Computing (HPC) in power system state estimation [24–27].

On the other hand, there have been a good number of works on distributed estimation in recent time. In [28], a distribution method is proposed using auxiliary problem principle. Based on the fast decoupled state estimator, an algorithm consists of ten steps is proposed in [29] which makes communication between the center and the local unit during the estimation. Another multi-agent based estimator is developed for distribution systems which complete the estimation, bad data analysis, and observability analysis at the cellular level [30].

In [31], the authors used the synchrophasor data to organize the distributed results. A novel algorithm is developed based on the alternating direction method of multipliers in [2]. In [32], the authors decentralized the dynamic state estimation using unscented Kalman filter. The purpose of the proposed project is the same of this work with parallel implementation and handling random changes. A general

distributed approach for WLS state estimator [33] will also be investigated.

2.2.4 Dynamic State Estimation

As mentioned earlier, the dynamic estimation is also proposed in the same year state estimation was proposed. It did not lose its interest in the research society over these four decades. As the Kalman filter is evolving to incorporate new challenges, it is becoming more suitable for power systems. The most prominent works on dynamic state estimation includes, but are not limited to, [20, 34–43].

The very first dynamic estimator appears in [18] where the existing Kalman filter is proposed instead of the static estimator. The main disadvantage of the dynamic estimator is the state transition matrix which is designed with a uniform random variable here. This is very effective under normal condition as the change of the power system is slow and unpredictable. A variable dimension stage invariant suboptimal discrete filter is used in [20]. It reduced the computational requirement by using a linear model of the system. Keeping the dimension variable enables it to add pseudomeasurements. The work of [34] focuses on the abrupt changes and its preventions. The abrupt change can be due to any fault in the system, as well as due to bad data. It is important that it be detected properly and no false alarm should appear. The paper presented a summary on some works and analyzed the characteristics, advantages, and tradeoffs of those methods. In [35, 36], the state transition is estimated by Kalman filter which is derived with a trend factor, and large errors in the state transition equation are detected by testing an innovation process.

The difference between the tracking estimator and the dynamic estimator is analyzed and a new hybrid estimator exploiting the advantages of both of them is proposed in [37]. As the state transition of the dynamic state is not well defined,

the tracking estimator works well under sudden change. On the other hand, the dynamic estimator has the advantage of prediction under normal operations. Due to the difficulties of the nonlinear systems, extended Kalman filter (EKF) became popular over time. However, the linearization of the EKF has some major drawbacks. As a result, a new variation of Kalman filter named Unscented Kalman Filter (UKF) is introduced for nonlinear estimation in [44]. It does not require the derivative of the states which makes it much suitable than EKF. [39] adds the state constraints with the UKF. UKF is applied in power system state estimation in [42]. However, the trend of dynamic state estimation changed its direction to synchronous machine angles and speed variation as these states have trackable dynamic nature. An EKF based filter is proposed which reduces the requirements of the input signals in [41].

2.3 DC State Estimation

DC state estimation is a linearized form of the complete state estimation based on 3 assumptions, a) the resistances of the transmission lines are significantly small compared to the the corresponding reactances, b) the phase angle difference between two connected buses is small, c) the voltage magnitudes of all the buses are 1.0 per unit.

Let $\mathbf{z}$ denote an $m \times 1$ vector of all measurements in a power system such as power flows at transmission lines and power injections and loads at buses. The power flow measurements can be taken at one or both ends of a transmission line. The measurements include errors of different levels. In state estimation, the collected set of measurements is used to estimate an $n \times 1$ vector of unknown states $\mathbf{x}$. The number of measurements is usually higher than the number of state variables, i.e. $n < m$. This makes the process an overdetermined system. Let $\mathbf{H_d}$ denote an $m \times n$

matrix which represents the network topology. The linear relation can be written as,

$$\mathbf{z} = \mathbf{H_d}\mathbf{x} + \mathbf{e}, \tag{2.1}$$

where $\mathbf{e}$ denotes the measurement error vector. In general, there are three criteria that are commonly used to estimate the system states: maximum likelihood, weighted least-square, and minimum variance. When the measurement noise is Gaussian with zero mean, these criteria lead to the same estimator [1],

$$\hat{\mathbf{x}} = (\mathbf{H_d}^T\mathbf{W}\mathbf{H_d})^{-1}\mathbf{H_d}^T\mathbf{W}\mathbf{z}, \tag{2.2}$$

here, $\mathbf{W}$ is a $m \times m$ diagonal matrix which is called the noise co-variance matrix. It represents the relative weight of all measurements,

$$\mathbf{W} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2^2} & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{\sigma_3^2} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \frac{1}{\sigma_m^2} \end{bmatrix} \tag{2.3}$$

2.4 Nonlinear State Estimation

Let, $\mathbf{z}$ denotes an $m_s \times 1$ measurement vector with errors. So, the relation between $\mathbf{z}$, the nonlinear function of the measurements $h(.)$, the state vector $\mathbf{x}$, and the measurement error $\mathbf{e}$ is written as,

$$\mathbf{z} = h(\mathbf{x}) + \mathbf{e} \tag{2.4}$$

In power system state estimation, voltage magnitudes and angles are considered as the state variables as they form the set with minimum cardinality that can describe the whole system [45]. The angle of the reference bus is considered as the reference angle and all other angles are calculated with respect to that. If there are N buses, the state vector $\mathbf{x}$ can be represented as,

$$\mathbf{x} = [\theta_2\ \theta_3..\theta_N\ V_1\ V_2..V_N]^T \tag{2.5}$$

Here, θ and V, with proper subscripts, represent voltage angles and magnitudes respectively. If the number of buses in the system is N, there will be $2N-1$ state variables. In case, there is no measurement of voltage magnitude, the magnitudes also become relative and the magnitude of the reference is set to 1.0. Therefore, the number of states reduces to $2N-2$. In the process of estimation, the number of measurements exceeds the number of states to form an overdetermined system.

As mentioned earlier, $h(.)$ denotes the nonlinear relation between the states and the measurements. For example, power flows through the transmission lines from bus i to j as well as the power injections of the buses maintain the following nonlinear relationship with the bus voltage magnitudes and angles,

$$P_{ij} = V_i^2 g_{ij} - V_i V_j g_{ij} cos(\theta_{ij}) - V_i V_j b_{ij} sin(\theta_{ij}) \tag{2.6}$$

$$Q_{ij} = -V_i^2 b_{ij} + V_i V_j b_{ij} cos(\theta_{ij}) - V_i V_j g_{ij} sin(\theta_{ij}) \tag{2.7}$$

$$P_i = V_i \sum_{j \in \mathcal{M}} V_j (G_{ij} cos(\theta_{ij}) + B_{ij} sin(\theta_{ij})) \tag{2.8}$$

$$Q_i = V_j \sum_{j \in \mathcal{M}} V_j (G_{ij} sin(\theta_{ij}) - B_{ij} cos(\theta_{ij})) \tag{2.9}$$

Where, $\mathcal{M}$ represents the set of all buses connected to i, θ_{ij} represents the

difference of θ_i, and θ_j, g_{ij}, and b_{ij} represent the admittance and susceptance of transmission line ij, G, and B represent the admittance and susceptance matrices respectively.

The purpose of the state estimator is to find a value $\hat{\mathbf{x}}$ that minimizes the difference between the actual value, $\mathbf{z}$ and the estimated value, $h(\hat{\mathbf{x}})$ of the measurements. As there are multiple measurements, the accuracy is measured by the L_2-norm of the differences/residues. Minimizing the norm is the objective function of the optimization problem,

$$\min_{\hat{\mathbf{x}}} \quad ||\mathbf{z} - h(\hat{\mathbf{x}})|| \tag{2.10}$$

2.4.1 Weighted Least Square Estimation

Like other nonlinear problems, WLS estimator linearizes the system over a small range. Then it applies linear operations to get an updated value. The system is linearized again based on this updated value and uses the linear estimation. This process is repeated unless the estimated value converges. In these methods, $\mathbf{x}$ is started with a close value to the solution. In the beginning, when there is no previous value, all voltage magnitudes start as 1 and all voltage angles as 0 which is known as flat start [1],

$$\mathbf{x} = [0\ 0...0\ 1\ 1...1]^T$$

After collecting m_s measurements and constructing the Jacobian matrix $\mathbf{H}(\mathbf{x})$ at flat start, in WLS estimation, the following steps are repeated until the state vector converges to a solution,

- step 1: $\mathbf{\Delta x} = (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}(\mathbf{z} - \mathbf{h}(\mathbf{x}))$
- step 2: $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{\Delta x}$
- step 3: update $\mathbf{h}(\mathbf{x})$ with $\mathbf{x} = \mathbf{x_{n+1}}$
- step 4: update $\mathbf{H}(\mathbf{x})$ with $\mathbf{x} = \mathbf{x_{n+1}}$

Here, the matrix, $\mathbf{W}$ denotes the relative weights of the measurements that are usually taken as the inverse of the corresponding error variances. This is also known as the honest Gauss Newton method.

Though the WLS estimator search for the solution with a linear gradient, it converges very fast as the search space is very narrow and organized for most cases. However, it has the restriction of differentiability of the functions of some measurements like the current flows. Moreover, due to the linearization, it also contains the issues related to ill-conditioning that occur when the product, $\mathbf{H}^T\mathbf{W}\mathbf{H}$ becomes singular or near singular. The heuristic methods are free of these issues.

2.4.2 Dishonest Gauss Newton Method

In dishonest Gauss Newton method, step 4 of the WLS method is not executed [1]. $\mathbf{H}$ is calculated at the beginning and updated after a certain period. If $\mathbf{H}$ does not change throughout the whole process, the method is called very dishonest.

The constant $\mathbf{H}$ helps in reducing the computation of step 1. As $\mathbf{H}$ remains constant, $(\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}$ does not change. Therefore, a constant matrix can be multiplied with the vector $\mathbf{z} - \mathbf{h}(\mathbf{x})$ to complete step 1. The matrix-vector multiplication is very suitable for GPU. The three steps can be reorganized as,

- Before estimation: Calculate $\mathbf{M} = (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{W}$
- During estimation:
  - Take previous estimation, $\mathbf{x}$
  - For each measurement set, repeat the following steps for several times,
    - step *i*: Calculate residuals, $\mathbf{r} = \mathbf{z} - \mathbf{h}(\mathbf{x})$
    - step *ii*: Calculate $\Delta\mathbf{x} = \mathbf{M}\mathbf{r}$
    - step *iii*: Calculate $\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}$

Though the method described in [1] proposes the flat start values for calculating $\mathbf{H}$ before estimation, it is not mandatory from the viewpoint of computational requirements. In fact, instead of using one Jacobian, different Jacobians can be used for handling different situations of the system.

2.5 Summary

Though there are different methods for estimating the states of the power system, a unified approach is needed to reflect the important characteristics. Though dishonest method is fast and parallelizable, it needs to be distributed. The CCN framework can help distribute the jobs of the dishonest method. This is the final objective of this work.

Chapter 3

Computational Intelligence and Platforms

3.1 Introduction

Computational intelligence (CI) is a practical way of solving problems based on its nature. Out of many methods of solutions, five basic methods have taken the central place of CI - Neural Network, Genetic Algorithm, Particle Swarm Optimization, Fuzzy Logic, and Artificial Immune System. Out of the five, the first three will be discussed in this dissertation.

3.2 Cellular Computational Network

Cellular Computational Network (CCN) is a simplified version of the NN. It is primarily proposed to divide a large network into small subsystems. In power systems, it forms a computational cell at each bus. The cells complete local estimations, and exchange and update their result. As the cells run in parallel, it becomes a completely

scalable framework. The process of exchange and update of CCN based network is shown in Fig. 3.1. However, estimation at the cellular level reveals some unique aspects which are not found at the traditional distributed estimators.

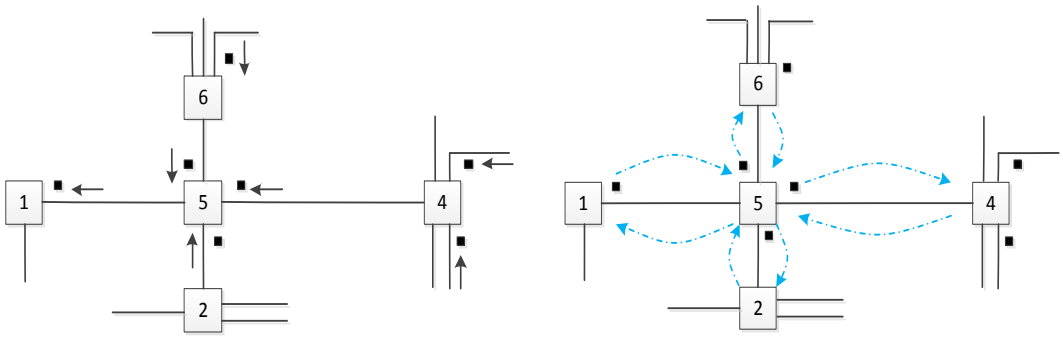

Figure 3.1: The process of exchange and update of the CCN.

3.3 Challenges

In the CCN based estimator of power system, each bus is equipped with a computational cell which collects the local data to perform a local estimation. Running the estimation of all the cells simultaneously yields two different methods - static and semi-dynamic. There are some challenges which must be met before implementing the architecture.

3.3.1 Normalization of the Powers

State estimation of a power system is done with respect to a reference bus. The angle of the reference bus is considered as zero. Moreover, the measured real and reactive powers as well as the voltage and current magnitudes are represented in per unit quantities. Usually the operating voltage of the reference bus is considered as

the base voltage, and all measured powers are converted to per unit based on that. This is helpful for the centralized state estimation.

For the cellular estimation, it creates a problem. Each cell requires a reference bus. As the measurements are normalized using the voltage angle and magnitude of the reference bus, they are needed to be normalized with the local reference bus. But, in the beginning, the voltage magnitude of the local reference bus is unknown.

The problem of normalizing can be solved in two different approaches. One of them is referred as the static estimation, another as the semi-dynamic estimation. In the static estimation, the whole system is divided in some layers. The layer-based architecture is discussed in details in Section 3.3.2.

In the semi-dynamic approach, the power flows are normalized with the previous estimated values of the reference buses. As the values do not change a lot in two consecutive samples, the values of sample t works fine for $t+1$. This approach is discussed in Chapter 7. A better solution can be achieved with the predicted values of $t+1$. The process of prediction is presented in Chapter 8.

However, the scaling of the measurements does not take much effort. From the standard power flow equations of (2.6)-(2.9), it can be seen that the angles are completely relative and they do not affect the power flow. But, the power flow measurements are dependent on the scaling of voltage magnitudes. As a result, every power flow as well as power injection measurements are needed to be scaled with the square of the voltage magnitudes of local reference bus voltage magnitude.

3.3.2 Layer Based Architecture

Though a cell cannot move forward without having the voltage magnitude of its reference bus in the static estimator, a group of cells can work simultaneously on

the same measurement set. Based on the execution of the reference cells, the whole system is divided into several layers which can be seen in Figure 3.2. The cells of the same layer work on the same measurements taken at the same time slot. The previous layer can be described as the reference layer for the next layer. When all layers are done, the states are accumulated to form the ultimate solution of the estimation problem. As a result, if the system has L layers, L time slots will be needed to have the full estimation.

Here comes a question, in the static estimation, what will the previous layer do when the next layer is estimating their states with the scaled measurements taken at time t? Do they have to wait for the whole system to complete estimation? The answer is, no. The previous layer can run with the measurements taken at time slot $t+1$ at the same time the next layer is running with measurements of time slot t. In this way, all the cells of all the layers can run simultaneously with measurements taken at different time slots.

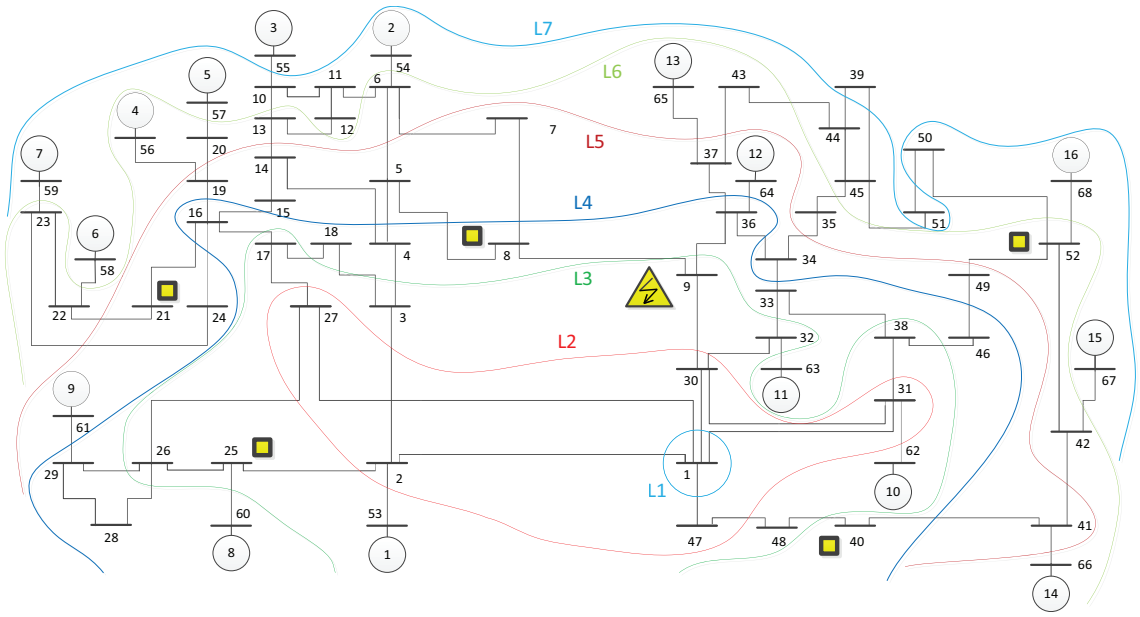

Figure 3.2: The layered IEEE 68-bus NY-NE test system. Bus 1 is the system reference bus. The simulation results are taken under a fault in line between bus 8 and 9. The estimation of five states which are marked with yellow boxes are shown in Fig. 6.4

.

3.4 Heuristic Methods

3.4.1 Genetic Algorithm

Genetic algorithm directly follows the process of evolution. It is based on the principle of the *survival for the fittest*. In GA, three main steps, selection, crossover, and mutation, are repeated over a group of chromosomes which consists of some possible solutions [46, 47]. Selection is the ordering of the chromosomes based on their fitness. Crossover exchanges genes of one chromosome with another. Mutation randomly changes a few genes of the chromosomes. The cross-over, and mutation operations are shown in Fig. 3.3.

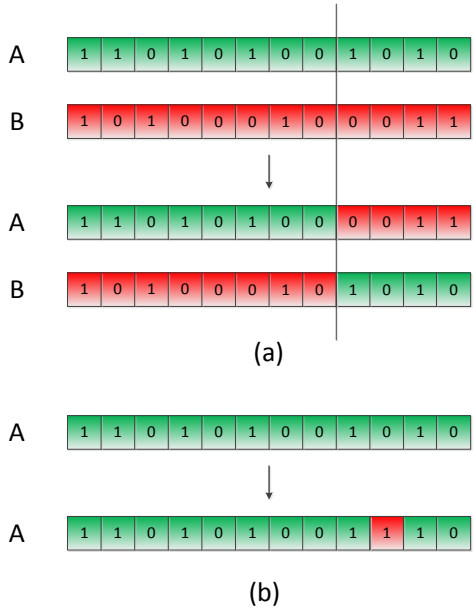

Figure 3.3: The cross-over (a), and mutation (b) operations of the genetic algorithm.

The strength of GA over random search is thoroughly analyzed with hypercubes in [48]. While the random search work on a single individual chromosome, GA works on a population that gives it the benefit of intrinsic/implicit parallelism. With the selection operation, it increases the number of those chromosomes which are better fits for the solution. Crossover, and mutation operations create small diversities to find a better solution hyperspace.

3.4.2 Particle Swarm Optimization

Inspired by the method of searching of the swarms and flocking of the birds, particle swarm optimization was first proposed by James Kennedy and Russell Eberhart in [49]. It was modified over time to adapt to new requirements of different systems. PSO has been and continues to be used in many applications of power systems [50]. In this method, a set of possible solutions are taken and the fitness

function is evaluated for each of them [51]. Based on the fitness, the velocity as well as the position of each dimension of each particle is updated towards the global and the local best solutions with some random motion [52],

$$v_{id} = wv_{id} + c_1 r_1 (P_{bd} - x_{id}) + c_2 r_2 (G_{bd} - x_{id}) \tag{3.1}$$

$$x_{id} = x_{id} + v_{id} \qquad \forall i \in \mathcal{S}, d \in \mathcal{N} \tag{3.2}$$

Here, v_{id} and x_{id} represent the velocity and the position of the d^{th} dimension of the i^{th} particle respectively. P_{bd} and G_{bd} are the local and the global best positions of the corresponding dimension. The parameters w, c_1, and c_2 are the inertia weight, the cognitive acceleration constant, and the social acceleration constant. Two random numbers r_1 and r_2 which lie in between $[0, 1]$ control the randomness of the velocity update. $\mathcal{S}$ and $\mathcal{N}$ are the sets of all particles and all dimensions. In the case of power system estimation, $\mathcal{N}$ will be the set of all state variables.

At the end of each velocity and position update, these variables are checked against their feasible limits. In case these cross their limits, these are set at the corresponding maximum or minimum limit.

3.4.3 Comprehensive Learning PSO

CLPSO was first proposed in [53] to solve the problem of dimensionality. It introduces mutation in at least one dimension of each particle [54]. The velocity update equation is modified as follows,

$$v_{id} = wv_{id} + c_1 r_1 (P_{cd} - x_{id}) \tag{3.3}$$

Here, P_{cd} can either be the particle's own local best, or it can be another particle's local best. The characteristics of CLPSO are,

- Whether a particle uses its own best or another particle's best depends on a predefined value P_c.
- For each dimension a random value is picked and it is compared with P_c of the dimension of that particle. If the random value is greater than P_c, it will learn from its own local best. Otherwise, it will randomly pick two particles and compare their fitness. Then it will pick the corresponding dimension of the better of the two.
- If all dimensions of a particle follow its own local best, a dimension is chosen randomly which takes another particle's best value.

Thus, CLPSO ensures the diversity of the particles to increase the search area. It is shown to perform better than the basic PSO for high dimensional multimodal functions.

3.4.4 Orthogonal Learning PSO

OLPSO was first proposed in [55]. In this method, an orthogonal experimental design (OED) based algorithm is used to direct the velocity of the particles. Instead of taking the local best from other particles, the local best of one particle competes with the global best. The update equation is expressed as,

$$v_{id} = wv_{id} + c_1 r_1 (P_{od} - x_{id}) \tag{3.4}$$

Here, P_{od} is chosen either from its own local best or from the global best. So, for each dimension, there are two options. But, the fitness of them should not depend

on a specific combination of other dimensions, rather it should be universal. Even with two options for each dimension, there would be 2^{2N-1} set of experiments. To solve the problem, an orthogonal experimental design (OED) method is used. With the help of an Orthogonal Array (OA), OED reduces the number of experiments to $2\times(2N-1)$. The outputs of the experiments are run through a factor analysis (FA) to select the appropriate candidate for each dimension. To avoid the stagnation at a local minimum, it updates P_{od} after certain number of iterations with no improvement.

3.5 Graphics Processing Unit

Graphics processing units were originally developed for rendering the images, animation or video on the computer screen. It is a computation intensive process. Due to its high computation power, it has got a big alternative application in parallel computation. It divides the processors in blocks and threads which fits the structure of a matrix. As a result, the implementation of a matrix or a vector becomes straightforward. The blocks are run in streaming multiprocessors (SMs) that provide the processors of moderate computation power.

3.6 Benchmark Power Systems

3.6.1 IEEE 68-bus Test System

The 16-machine, 68-bus system is basically developed as a benchmark system for stability controls. It is based on the actual New England test system (NETS) and New York power system (NYPS), with five geographical regions. A reduced order equivalent of the interconnected systems is shown in Figure 3.4 with three other neighboring areas approximated by equivalent generator models. It has 83

transmission lines.

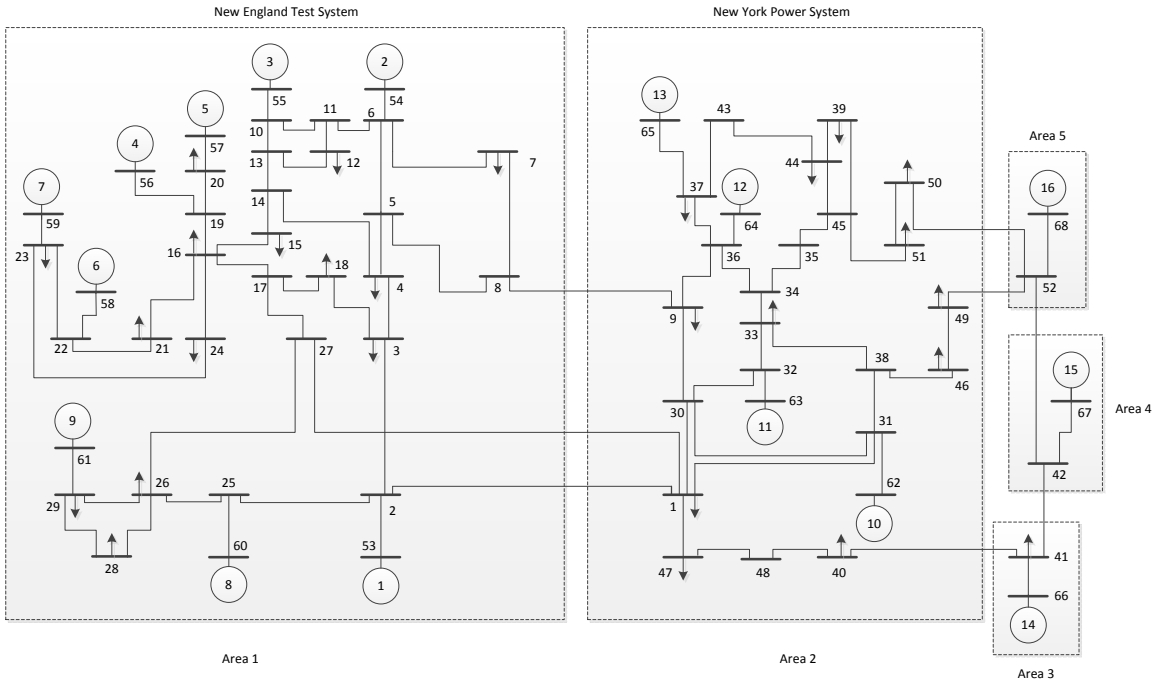

Figure 3.4: The 5-area, 16-machine, 68-bus IEEE NY-NE test system. Bus 1 is the system reference bus.

3.6.2 IEEE 118-bus Test System

The IEEE 118-bus test system represents a portion of the American Electric Power System (in the Midwestern US) as of December, 1962 [56]. It has 118 buses, 186 transmission lines, 91 loads, 9 transformers, 19 generators and 35 synchronous condensers [57].

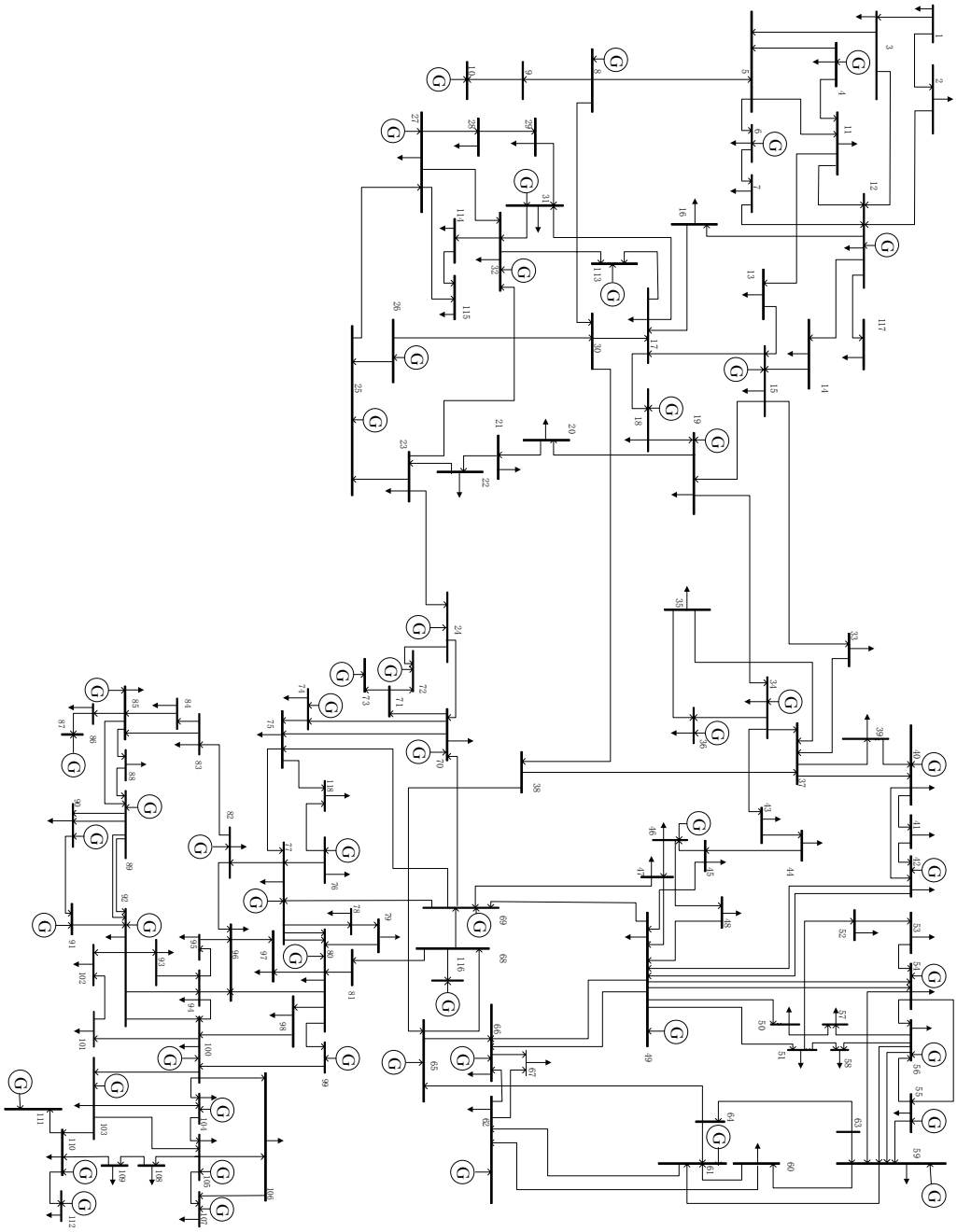

Figure 3.5: IEEE 118-bus test system.

3.7 Summary

Computational intelligence is an advanced method of computation. They can be used properly in different areas of power systems. State estimation is a potential area to test their eligibility. A semi-dynamic estimator is built in Chapter 6 using the CCN and the GA which performs better than other intelligent methods.

Chapter 4

Centralized Static Estimation Using Dishonest Gauss Method

4.1 Introduction

In the traditional SCADA system, the measurements are collected at every 2-4 seconds which is much slower than the upcoming PMU rates. In order to make the best use of the PMU data, the estimator has to run at the same speed as the PMU rate of collection. Otherwise, the collected data may have to be downsampled or we have to use the raw PMU data with the errors. One of the major solutions to this problem is to use the parallel estimators.

Parallelizability is a specific quality of any algorithm. It does not exists in most of the algorithms as they are not developed for parallel implementation. With the advent of GPU technology, parallelization has got newer dimensions. In this chapter, it will be shown that the dishonest method suits the structure of GPU, and it runs very fast. The basic dishonest method is described in Section 2.4.2. Due to the concern of the convergence, it is analyzed first in this chapter. Then the method

is implemented on a GPU for IEEE 68, and 118-bus systems.

4.2 Convergence Analysis of Dishonest Method

Before analyzing the nature of convergence of the dishonest method, it is important to illustrate the difference between the honest and the dishonest method. To make it simpler, a single variable function, $y = f(x)$ is analyzed. The applicability of this analysis on multi-variable functions will be made clear through simulations.

Let, the iterations start at $x = x_0$ with an objective of $y = y_f$ (Fig. 4.1(a)). The slope at x_0 is denoted with m_0. In the honest method, m_0 is used with the difference between $f(x_0)$ and y_f to find the new position, x_1. For x_1, the slope is calculated as m_1 and the process is repeated to find the solution.

On the other hand, the dishonest method starts with a fixed slope, m. The difference is always multiplied with this constant to find the new position of x as shown in Fig. 4.1(b). The use of a constant slope, m does not only eliminate the calculation of m, but it also changes the division operation to multiplication (m^{-1}). The contribution is not significant for a single variable system, but it becomes an important improvement for multi-dimensional large-scale systems.

However, the dishonest method does not ensure convergence for any slope, m. The choice of m depends on the functions, the region of operations, the target values, and on the starting values. Calculating the Jacobian for the extreme target and extreme starting values can make the process slow. So, for each function, the Jacobian can be developed for a normal, and some extreme conditions.

In power system state estimation, there exist a few specific types of functions between the state variables and the measurements. It is sufficient to find a suitable m for these functions. From the standard power flow equations, the major functions

can be written as,

- $P_{ij}, Q_{ij} = a_1 V_j + b_1 = f_1(V_j)$
- $P_{ij}, Q_{ij} = a_2 V_i^2 + b_2 V_i = f_2(V_i)$
- $P_{ij}, Q_{ij} = a_3 sin(\theta_{ij}) + b_3 cos(\theta_{ij}) + c_3 = f_3(\theta_{ij})$
- $\theta_{ij} = \theta_{ij}$ (PMU based phase difference)
- $V_i = V_i$

Here, the flows are measured from bus i to bus j.

The measurement of current is excluded in this study. As the last two equations do not include any function, they are skipped in this analysis as well. The power injections are the combinations of power flows; so their analysis resemble that of the flows. Before jumping to the specific functions, the nature of convergence is discussed first.

4.2.1 Nature of Convergence

The state can converge under two major scenarios. They are referred as the underdamped and the overdamped case. In the first case, there is an overshoot and it follows a zigzag path to reach the final value. In the overdamped case, there is no overshoot, and x changes monotonically to reach the final value as shown in Fig. 4.2. In some situations, a mixture of the two cases appears in the same problem.

4.2.2 Linear Functions

The analysis of the linear function is simple. The overdamped and the underdamped cases are shown in Fig. 4.3. There is no event where both cases can appear

simultaneously.

To analyze the condition for convergence, the value of m is started with the maximum. For $m \to \infty$, the process reduces to an incremental search method. By reducing the slope, the convergence can be made faster. It stays under overdamped case for $a_1 < m < \infty$, where a_1 is the slope of the line. With further reduction, the process converges up to a certain limit under underdamped case. After that, it fails to converge.

4.2.2.1 Lower limit of m

In Fig. 4.3(b), if the process starts with x_0 to find $y_f = f(x_f)$, it changes according to the following equations,

$$\begin{aligned} &at\ k = 0,\ x_0 = x_0 \\ &at\ k = 1,\ x_1 = \frac{y_f - f(x_0)}{m} + x_0 \\ &at\ k = 2,\ x_2 = \frac{y_f - f(x_1)}{m} + x_1 \end{aligned} \tag{4.1}$$

If x_2 is closer to x_f than x_0, it will be able to converge. In case of $x_f > x_0$, the condition of convergence can be written as,

$$\begin{aligned} &x_2 > x_0 \\ &\Rightarrow m > \frac{y_f - f(x_0)}{f^{-1}(2y_f - f(x_0)) - x_0} \end{aligned} \tag{4.2}$$

The expression of (4.2) is common for any system with monotonically increasing slope. For linear functions, it can be written as,

$$m > \frac{a_1(y_f - f(x_0))}{-b_1 - a_1 x_0 + 2y_f - f(x_0)} \tag{4.3}$$

$$as,\ f^{-1}(x) = \frac{x - b_1}{a_1}$$

By replacing $y = f(x) = a_1 x + b_1$ in (4.3), the final expression can be derived as,

$$m > \frac{a_1}{2} \tag{4.4}$$

For linear functions, (4.4) shows that the minimum slope does not depend on the starting or the final value. It only depends on the slope of the line. If the quadratic and sinusoidal functions can be linearized over a small portion, it is also applicable for that. This is the proof why a constant Jacobian always works for a change over the linear region of the system.

However, it is noticeable that the best value for m is not the value given by (4.2) or (4.4). Using a marginal value can lead to a very large number of iterations. Those are the minimum values for which convergence can be secured. The best value for a linear function is the constant slope, i. e., $m = a_1$.

4.2.3 Quadratic Functions

The underdamped and the overdamped cases for the positive side of a quadratic function are shown in Fig. 4.4. By taking a large value for m, convergence can always be ensured. But, having a big m makes the steps small and the number of iterations increases. The slope should be taken in such a way that it ensures convergence within a limited number of iterations.

If the slope is reduced, a mixture of the overdamped and the underdamped response is found first. With further reduction, a complete underdamped case is found as shown in Fig. 4.4(b).

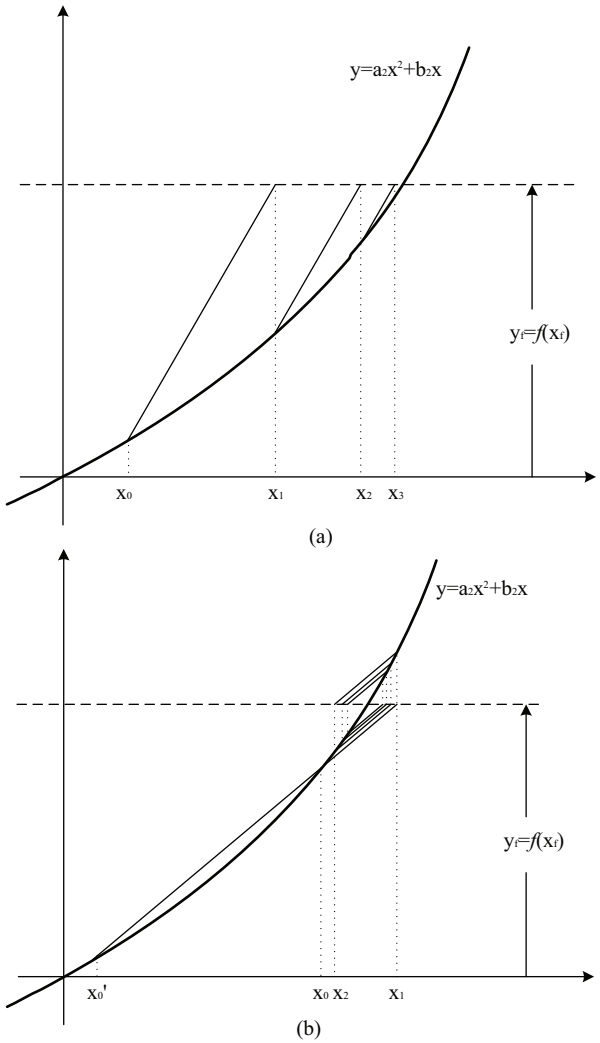

Figure 4.4: Convergence of a quadratic function in two different ways when the objective is higher than the starting value, (a) overdamped, (b) underdamped.

4.2.3.1 Lower limit of m

The analysis is the same as the linear functions. For quadratic functions of P_{ij}, and Q_{ij}, $f^{-1}(.)$ can be written as,

$$f^{-1}(x) = \frac{-b_2 \pm \sqrt{b_2^2 + 4a_2 x}}{2a_2} \tag{4.5}$$

$$where,\ f(x) = a_2 x^2 + b_2 x \tag{4.6}$$

As the voltage magnitude can only be positive, the expression of (4.2) can be written as,

$$m > \frac{2a_2(y_f - f(x_0))}{-b_2 - 2a_2 x_0 + \sqrt{b_2^2 + 4a_2(2y_f - f(x_0))}} \tag{4.7}$$

Any value of m above this value will make the system converging. The minimum value depends on y_f, x_0, a_2, and b_2. The value increases with the increase of y_f. The relation between m, and x_0 is complicated. To avoid the complication, the slope at maximum possible x_f is taken as the value of m that works for every x_0. If x_0 is close to x_f, it is the most efficient slope as analyzed in Section 4.2.2. If not, the process operates in the overdamped case that is inefficient, but it is still better than calculating a new Jacobian for practical power systems as shown in Section 4.7.3. For the 118-bus system, one iteration of the WLS estimator takes around 42 times more time than the dishonest one, while around six iterations of dishonest method gains the same accuracy of the WLS method.

It is not expected that the starting point will always be lower than the target value; it may also be at a higher position. In case of $x_f < x_0$, the searching occurs in the downward slope. The two cases are shown in Fig. 4.5. The analysis is very much similar to the upward case, and a similar expression can be derived.

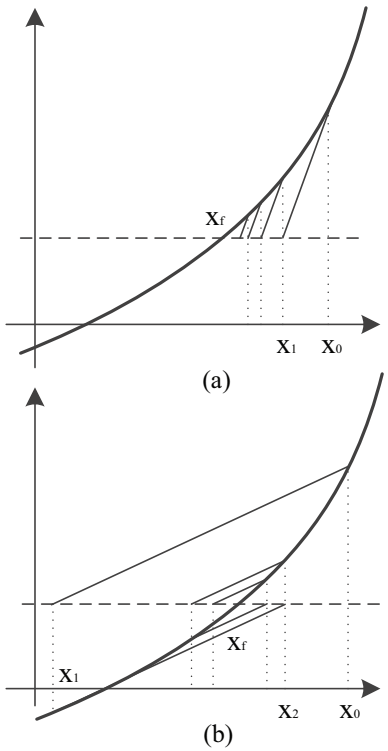

Figure 4.5: Convergence of a quadratic function for two different ways of convergence when the objective value is lower than the starting value, (a) overdamped, (b) underdamped.

However, there has to be a single m irrespective of the position of x_f. The problem is resolved by taking the solution for the upward search. It is evident from the fact that for any x_{fu}, x_{fd}, and x_0, if $x_{fd} < x_0 < x_{fu}$, then, $m_u > m_d$. It can also be inferred that there will be only the overdamped case for the downward search with this value of m.

4.2.4 Sinusoidal Functions

Though the two typical cases can appear in sinusoidal operations, it is a bit complicated, as the slope of the function does not increase monotonically. The overdamped case is simple as shown in Fig. 4.6(a). In the underdamped case, if the slope

at x_1 is less than m, the convergence cannot be ensured. So, (4.2) is applicable in a range of x where the slope is greater than m.

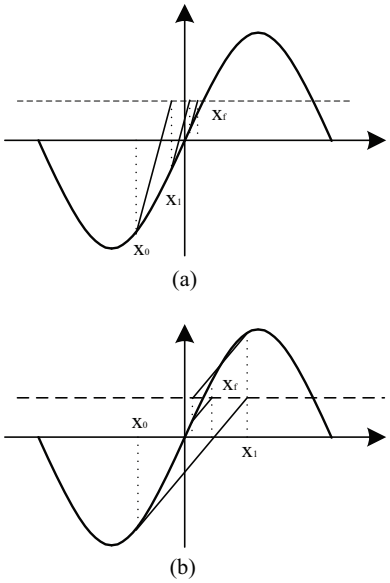

Figure 4.6: Two cases of convergence for sinusoidal function, (a) overdamped, (b) underdamped. As the slope at x_1 cuts the function at another point, the convergence cannot be ensured.

For power system, the phase angle of a bus with respect to the reference bus can vary a lot. Two connected buses usually keep a constant phase difference to maintain an expected real power flow between them. In case of typical faults, disturbances, or sudden load changes, the phase difference of the connected buses can change, but it usually does not exceed $\pm 20°$. In this short region, the sinusoidal function can be considered linear and the analysis for linear function can be applied. An easier choice is the maximum slope of this region that can ensure convergence. The maximum slope for a sine function occurs at $\theta = 0$ and that for a cosine function occurs at $\theta = \frac{\pi}{2}$. Due to the comparative values of a_3, and b_3, a value close to zero is preferred. However, taking the maximums slope can make it a slower process, and a better value can be obtained by choosing the closest possible value. For any transmission line ij, the

angle corresponding to the maximum slope, θ_m can be found with Algorithm 4.2.4.

Algorithm 1 Selection of θ_{ij}

1: $\theta_{min} = min(possible\ values\ of\ \theta_{ij})$
2: $\theta_{max} = max(possible\ values\ of\ \theta_{ij})$
3: **if** $\theta_{min} > 0$ **then**
4: $\quad \theta_m = \theta_{min}$ // closest to zero
5: **else if** $\theta_{max} < 0$ **then**
6: $\quad \theta_m = \theta_{max}$ // closest to zero
7: **else**
8: $\quad \theta_m = 0$
9: **end if**

4.3 Multi-Jacobian Method

In the previous section, it is shown that the constant Jacobian calculated at the nominal values can fail with diverse states. On the other hand, a Jacobian with higher slopes can converge slower than the Jacobian with the nominal slope. A combined effort yields the proper solution.

The proposed method is a simple addition to the existing method. From Fig. 4.13, it is clear that there exists a trade-off between the range and the speed of convergence for the dishonest method. The higher the slope, the bigger the range, and the slower the speed. As the power system rarely runs into any non-converging situation with nominal Jacobian, the main process runs with the existing method.

At the same time, a few optional Jacobians are added in the process. The optional Jacobians are calculated with larger slopes ($|V_i| = 1.2, 1.4$ etc.) and saved before starting the process. In case the nominal Jacobian fails, the options can be tried one by one as shown in Fig. 4.7. The number of options can be set with practical experiences.

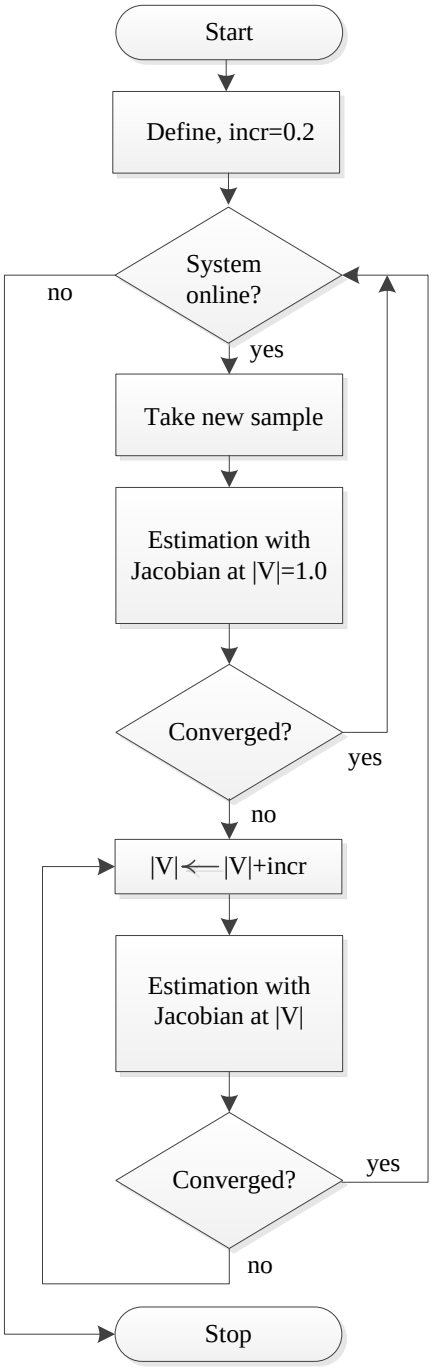

Figure 4.7: A simplified block diagram of the proposed multi-Jacobian method.

4.4 Practical Importance of the Multi-Jacobian Method

In reality, the power system can be very large including thousands of buses with two states at each bus. For example, if a system contains 5000 buses, the number of states will be 9999, and the number of measurements can be more than 12000. With this large size, the WLS estimator may take as much as $4 - 8$ seconds. This may be sufficient for SCADA rate of data collection. But, the PMU rate is much high, and the slow process may not work at all. As a result, the dishonest method should get preference over the WLS method.

Though the dishonest method may perform fast, it may fail to converge at some cases. It is very difficult to find the special combinations of the states for which it fails. However, no solution is found in the literature to make it work in case of failures. The proposed multi-Jacobian method gives a direction for that. This makes the existing estimator much robust to system changes. The utilities should feel much confident in using the dishonest method with the extension.

4.5 Illustrative Examples of Failure

It is already shown that a higher slope ensures convergence of the dishonest method. Though it is easy to derive the expression for the range of convergence for a single state, it is difficult for the multi-state case. However, the importance of the proposed method for the multi-state case can be realized through simulation.

To show the case of failure, two Jacobians (at $|V_i| = 1.0$ and at $|V_i| = 1.2$) are applied on IEEE 68, and IEEE 118-bus test systems operating under disturbances. The 68-bus system has 16 machines with 83 transmission lines. The details of the systems can be found in [58] (68-bus), and [57] (118-bus). The states for the failed

cases are shown in Table 9.1, and 9.2 of the Appendix. As the angles stay very close to zero, the nominal Jacobian is taken at the highest slopes at $\theta = 0$. The voltage magnitudes vary a lot under the specified case.

The norms of the residues over the iterations of estimation are shown in Figs. 4.8, and 4.9. For both cases, it can be seen that the norms decrease in the beginning for both Jacobians. Then the nominal Jacobian ($|V_i| = 1.0$) starts a gradual increase after around ten iterations. It continues increasing and the process explodes eventually. On the other hand, the Jacobian with $|V_i| = 1.2$ converges with the iterations. These two examples clarify the importance of the proposed method.

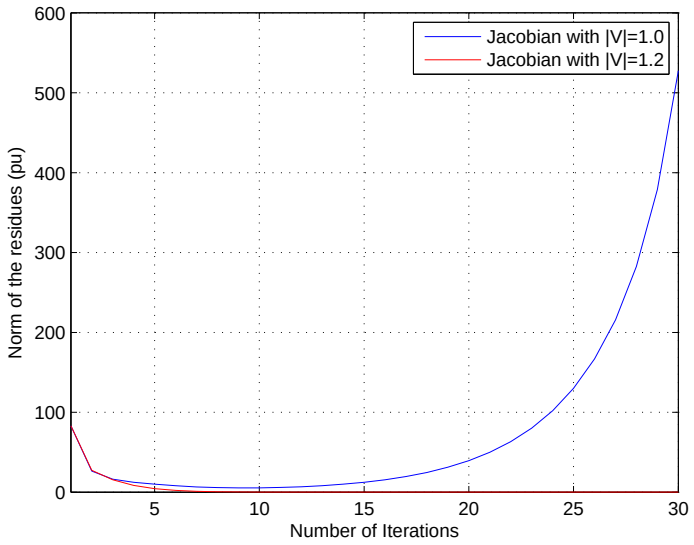

Figure 4.8: The trends of convergence for a special case of 68-bus system where the nominal Jacobian fails to converge and the Jacobian at |V|=1.2 converges successfully.

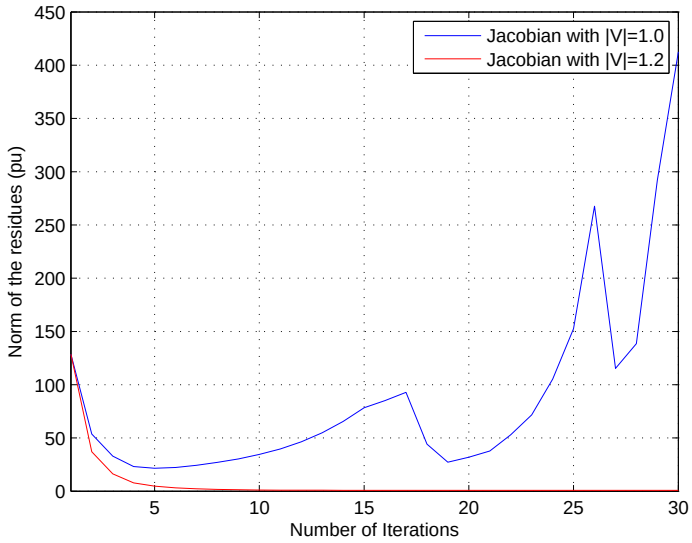

Figure 4.9: The trends of convergence for a special case of 118-bus system where the nominal Jacobian fails to converge and the Jacobian at |V|=1.2 converges successfully.

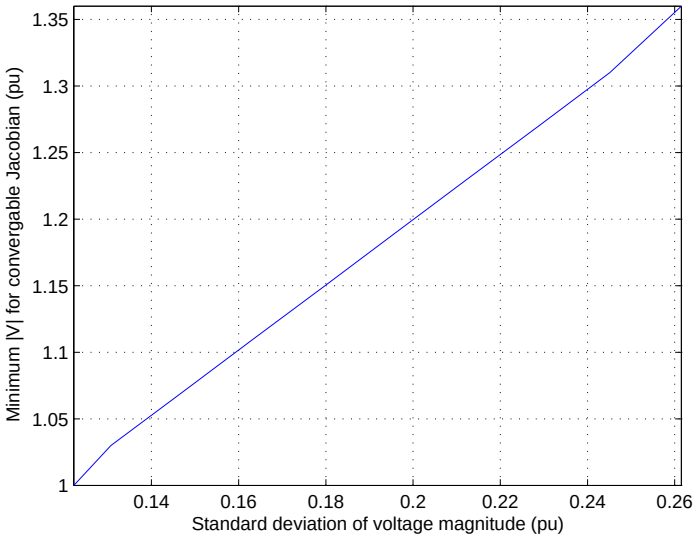

Figure 4.10: The minimum voltage magnitude to calculate the Jacobian that is required for convergence of 68-bus system. The Jacobian needs to be calculated with higher slope with increasing standard deviation of the states.

An important point to be noted that the norm can increase after a low value for $|V_i| = 1.0$. This means that a short distance with the starting value of the states, $\mathbf{x_0}$ does not ensure convergence. The convergence depends on the closeness of the point of Jacobian and the point of operation.

As analyzed in Section 4.2, the Jacobian should require higher $|V_i|$ for higher variations of the voltage magnitudes of the buses. The minimum required $|V_i|$s are shown for different standard deviations of the magnitudes in Fig. 4.10. It can be seen that the nominal Jacobian fails for a standard deviation more than 0.1222. It is also observable that the minimum $|V_i|$ keeps a linear relation with the standard deviation of the states.

It is important to remember that, the existing method with the Jacobian calculated at $|V_i| = 1.0$ that worked upto a variance of 0.1222 is still a very strong tool. Because, under normal operating conditions, the variance usually does not exceed 0.05. Even with 10-20% load change of the 68-bus system, the magnitudes does not change much and they can be easily estimated. However, with very low probability, the states may reach some values that may not be possible to estimate using $|V_i| = 1.0$. Two such cases are shown in the Appendix. In one study, one out of 20000 samples failed to converge with $|V_i| = 1.0$. Though the probability of such cases is low, it can be crucial as the states may undergo very high change during that time. Under these failed cases, the safer choice will be to calculate the Jacobian with a higher value of $|V_i|$, not with a lower one.

The single and the multi-Jacobian methods are compared in Table 4.1. The range of convergence refers to the maximum variation of the states with which the method can converge. The speed of convergence is the inverse of the time required to converge. The computational requirement is shown for the case where both methods converge. The storage requirement denotes the memory needed for saving the **M**-

matrices for different $|V_i|$.

Table 4.1: Comparison of the Single and the Multi-Jacobian Methods

Qualities	**Single Jacobian method**	**Multi-Jacobian method**
Range of Convergence	Limited	High and not limited
Speed of Convergence	Fast Fast	Equal/slower than the single-Jacobian method
Application	All cases, except very high variations of voltages	All cases
Computation requirement	Low	High
Storage requirement	Low	High

4.6 Implementation on a GPU

The reordering of the steps of WLS estimator in Section 2.4.2 comes as a result of the arrangement of the processors. In GPU computing, a set of threads are called to execute a kernel. The threads belong to one or more blocks. So, the number of threads needs to match the number of tiny operations. In the first step, there will be an equal number of measurements and power flow equations. Therefore, they can be combined in a single equation and can be run in a single thread. If there are m_s measurements, the required number of threads is m_s. As the GPU can handle a maximum of 1024 threads per block (for K20c), it will require only one block if $m_s \leq 1024$ [59].

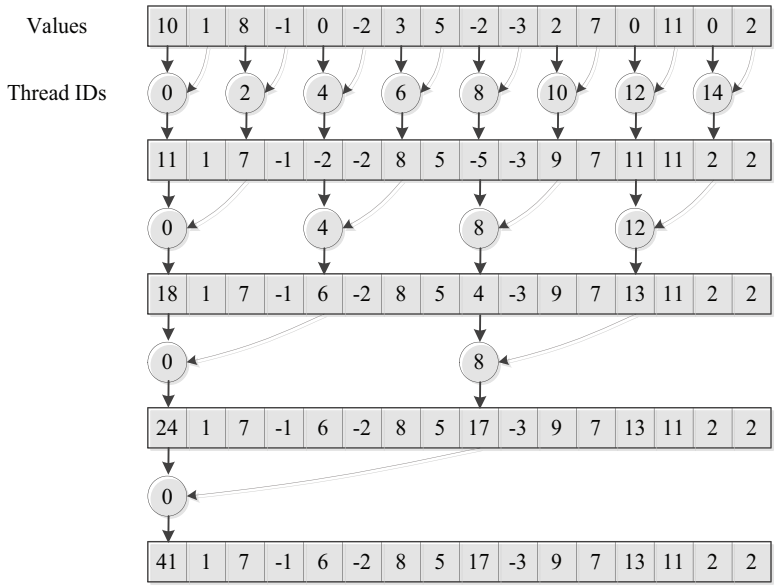

Figure 4.11: Parallel addition of 16 numbers.

Step *ii* is a matrix-vector multiplication that can be divided into two parts - multiplication by rows to columns, and addition by rows. Though it can be implemented in different ways, the best possible method is to assign one processor for every multiplication. As the size of $\mathbf{M}$ is $(2N-1) \times m_s$, it can be separated by blocks and threads. Each block will be responsible for each row and the threads of that block will take care of the columns of that row. However, after completing the multiplications, the columns need to be added. Adding m_s elements of a row vector can be made parallel according to the method shown in Figure 4.11 [60]. Instead of using the full method, the first part can be implemented to get an acceptable speedup. This simple method can add up to 2^n elements in the required time of n addition.

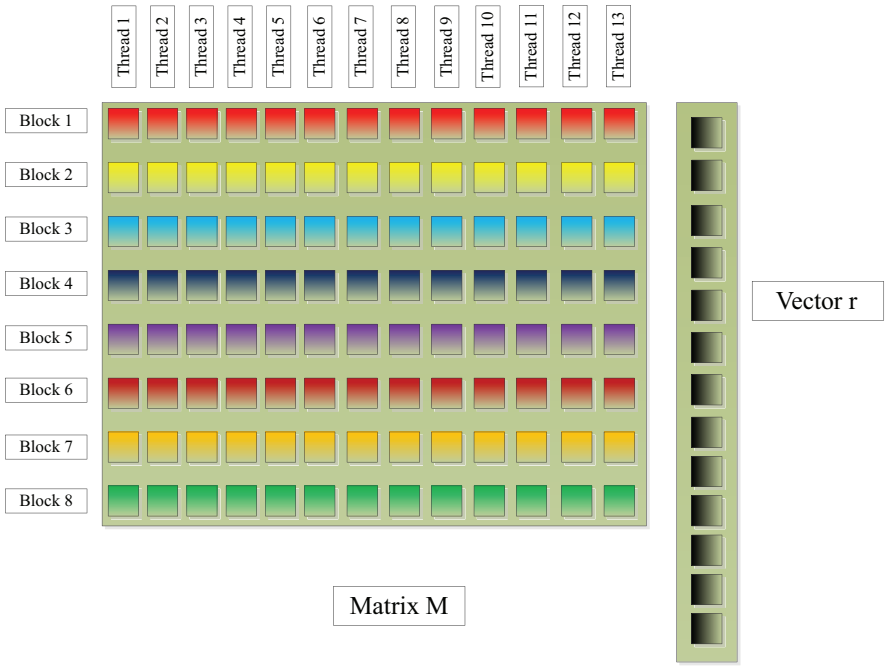

Figure 4.12: Parallel multiplication of a matrix and a vector.

Step *iii* is a simple vector-vector addition. It will require $2N-1$ threads that can be accommodated in one block for a system of 511 buses.

4.7 Simulation Results

To test the proposed dishonest method based state estimation, it is implemented on the IEEE 16-machine 68-bus New York -New England power system with 83 transmission lines. To analyze it's timing profile, it is also implemented on IEEE 118-bus test system. Measurement errors are added artificially which varies from 0.25-4% of the original value. The measurements are taken for three seconds under a sudden change of load at bus 8 at a rate of 30 samples per second that is the typical sampling rate of the Phasor Measurement Units (PMUs). The data under fault ensures a big variability to test the dishonest method.

For estimating the states from the measurements, an NVIDIA Tesla K20c GPU card with compute capability 3.5 is used. Based on the three steps, three different kernels are written which use different number of blocks and threads. One of the major advantages of GPU is that it does not require any *extra* time to launch and finish a new kernel. So, the kernels can be executed sequentially without any delay.

As analyzed in Section 4.2, it is safe to take the slopes at high values of the state variables. Under normal operations, the voltage magnitudes do not deviate more than ±5% of 1.0 pu. With large load changes, it may vary by ±20%. For simulation, three sets of magnitudes are chosen to build the Jacobian matrix, at 0.9 pu, 1.0 pu, and 1.2 pu. For all cases, the angle is set according to Section 4.2.4.

4.7.1 Accuracy

Accuracy is the most important characteristic of an estimator. It is well known that the WLS estimator is the most accurate estimator for Gaussian noise [61]. As expected from the analysis of the dishonest method, it also converges to the solution of the WLS estimator. But, the number of iterations required for the dishonest method to achieve the same accuracy is more than the WLS estimator. On the other hand, it takes a very short time for each iteration. If it takes less time for achieving the same accuracy, then it is meaningful to use the dishonest method.

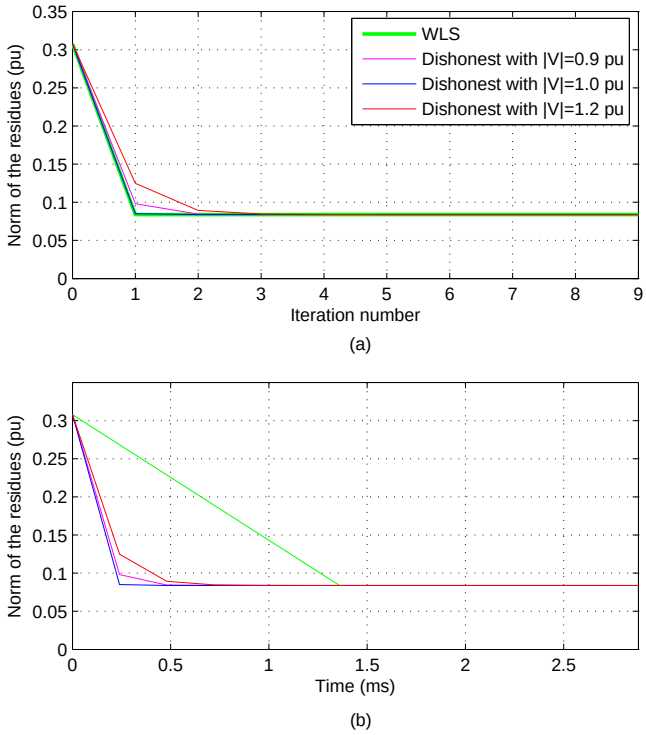

Figure 4.13: The trend of norms of the estimations with (a) iterations, (b) time. The required time is taken from the serial implementation. It is very low and incomparable in parallel programming with CUDA.

The accuracy achieved by the WLS estimator and the dishonest one over iterations are shown in Fig. 4.13. It can be seen that the dishonest one with the Jacobian calculated at the maximum points (|V|=1.2) takes longer time than the one calculated at a nominal value. It takes around four iterations to reach the accuracy of the WLS method while with |V|=1.0, it takes only two. In serial implementation, an iteration of WLS method takes around six times more time than that of dishonest method. So, the dishonest method saves around 33% time. However, it is way too effective in parallel implementation.

The estimated as well as the actual voltage angle and magnitude for bus 8 are shown in Fig. 4.14. The simulation results are taken with one iteration of WLS, and

four iterations of the dishonest method. It can be seen that the results are very close, and they can be considered equivalent.

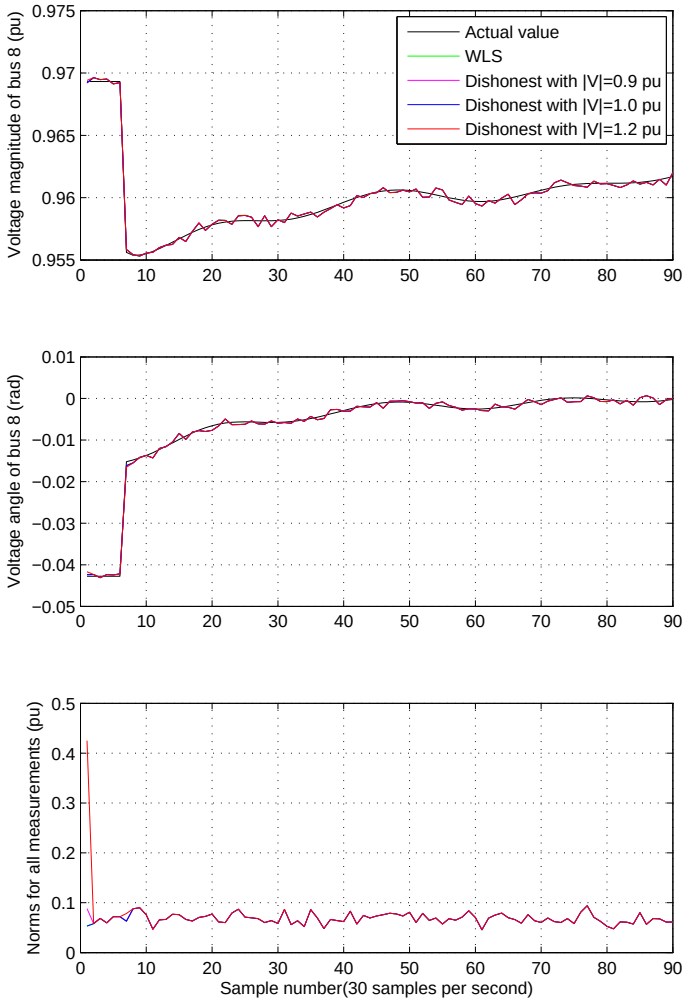

Figure 4.14: The accuracy of the estimation for different methods. The first two parts show the accuracy for voltage magnitude and angle of bus 8. The last part shows the overall norm of the residues for all measurements.

Noise is not the single factor; the accuracy depends on some other factors as well. Among the other factors, measurement collection rate and the number of

iterations are connected together. If the measurements are collected at a slower rate, the states change by a greater amount. So, starting from the previous estimation, it takes longer time to reach the new estimation.

However, it does not create any problem. Assume that the measurement is collected at a rate of n_m samples per second, and the estimator runs at n_e times per second with k iterations per time. Definitely, $n_e \geq n_m$. Now, if the rate is decreased by a factor of d, i.e., n_m/d, the number of iterations per sample can be increased by a factor of d. If the estimator can achieve the desired accuracy in that iterations, it will not create any problem.

In practice, the rate of collecting measurements is way too slow than the time required with dishonest method. Even with the fastest rate of the PMUs, the data is collected at every 8.33 ms where the estimator runs at 200 μs with three iterations. Even if the estimator runs for ten iterations, it will not take more than 670 μs.

The relation between the sampling rate of the measurements and the accuracy for different number of measurements for a 68-bus system [58] is shown in Fig. 4.15. It can be seen that the accuracy increases with the increase of sampling rate as well as with the number of iterations per sample. It also shows that seven iterations per sample give a good accuracy for any rate of collection over three samples per second. As the sampling rate is known, the number of iterations can be settled accordingly.

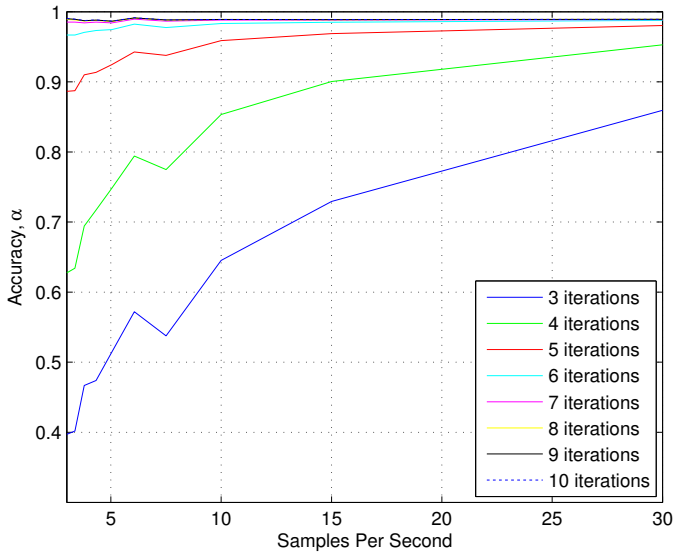

Figure 4.15: Accuracy of the dishonest Gauss Newton method compared to the honest Gauss Newton method for different rate of data of 68-bus system.

4.7.2 Impact of Noise

In power systems, the noise is assumed to be Gaussian. It can play a big role on the accuracy. Not only the level of noise, rather the pattern of noise can also change the accuracy. The same values of noise redistributed to different measurements can cause different accuracy.

The impact of noise on the accuracy is shown in Fig 4.16 for six different combinations of the noise values. The level of noise is defined as the mean noise to measurement ratio. The simulation shows that the accuracy is a bit low (around 98%) for low level of noise. However, it gets upto a certain level in between 99 to 99.5%. It does not give 100% accuracy at any time.

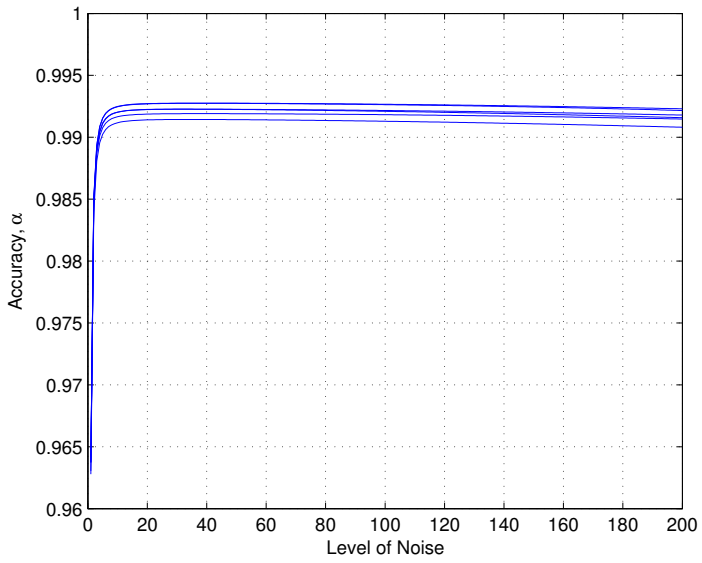

Figure 4.16: The accuracy of the estimator under different level of noise.

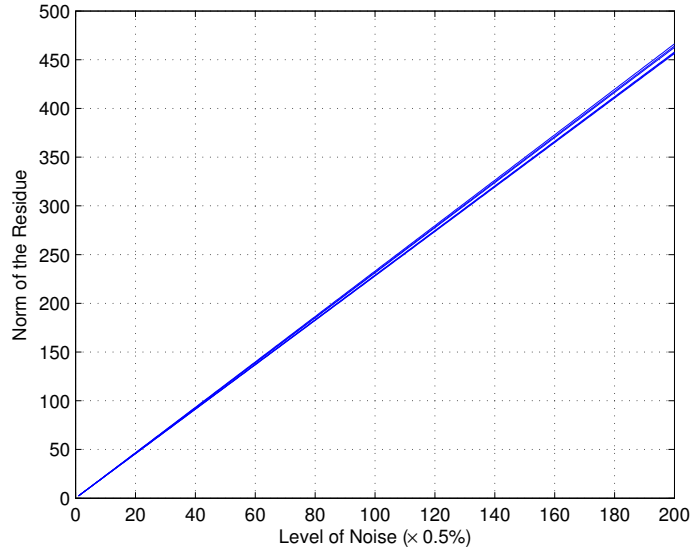

Figure 4.17: The norm of the residue of the estimated values under different level of noise.

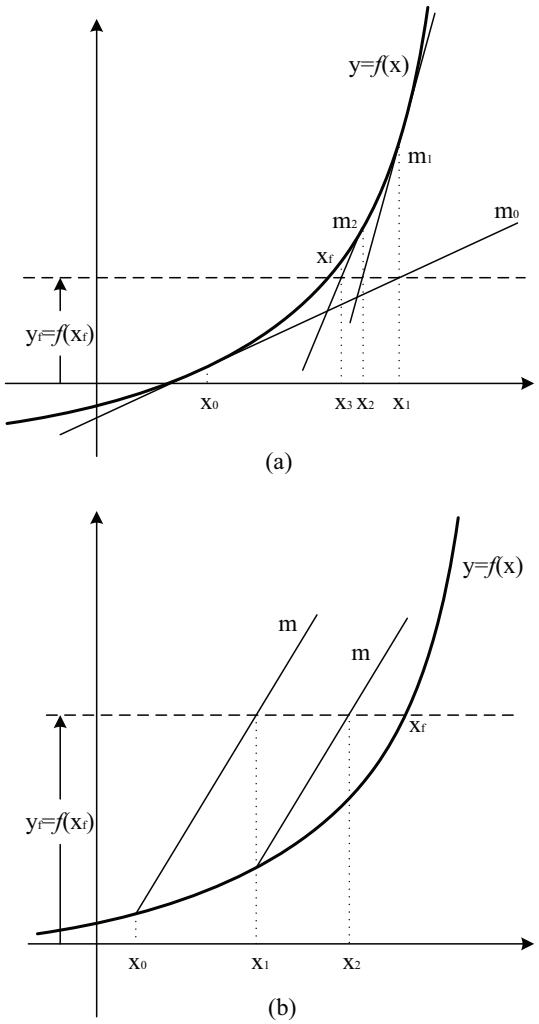

Figure 4.1: The working principle of Gauss Newton method, (a) honest, and (b) dishonest.

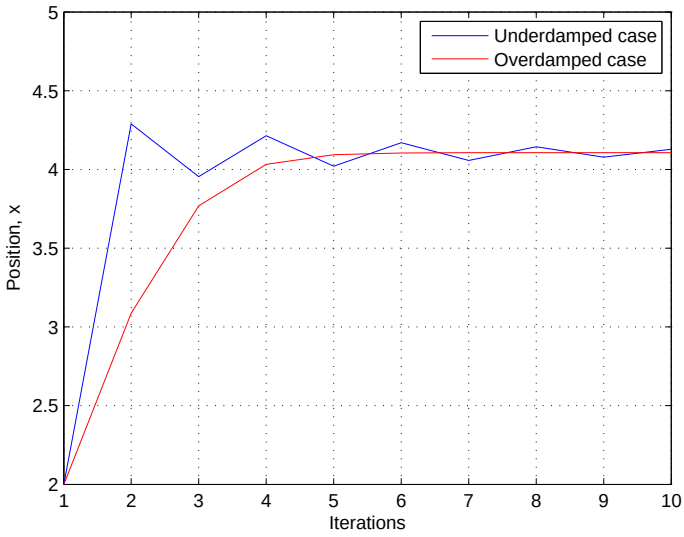

Figure 4.2: Two ways of convergence of dishonest Gauss Newton method, underdamped and overdamped case.

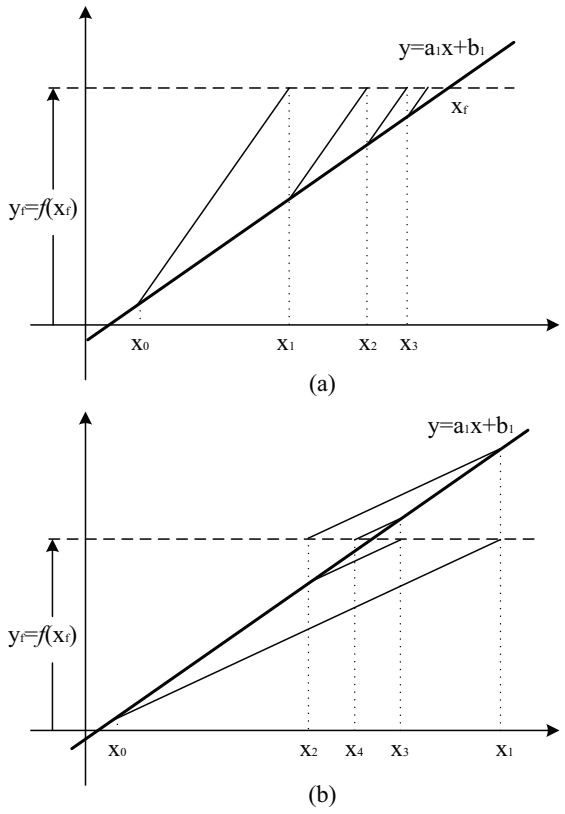

Figure 4.3: Two cases of convergence for a linear function when the objective value is higher than the starting value, (a) overdamped, (b) underdamped.

The effect of noise on the norm of the residue is shown in Fig. 4.17. This can be referred as the absolute accuracy. It is completely linear which means that the residue increases linearly with the increase of the level of noise. It is also shown for six different noises.

4.7.3 Computation Time

The main advantage of the dishonest method is that it is very suitable for the GPU architecture. From the parallel programming on GPU platform, it is found that on an average over 50 trials, the required time for each sample of IEEE 118-bus system with four iterations is about $282\mu s$. To the best of our knowledge, it is the lowest time ever reported in any research work on parallel and distributed state estimation [2, 29, 62].

On the other hand, for parallel implementation, the WLS estimator takes around $2.77ms$ with one iteration that is around 42 times of one iteration of the dishonest method. Due to a large portion of non-parallelizable parts such as Cholesky decomposition, back-substitution or Gauss-Jordan elimination, it cannot speedup much on a GPU.

The reported time of $282\mu s$ includes the transfer time of measurements and estimation results between the CPU and the GPU. To find the communication time, it is run for different number of iterations and the required times are plotted in Fig. 4.18. For 118-bus system, the communication time is found to be around $26\mu s$, and each iteration takes around $64\mu s$.

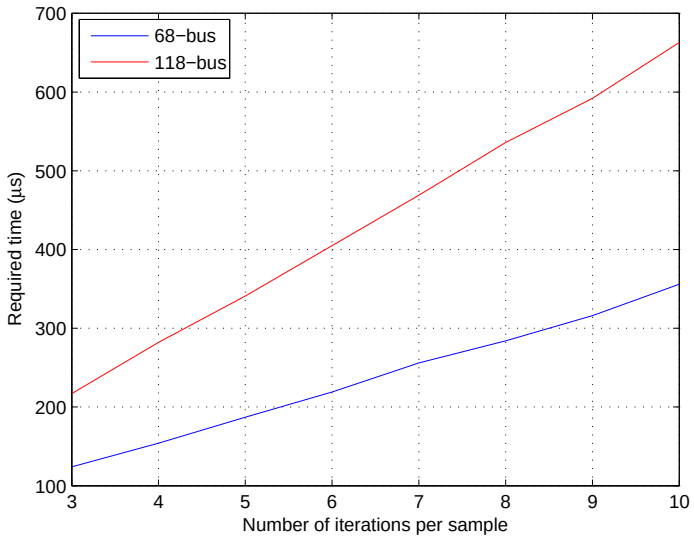

Figure 4.18: The required time for different number of iterations for IEEE 68-bus and 118-bus system.

4.8 Estimation for Very Large Systems

In reality power systems can be very large consisting of thousands of buses. A single GPU is not enough for estimating the states of a thousand bus system. However, it is possible to estimate the required time for system with any size.

Though it is expected that the blocks and the threads of a GPU will run simultaneously, it is not possible in practice. Like other computation devices, GPU has its own limitations.

Every block of a GPU runs a maximum number of threads at a time. In case of the k20 series, 32 threads run at a time in one block. These 32 threads are called a warp. On the other hand, the blocks are run on the streaming multiprocessors (SMs). These SMs are physical entity, and they are limited as well. Each SM can take care of more than one block, but their scheduling can differ.

For this analysis, let the computation unit consists of N_G GPUs which are connected through a suitable protocol like MPI. The maximum number of threads and blocks that can run simultaneously on a single GPU be N_t and N_b, respectively. A block can hold a maximum of N_{max} threads. The times required for each addition/subtraction and multiplication are t_a, and t_m. The number of states and measurements are $n(= 2N - 1)$, and m_s.

The first step has m_s calculations of $h(\mathbf{x})$, and m_s subtractions. As it is dependent on the nature of $h(\mathbf{x})$, let us take the time for $h(\mathbf{x})$ is t_h. For m_s additions, it will require $ceil(m_s/N_{max})$ blocks which will be provided by N_G GPUs. So, the total time for step i is,

$$t_1 = max(t_h) + ceil(\frac{ceil(\frac{m_s}{N_{max}})}{N_b N_G}) \times ceil(\frac{N_{max}}{N_t}) \times t_a$$

It is highly likely that there will be enough blocks to handle this addition operation and the expression can be simplified as,

$$t_1 = max(t_h) + ceil(\frac{N_{max}}{N_t}) \times t_a \tag{4.8}$$

The most time consuming part is step ii. There are n rows and m_s columns in matrix $\mathbf{M}$. They can be distributed differently. As for regular cases $m_s >> N_{max}$ and $n > N_b \times N_G$, a good choice is to assign one block to each row and one thread to each column. So, each block will require $ceil(m_s/N_{max})$ stages to complete the corresponding row. They will be called in $ceil(n/N_b N_G)$ stages to complete all rows. Each stage will take the time of N_{max}/N_t multiplications.

After completing the multiplication of one stage, there will be N_{max} additions before taking the new values for the next stage. This will take the time of $ceil(log_2 N_{max})$ addition operations. So, the total time required to complete $N_G \times N_b$ rows is,

$$t_{21} \approx (\frac{N_{max}}{N_t} \times t_m + ceil(log_2 N_{max}) \times t_a) \times ceil(\frac{m_s}{N_{max}})$$

The approximation comes from the last stage of the row which will have $remainder(m_s/N_{max})$ additions. As there are $ceil(n/N_b N_G)$ stages to complete all rows, the total time required to complete step ii is,

$$t_2 = ceil(\frac{n}{N_b \times N_G}) \times t_{21} \tag{4.9}$$

Step ii produces $ceil(m_s/N_{max})$ columns for each row which need to be added with $\mathbf{x}$. In step iii, they are added together to find the new $\mathbf{x}$. Using the same assumption of step i, the required time for step iii can be calculated as,

$$t_3 = ceil(\frac{N_{max}}{N_t}) \times (ceil(\frac{m_s}{N_{max}}) + 1) \times t_a \tag{4.10}$$

Now, all of them can be added together with the communication time to estimate the required time t to compute one iteration of estimation for a very large system,

$$t = t_1 + t_2 + t_3 + t_{comm} \tag{4.11}$$

It should be remembered that the launch or execution time of the threads or blocks are not always equal and there is randomness. So, a random number should be added with each term of (4.11). Moreover, each block or thread should wait for the slowest ones to complete their jobs.

There are other factors like the number of registers and size of shared memories that limits the exploitation of the processors. They are excluded from the calculations for simplicity.

4.9 Summary

In this chapter, the convergence and the speedup of the dishonest Gauss Newton method have been investigated. It is shown that the method can ensure convergence if the Jacobian contains the steepest slopes within the possible range of operation. With this analysis, a multi-Jacobian method is proposed and it is shown that it can converge at some cases where the traditional estimator fails.

With the parallel implementation of the dishonest method on a GPU, it is also shown that it yields a very high speedup of around 42 times compared to the WLS method. With this rate, every sample of PMU data can be estimated in real-time and they can be used for better understanding of the status of the system. Though PMU is not a mature technology to date and the outputs are used only for post-mortem analysis, it is expected to take some real-time roles in the coming smart grid technology. A fast estimator will have its impact in each real-time applications of the PMU.

For large systems with thousands of buses, a cluster of GPUs is recommended. The implementation strategy and the required time for large systems are provided. From the analysis, it can be inferred that the implementation on multiple GPUs en-

ables a fast state estimator while ensuring sufficient accuracy. Further investigations are needed on the implementation of multiple GPU.

Chapter 5

Distributed Static Estimation Using Cellular Computational Network

5.1 Introduction

In order to solve the problem of scalability of any networked system, a new framework referred as Cellular Computational Network (CCN) is proposed in [6]. In this framework, every cell will include a moderate powerful computational unit. These units will communicate with each other to exchange information related to their tasks.

This chapter presents a layer-based static estimator using cellular computational network. It is a preliminary work on developing the final distributed state estimator. Due to its sequential operations, the process cannot be completed in a very short time. However, the distributed nature of the estimator can keep the privacy of the deregulated market.

5.1.1 Power System Mapping Using CCN

The idea of the cellular architecture can be summarized as follows,

- The system reference bus does not require any computation as the voltage magnitude and the voltage angle are defined ($V_{ref} = 1$ and $\theta_{ref} = 0$). The powers are measured with respect to this bus. This single bus forms the first layer.

- Buses connected to the system reference bus form the second layer. They take that as their own reference bus, and they do not need to scale the powers. Each bus runs their own estimation using the measurement taken at t, and calculates voltage magnitude and angle.

- Every bus connected to the second layer forms the third layer. They collect the estimated voltage magnitudes and angles of the connected reference bus from the second layer and scale the local power measurements. Then they run the estimation for the measurements taken at t. After completing estimation, the estimated magnitudes and angles are adjusted for the system reference bus. At the same time, layer 2 runs their own estimation using the measurements taken at $t + 1$.

- Buses connected with the third layer form the fourth layer. They take the buses of third layer as their own references and run the estimation. By the time the fourth layer is working on the measurements taken at t, the third layer works on the measurements taken at $t + 1$, and the second layer works on $t + 2$.

In this way, all the buses of all layers work at the same time with different measurement sets taken at different time slots.

5.2 Test Systems and Results

In order to verify the feasibility of the cell based architecture, IEEE 68 bus test system is taken as the model network. In Figure 3.2, the 68 bus, 16 generator system is shown with its corresponding layers of computational cells.

5.2.1 Sequence of Layers

According to the proposed architecture, each bus is equipped with a cell capable of estimating states. As bus 1 is the reference bus for the whole system, it is the single element of the first layer, and the states are known for it. The second layer includes those buses which are connected with the reference bus. The sets of buses in different layers are given below,

- Layer 1: {1}
- Layer 2: {2 47 31 30 27}
- Layer 3: {3 17 25 53 48 62 38 32 9 26}
- Layer 4: {16 29 28 60 40 46 33 63 36 8 4 18}
- Layer 5: {19 21 24 61 14 15 5 7 37 64 34 49 41}
- Layer 6: {23 22 56 20 13 6 65 43 35 52 42 66}
- Layer 7: {59 58 57 10 11 12 54 44 45 50 68 67}
- Layer 8: {55 39 51}

The execution of the layers are shown in Figure 5.1. In the starting of the process, only the cells of layer 2 execute estimation with the most recent data as they

are directly connected with the system reference bus. In the second time slot, layer 3 takes the estimates from layer 2 and measurements of time slot t. At the same time, the cells of layer 2 do not wait, rather they keep estimating their states using measurements taken at time slot $t+1$. At the next time slot, layer 3 starts running with the measurements taken at t. It runs behind of layer 2 by two slots. Over time all the layers get added to the execution. When the system is completely on, all the layers run simultaneously with measurements taken at different time slots.

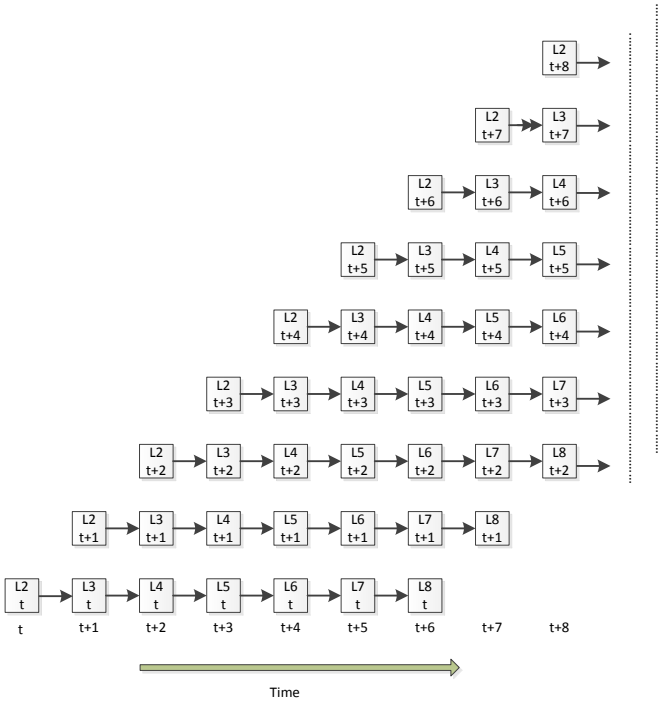

Figure 5.1: The starting and running of layer based estimation. Here, the labels in side the boxes have two parts. The upper parts denote the layer numbers starting with L, the lower parts show the time slots of the measurements for which the layer is running. The horizontal line at the bottom shows the original time slots. For example, at time slot $t+8$, the cells of layer 2 is running with the measurements taken at $t+8$, while the cells of layer 3 is running with the measurements taken at $t+7$.

5.2.2 Accuracy

Accuracy is an important property of an estimator. Like all other estimators, the accuracy of the cell based estimator depends largely on the measurement noise. To keep consistency with practical cases, the measurement noise is taken within $\pm 6\%$ with zero mean in these experiments.

The trend of accuracy of the estimator is shown in Figure 5.2. The states are organized based on their layers. The buses of layer 2 are taken first, then the buses of layer 3 and so on. It can be observed that the estimated state is getting much deviation as it is going far from the reference bus. The reason behind that is the flow of estimation error. As the estimations of the buses of layer 3 depend on the estimated values of the buses of layer 2, the errors of layer 2 get added with the estimated values of layer 3.

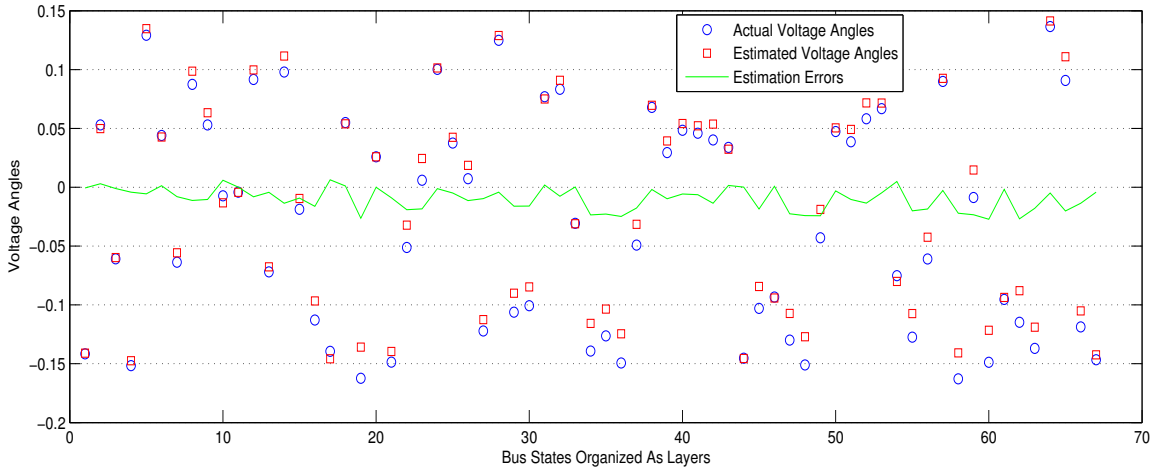

Figure 5.2: With a noise range of $\pm 6\%$, the actual and estimated voltage angles. The states are rearranged based on layers. The buses of layer 2 are taken on the left most side. Then layer $3, 4, 5, 6, 7$, and 8 are taken. It can be noticed that the layers far from the system reference bus get much deviation from the actual angles.

5.2.3 Flow of Errors

To solve the problem of the flow of error, some terms need to be defined first. The layers can be considered as *generations*. The reference of a bus is referred as the first *predecessor* for that bus. The reference of the reference bus can be considered as the second predecessor. The gap between two buses are known as *generation gap*. For example, if bus 3 is the reference for bus 4, bus 3 is the predecessor for bus 4. If bus 2 is the reference for bus 3, bus 2 is the second predecessor for bus 4. The generation gap between bus 2 and bus 4 is 2.

The problem of the flow of estimation error does not make big residue for those connected buses who have their common predecessor in near generation. It means that, if two buses have the same root in a near predecessor, the relative estimation will be much accurate. For example, bus 23 and 22 are connected together. They have the common root at bus 16. The generation gap is 2. So, the power flow between bus 22 and bus 23 will show less residue. But for the cells of the buses 11 and 12, they have a distant common root. So the power flow will show a big residue.

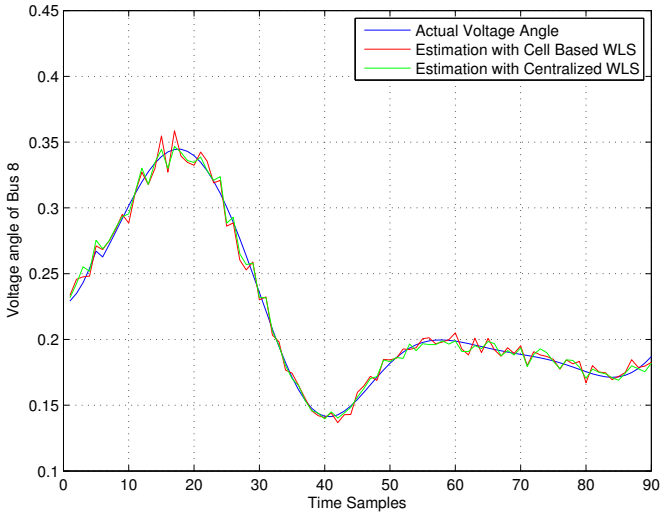

Figure 5.3: Comparison of the abilities of the cellular estimator with the centralized one to track the states under an unstable condition. The voltage angle of bus 8 is changing rapidly with time along with other bus voltages. Both estimators are following the actual value with reasonable accuracy.

In order to verify the ability of the new architecture, the system is estimated under an unstable situation created in RSCAD. The estimated voltage angle of one bus is shown in Figure 5.3. It can be seen that the new architecture is able to follow the unstable situation. Though the accuracy is not as good as the centralized estimator, it is still good enough to meet the requirement of speed.

5.2.4 Detection of Bad Data

One of the major aspects of any estimator is the ability to detect and identify the bad data. WLS estimator uses the L_2-norm of the measurement residues to detect the presence of the bad data. This norm is compared with a predefined threshold value. However, in this experimental setup, for a realization of noise, the norm of the residue for the centralized estimation comes 0.8311 where in the cellular architecture, it becomes 3.9247. This shows that the proposed architecture is not as accurate as

the centralized estimation. The speed is achieved with the cost of accuracy.

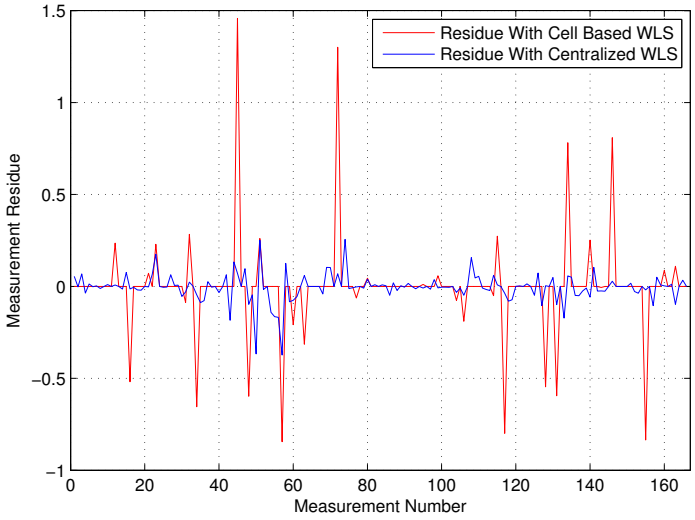

Figure 5.4: Comparison of the residues of the cellular estimator with the centralized one. As can be seen, the residues are significantly large than for the cellular architecture for some specific measurements. This is caused by the generations.

5.3 Summary

This is an intermediate stage of a study on the application of cellular computational network in static power system state estimation. The advantages and challenges of the new CCN based WLS estimator is presented in this chapter. Using offline simulation results, important characteristics of the estimator like speed, accuracy, observability etc. are analysed.

As the accuracy of the estimator was not well enough, it needs some modifications. Moreover, it can be understood that the addition of the layers will increase the computation time. It can be kept constant if the cells can run simultaneously. It is described in the next chapter.

Chapter 6

Semi-Dynamic Hybrid State Estimator Using CCN and Heuristic Methods

6.1 Introduction

The privacy of the data in the deregulated market is an important factor for energy market. It affects the policies of the utilities directly. With the centralized estimator, the system-wide information needs to be gathered in a center that may go against the interest of the energy market participants. The problem is partially solved in the previous chapter with some concerns about the sequential operations. In this chapter, a fully distributable method is developed that can keep the privacy of the data and run a parallel cellular estimation.

The static estimator requires a sequential flow of layers due to the dependency on the reference bus. It increases the overall computation time with the addition of new layers. To avoid the problem of the estimation of the reference bus, the previous

estimation can be used for normalization. As the states do not change much with a single time step, the normalizations of the powers do not deviate much from the actual value. This makes a semi-dynamic estimator which enables the simultaneous execution of computation cells.

To improve the accuracy of the semi-dynamic estimator, different heuristic methods are tested. Out of them, genetic algorithms show a better performance in terms of accuracy and time. Several heuristic optimization methods including Particle Swarm Optimization (PSO) have been studied for power system state estimation and they perform quite well for small systems. However, in case of larger systems with hundreds of states, they suffer from the *curse of dimensionality*. In this chapter, two important variants of PSO, Comprehensive Learning PSO, and Orthogonal Learning PSO, which are specialized for multimodal high dimensional systems, are implemented and found to provide inaccurate estimations over time. To overcome this problem, a hybrid state estimator which consists of the CCN based semi-dynamic estimator and the Genetic Algorithm (GA) is developed. Through simulation, the proposed CCN-GA is shown to outperform all other direct and hybrid methods in terms of accuracy and time.

6.2 Structure of the Estimator

Due to the dependency to the previous estimation result, the first estimation requires an independent development process which is taken from [10]. It is a multi-layer static estimation process. In the running mode, the local estimation gives the relative states and these are saved separately. Using these differences, the cells exchange and update their results until these converge (Fig. 7.3). Then, these converged states are used to normalize the measurements and the local estimator runs

again. The process repeats for a fixed number of iterations, k_{max}. At the end of k_{max}, it takes the next measurement set for estimation.

6.2.1 Value of Hybrid Methods

The proposed hybrid method based on CCN-GA is a fully distributed solution for power system state estimation. This maintains the privacy of the data, and takes moderate time that can be used for traditional SCADA rate.

A major practical importance of the heuristic methods is that it does not require any differentiable function. Though the WLS estimator is very accurate, being a gradient based method, there are some specific situations like the ill-conditions, where it fails. Under this case, the gradient becomes zero and the method cannot proceed. On the other hand, the heuristic parts of the hybrid methods do not require any calculation of gradient, and they can perform pretty well for these cases.

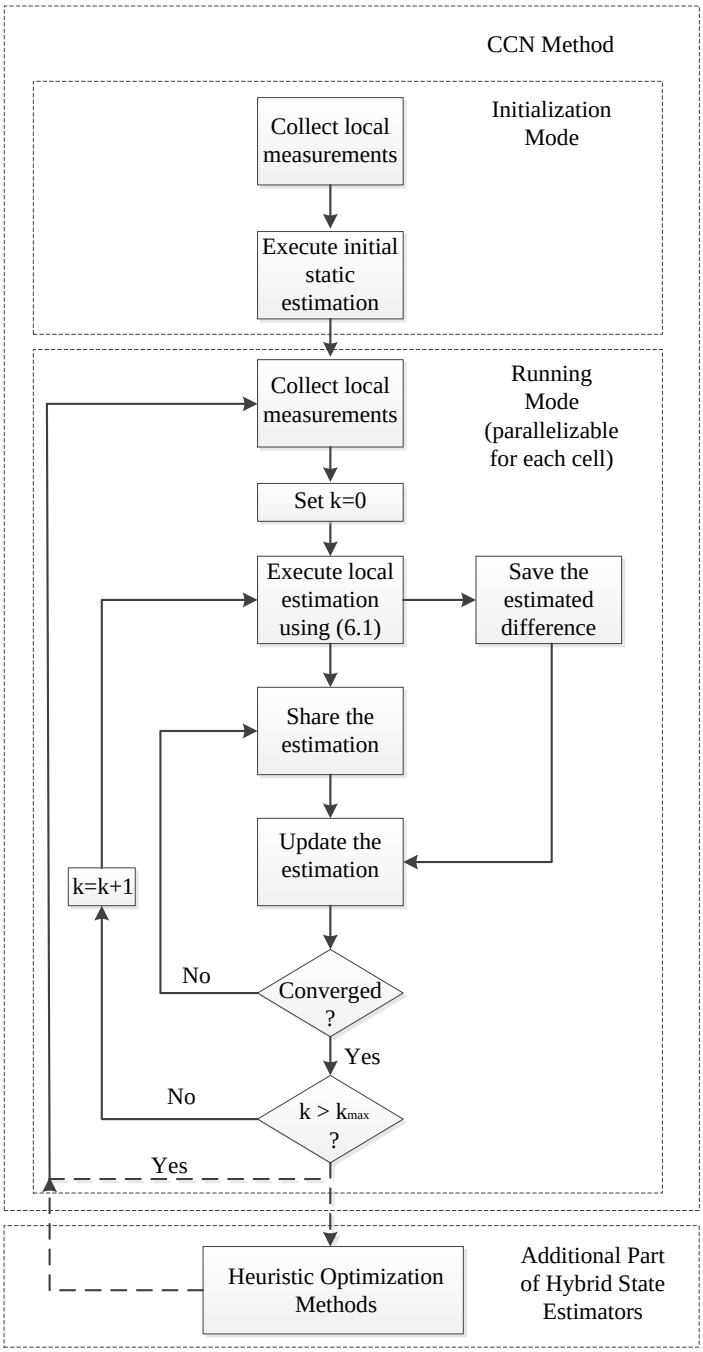

Figure 6.1: The complete estimation process of the hybrid estimator using CCN. The upper box shows the flowchart of CCN in details. The output of CCN is fed to the Genetic Algorithm or the PSOs.

6.2.2 Estimation of a Single Cell

If a cell includes any measurement other than the power flow, the estimation will require a complete iterative WLS process. Otherwise, the estimation can be done directly. Though the whole network can include loops, a single cell that is responsible for only one bus includes only a star network without any loop as shown in Fig. 6.2. For this single star network, Lemma 1 gives an insight about the WLS estimation.

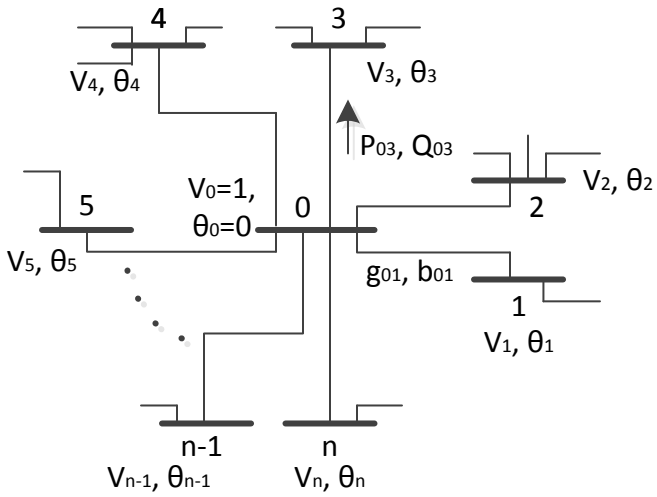

Figure 6.2: Estimation of a single cell. It forms a star network where no bus is connected with other except the center one.

Lemma 1. *For any star network with unique power flow measurements and Gaussian error, the WLS estimator yields an estimated value with zero residues given the network is observable.*

Proof. The WLS estimator is the most efficient estimator for Gaussian error and it gives the minimum L_2-norm of the residues [61] for overdetermined systems. No solution can give less residue than the solution given by WLS. For a square full-rank

system, there exists an exact solution and the residue becomes zero. Therefore, for a square system, WLS yields zero residue.

Let, n buses are connected with bus 0 whose state is to be estimated. For each connected bus, there can be a real and a reactive power flow measurement, P_{0j}, and Q_{0j}. If any one of them is missing, the network becomes unobservable. Therefore, according to the condition of observability of the lemma, there will be a total of $2n$ measurements (as the measurements are unique). As the star network has a total of $(n+1)$ buses, and there is no voltage magnitude measurement, there will be $(2(n+1)-2) = 2n$ number of states. So, the number of states becomes equal to the number of measurements that makes it a square system. As a result, the WLS estimator will yield zero residue. □

Under this situation of an equal number of equations and state variables, the states can be calculated directly from the equations and no help is needed from the WLS estimator. For example, for each transmission line, the voltage angle and the magnitude can be calculated by the following equations which can be easily derived from (2.6), and (2.7),

$$\begin{aligned} \hat{\theta_j} &= tan^{-1}(\frac{b_{0j}P_{0j} + g_{0j}Q_{0j}}{g_{0j}^2 + b_{0j}^2 - g_{0j}P_{0j} + b_{0j}Q_{0j}}) \\ \hat{V_j} &= \frac{b_{0j}P_{0j} + g_{0j}Q_{0j}}{(g_{0j}^2 + b_{0j}^2)sin\hat{\theta_j}} \end{aligned} \tag{6.1}$$

Where, θ_j and V_j are the relative angle and magnitude of bus j with respect to bus 0. In the estimation stage, every bus gets its relative states with respect to each of its neighbors. These are combined in the exchange and update process. The relative values are combined by converting them to absolute values. To get those, the states of the neighbors need to be known. In the beginning of each sample, these start with the previous estimation results, and these change with each inner-loop of

Fig. 7.3.

If there is no voltage magnitude as measurement in the system, the states can be calculated directly and CCN does not require any other estimation method like the WLS to execute the local estimation. It becomes a complete distributed algorithm of estimation.

6.2.3 Exchange and Update

Through estimation, each cell produces a residue free raw result which needs to be improved with the process of exchange and update. The exchange process makes the use of the law of large numbers to improve the raw result. From (6.1), it is understandable that the estimation result is nothing but the true value added with some Gaussian error with zero mean. The exchange process aggregates errors from all cells to cancel each others effects. A general expression of the voltage angle is derived to show the improvement.

The expression is developed for a star network with no loop. If there are n number of neighbors, the center state is estimated with respect to each neighbor using (6.1). Then, the weighted average is taken as the updated value. For simplicity, the weights are taken to be 1.0.

Let the estimation error of (6.1) of the phase differences of line i is denoted with v_i, and the initial error associated with state j is w_j. For each neighbor of Fig. 6.2, bus 0 gets an estimation which contains these two errors,

$$\hat{\theta_0^1} = \theta_0 + v_1 + w_1 \tag{6.2}$$

$$\hat{\theta_0^2} = \theta_0 + v_2 + w_2 \tag{6.3}$$

$$\cdots$$

$$\theta_0^{\hat{|A_0|}} = \theta_0 + v_{|A_0|} + w_{|A_0|}$$

Here, A_i denotes the set of all buses directly connected to bus i and $|.|$ denotes the cardinality of it. After getting the estimations from all neighbors, the updated value is,

$$\hat{\theta_0} = \frac{1}{|A_0|} \sum_{i=1}^{|A_0|} \hat{\theta_0^i} \tag{6.4}$$

From (6.2)- (6.4),

$$\hat{\theta_0} = \theta_0 + \frac{1}{|A_0|} \sum_{n_{01} \in A_0} v_{n_{01}} + \frac{1}{|A_0|} \sum_{n_{01} \in A_0} w_{n_{01}} \tag{6.5}$$

The improvement of the update process is realizable from (6.5). As multiple v, and w terms are getting averaged, these cancel each other to some extent and produce a better result. For neighbor i of bus 0, the update results in,

$$\hat{\theta_i} = \theta_i - \frac{1}{|A_i|} \sum_{n_{i1} \in A_i} v_{n_{i1}} + \frac{1}{|A_i|} \sum_{n_{i1} \in A_i} w_{n_{i1}} \tag{6.6}$$

The second update of $\hat{\theta_0}$ through the process of average adds another term,

$$\hat{\theta_0} = \theta_0 + \frac{1}{|A_0|}\sum_{n_{01}\in A_0} v_{n_{01}} - \frac{1}{|A_0|}\sum_{n_{01}\in A_0}\frac{1}{|A_{n_{01}}|}\sum_{n_{02}\in A_{n_{01}}} v_{n_{02}} + \frac{1}{|A_0|}\sum_{n_{01}\in A_0}\frac{1}{|A_{n_{01}}|}\sum_{n_{02}\in A_{n_{01}}} w_{n_{02}} \tag{6.7}$$

Advancing in the same way, the expression of the estimated value after i^{th} exchange is found as,

$$\begin{aligned}\hat{\theta_0} = \theta_0 &+ \frac{1}{|A_0|}\sum_{n_{01}\in A_0} v_{n_{01}} - \frac{1}{|A_0|}\sum_{n_{01}\in A_0}\frac{1}{|A_{n_{01}}|}\sum_{n_{02}\in A_{n_{01}}} v_{n_{02}} \\ &... + (-1)^i\frac{1}{|A_0|}\sum_{n_{01}\in A_0}...\frac{1}{|A_{n_{0,i-2}}|}\sum_{n_{0,i-1}\in A_{n_{0,i-2}}} v_{n_{0,i-1}} \\ &+\frac{1}{|A_0|}\sum_{n_{01}\in A_0}...\frac{1}{|A_{n_{0,i-1}}|}\sum_{n_{0,i}\in A_{n_{0,i-1}}} ((-1)^{i+1}v_{n_{0,i}} + w_{n_{0,i}})\end{aligned} \tag{6.8}$$

With the increase of exchange, the number of terms containing $v_{n_{0,i}}$ and $w_{n_{0,i}}$ increases in the higher order summations as long as the system does not come to an end. As a large number of random numbers are getting added in the higher order terms, from the law of large numbers, these get converged to their true values which are zeros. Thus the higher order terms can be neglected, and the final estimated values can be approximated up to the third level as,

$$\hat{\theta_0} \approx \theta_0 + \frac{1}{|A_0|}\sum_{n_{01}\in A_0} v_{n_{01}} - \frac{1}{|A_0|}\sum_{n_{01}\in A_0}\frac{1}{|A_{n_{01}}|}\sum_{n_{02}\in A_{n_{01}}} v_{n_{02}} + \frac{1}{|A_0|}\sum_{n_{01}\in A_0}\frac{1}{|A_{n_{01}}|}\sum_{n_{02}\in A_{n_{01}}}\frac{1}{|A_{n_{02}}|}\sum_{n_{03}\in A_{n_{02}}} v_{n_{03}} \tag{6.9}$$

In (6.9), the connection with the previous estimation ($w_{n_{0,i}}$) vanishes over exchange which proofs that the initial value has less impact on the final result. However, the analysis is shown for only one iteration (for $k = 0$). The error of first estimation will get reduced in the subsequent iterations in a similar way.

The trend of the norm over the exchange and update process is shown in Fig. 6.3. It starts with a large residue, and reduces with the process.

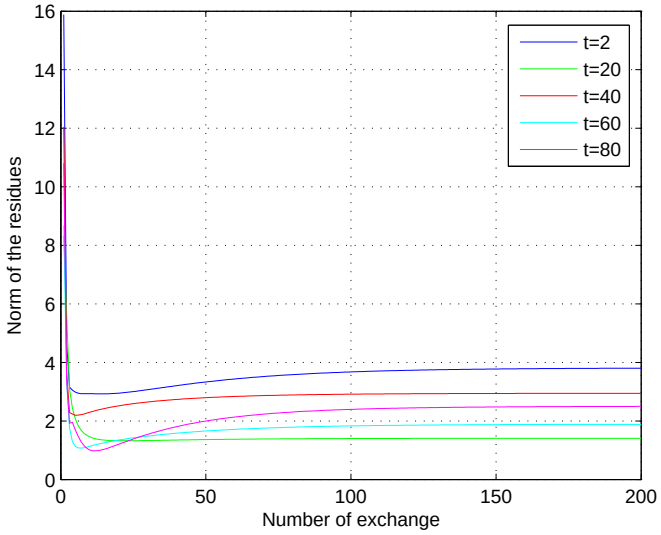

Figure 6.3: The convergence of the norm of the residues through exchange and update. The local estimation does not give a good result. By exchanging, the norm gets reduced and the states get closer to the actual values.

In summary, CCN has two main parts, estimation, and exchange and update. The estimation for a single cell creates star networks, and it reduces the WLS estimation to some direct calculations. The exchange and update process reduces the errors introduced by the estimation process.

6.2.4 Hybrid Estimation

The final result of CCN can be further improved by the use of the heuristic optimization methods. Integrating the heuristic methods with CCN creates the hybrid estimators as shown in the lower part of Fig. 7.3. The heuristic methods fail to keep the track of the states over time with fixed or random initialization. CCN guides the process and narrows the search range for the heuristic methods. The methods get a better point to start, and produce better results.

With four heuristic methods, GA, PSO, CLPSO, and OLPSO, four hybrid methods, CCN-GA, CCN-PSO, CCN-CLPSO, and CCN-OLPSO, are developed in this chapter. Out of them, the CCN-GA improves so fast that it can also be used for real-time estimation when parallelized. Its performance is discussed in Section 6.3.

Though, at first look, it seems that the required time will increase due to the addition of CCN with the existing methods, it does not. CCN choses a good region within a very short time where GA can find a better solution with a little effort. So, the number of iterations reduces and it shortens the total required time.

6.3 Test Systems

To verify the performance of the CCN based estimation, an estimator is developed in MATLAB for IEEE 68-bus New York- New England test system which is shown in Fig. 3.2. It is a 16-machine, five-area power system with 135 states. The de-

tails can be found in [58]. The system is simulated on a Real-Time Digital Simulator (RTDS). Two case studies are presented in this paper. For *Case I*, measurements are collected for 90 time samples over a time period of 3s under a disturbance. *Case II* is taken with some random changes in the system. Measurements are taken in the form of voltage magnitudes and voltage angles. These measurements are used to calculate the raw power flows. Only the real and reactive power flows of the transmission lines are taken as measurements. The raw data is mixed with some artificially generated Gaussian measurement errors. The variances of errors vary in between 1%-15% of the original measurements. However, out of the 90 time samples, the first one is executed with the principle mentioned in [10]. The rest of the estimation follows the running mode shown in Fig. 7.3. The value of k_{max} in Fig. 7.3 is set at 20.

6.3.1 Parameters for PSO and Its Variants

PSO is implemented with 20 particles. To set the parameters, different values are tested and the best one is chosen. The inertia weights of velocity of θ and V are swept from 0.5 to 0.3 and from 0.2 to 0.1 respectively. The cognitive acceleration constant is set three times the social acceleration constant. This ensures much variation in the search space for the particles. The velocities as well as the states are limited with a practical boundary for the experiment.

For CLPSO, the values of P_c are chosen in a quadratic order as done in [53]. The values vary in between 0.004 and 0.4328 which makes it choosing more frequently from the particle's own local value. The complete list is shown in *Set I*. c_1 is set to

1. Based on the experience, P_{cd} is refreshed every 20 iterations.

$$\begin{aligned} Set\ I = \{&0.0040, 0.0058, 0.0079, 0.0106, 0.0138, 0.0177, 0.0226, 0.0284, \\ &0.0356, 0.0444, 0.0551, 0.0682, 0.0842, 0.1037, 0.1276, 0.1567, \\ &0.1923, 0.2357, 0.2888, 0.3536, 0.4328\} \end{aligned}$$

In OLPSO, the OA is built on the algorithm given in the appendix of [55]. The maximum number of iterations under stagnation is set to 5. During the recovery from the stagnation, the velocities are also increased. However, in all the experiments with PSO, the random numbers are generated from a uniform distribution. The maximum number of iterations for the basic PSO, and the CLPSO is set to 5000, while it is set to 1000 for the OLPSO. For all of them, the variances of the initial particles are are set to 0.007. The variances of the initial velocities are set to 0.0007 rad and 0.000035 pu for the angle and the magnitudes respectively. For θ, the velocity is limited to ± 0.2, and for V, it is ± 0.08.

6.3.2 Implementation of GA

GA is implemented in a different way. As the improvement continues for a long time which is impractical, if the process cannot yield a specific improvement (3%) in a fixed number of iterations (500), it stops. However, the similar scheme is not possible for the PSOs due to their low accuracy. The number of chromosomes are set to 20. There are 20 crossovers and 50 mutations in each iteration. 50% of the chromosomes survive each iteration.

6.4 Simulation Results

In this section, the accuracy, required time, effects of noise, and observability of different methods are analyzed. For accuracy, both direct and hybrid methods are shown. For the hybrid method, two different cases, one under a fault, another under random changes are presented. A statistical test is run to find the level of significance of the proposed CCN-GA method.

6.4.1 Accuracy of Dynamic Behavior

State estimation is a continuous process. Accuracy is the most important quality of an estimator. As the measurements are costly, it is expected that the estimator will make the best use of it. However, it is also important to have the estimation completed within a limited time. To reduce the estimation time, the initial values for time t are taken from the estimation of $t-1$ as the states do not change much during two consecutive estimations. It is very important that the estimator keeps the track of the true values over time.

6.4.1.1 Direct methods

The abilities of the estimators to track the dynamic behavior of a specific bus are shown in Fig. 6.4. For all of them, the estimation at $t = 2$ is started with the actual values of $t = 1$. It can be seen that due to the closeness, the PSOs are performing quite well at $t = 2$. In fact, OLPSO performs better than CCN at this point.

For $t = 3$, the starting values are taken from the previous estimation by OLPSO at $t = 2$ which are not very accurate. The inaccuracies accumulate over time and make the estimator completely unsuitable for such operations. On the other

hand, though CCN takes help from the previous estimation, the final values do not depend on the starting values. As a result, CCN does not deviate much from the actual values.

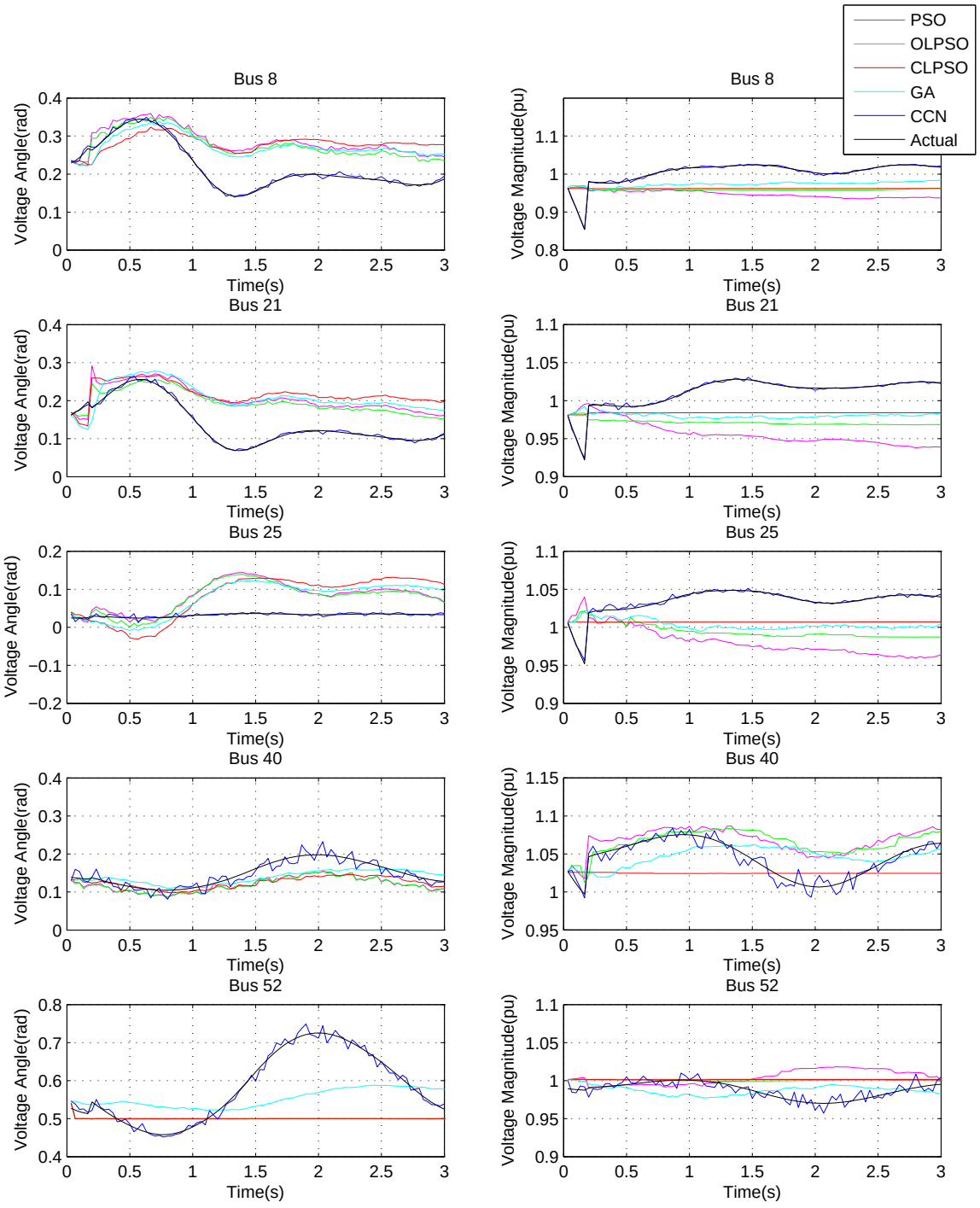

Figure 6.4: The abilities of the estimators to track the states over time. It is clear that the basic PSO, CLPSO, and OLPSO are not able to track it while CCN does quite well.

The trends of convergence of the PSOs are analyzed in Fig. 6.5. The norms of the residues over the iterations are shown for $t = 2$ where the PSOs work best. It can be seen that the norms take some time to start improving. Once it gets a better global best, the improvement starts and continues at a decreasing rate. Eventually, these get stuck at a local minimum. It is interesting that the estimator which takes the longest time to start improving, improves at the fastest rate.

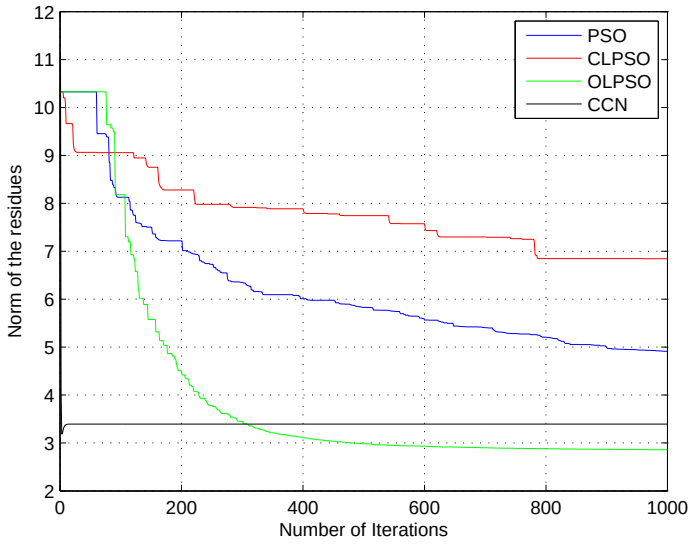

Figure 6.5: The trend of the norms of the residues over iterations. It seems that the methods take some time to start improving. Among all of them, OLPSO seems to perform better than others.

6.4.1.2 Hybrid methods

The norms of the residues for the estimators are shown in Fig. 6.6. Though CCN works better than others for most of the times, it can easily be understood that a better solution can always be obtained through a combined effort of CCN and other estimators. To direct the estimators to the right path, CCN takes lead and others work on the results of CCN. Thus the hybrid estimators are formed.

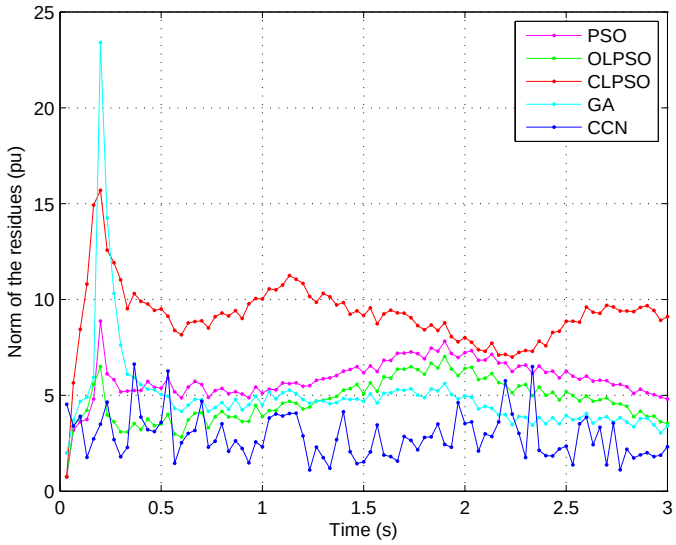

Figure 6.6: The trend of the norms of the residues of the direct methods over time. Except a few cases, CCN is performing better than others.

The settings of the PSOs and the GA for the hybrid estimators are a bit different than their direct implementations. As these can start from a better value, the number of iterations/generations is reduced to one fifth. In case of CCN-GA, the stopping criterion is changed to a threshold value within a certain number of generations. If the estimator achieves an expected norm of the residues within a maximum number of iterations, it stops. If not, it takes the best norm. However, this threshold based criterion is not applicable for the PSOs due to their poor performance.

6.4.1.2.1 Case I The norms of the residues over time for hybrid estimators are shown in Fig. 6.7. It is taken under a fault that disturbs the system states significantly. It can be seen that the CCN based GA is giving the minimum residue. The second best is the CCN based OLPSO. It is to be noted that the direct application of GA does not yield a good result, but getting directed by CCN, it gives the lowest norm within a limited time. The required time is discussed in Section 6.4.3.

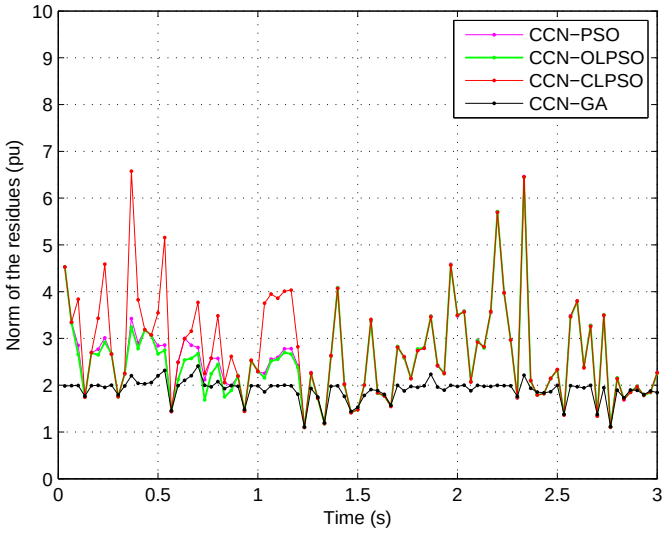

Figure 6.7: The trend of the norms of the residues of the hybrid methods over time. The CCN led GA gives the lowest norms. The threshold norm is set at 2. Except a few cases, it successfully reduces the norms to the limit.

6.4.1.2.2 Case II To increase the reliability of the proposed method, it is also applied with random changes of generation in the system. The situation resembles the normal operation that undergoes continuous load and generation imbalance. The norms of the residues are plotted in Fig. 7.4 for a longer time with higher level of measurement error. Where other estimators are suffering from periodic inaccuracies, CCN-GA is able to keep a relatively constant performance.

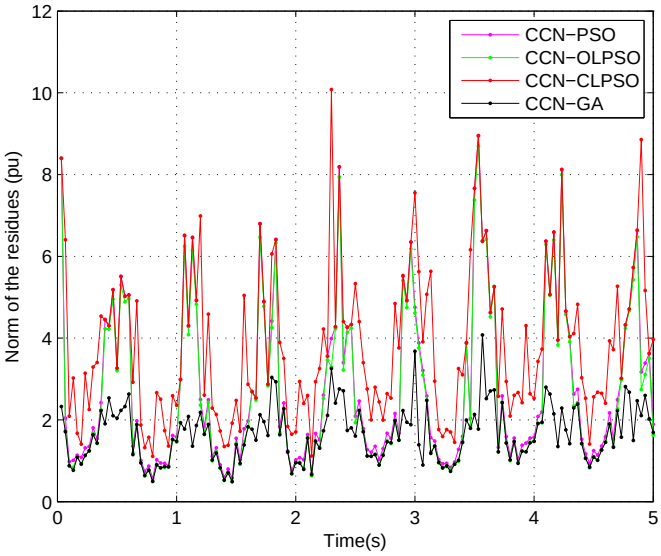

Figure 6.8: The norm of the residues of the hybrid methods with random changes in generations. The CCN led GA keeps a moderate constant rate.

6.4.2 Statistical Comparison of Accuracy

It is important to compare the results using statistical method. Kolmogorov-Smirnov test is a well-known test for the purpose [63]. It compares the cumulative distribution functions (CDFs) of the samples and takes the maximum absolute value of their difference, d. This is compared with the significance level α. If d is greater than the significance level, then there is a significant difference between the samples.

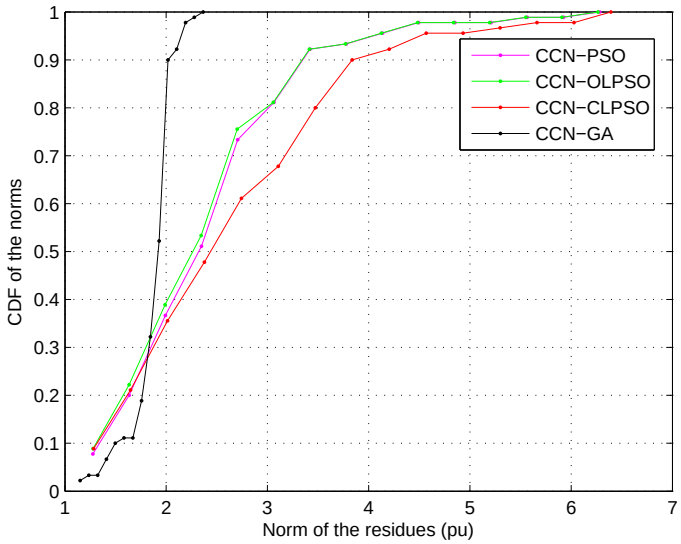

Figure 6.9: Statistical comparison using the K-S test of the results found for Gaussian noise with fault. The CDFs are plotted for the overall norms of the 90 samples of data.

Using the method, it is found that CCN-GA is significantly better than the others in term of norm of the residues for all typical values of α, 1%,5%, and 10%. The CDFs shown in Fig. 6.9 are taken for *Case I*. Both CCN-PSO and CCN-OLPSO have significant differences with CCN-CLPSO for 5% or higher value of α. However, there is no significant difference between PSO, and OLPSO.

6.4.3 Required Time

As mentioned earlier, integration of the CCN to the existing optimization methods helps to reduce the number of iterations/generations. Thus, it helps to save a significant amount of time. Among the hybrid estimators, CCN-GA does not only yield the best accuracy, the required time is also noticeable. In Fig. 6.10(a), it can be seen that the CCN-GA requires the lowest computation time in all cases. The

experiments are run under similar conditions on an Intel(R) Xeon(R) CPU (E5-2609) with 2.4 GHz core and 48GB of memory. For some samples, the output from CCN yields a norm below the threshold and in these cases, the required times shown are the times used by the CCN. Real-time state estimation can be realized if these methods can be parallelized. As seen from Fig. 6.10(b), CCN-GA is the method with the least cumulative runtime for parallelization and real-time implementation.

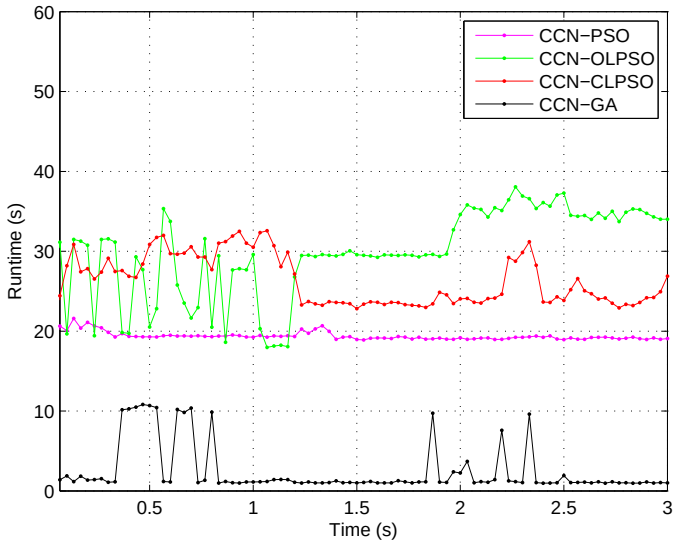

Figure 6.10: The required time for the hybrid methods to achieve the accuracy shown in Fig. 6.7. The CCN led GA requires the minimum time.

6.4.4 Effects of Noise

Noise is an important factor in power system state estimation. The level as well as the distribution of noise needs to be analyzed. The effect of the level is shown in Fig. 6.11. Each point shows the cumulative norm of four consecutive samples. It is clear that the accuracies of all estimators decrease with the increase of noise level. However, CCN-GA provides the best solution for every level of noise. To analyze the

effect of the distribution of noise, the estimators are run with uniform errors. The simulation results are shown in Fig. 6.12. CCN-GA outperforms other estimators in almost all samples.

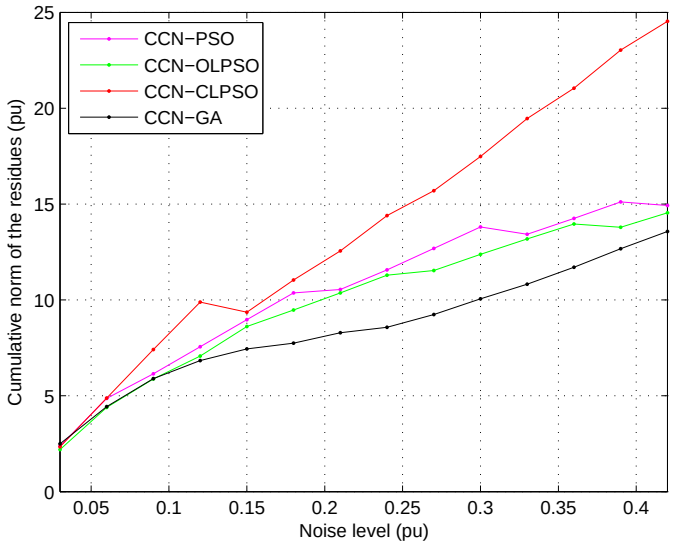

Figure 6.11: The response of the estimators to different levels of noise. It can be seen that the norms of the residues increases for all of them while remains lowest for CCN-GA.

6.4.5 Observability

Observability is an integrated part of the estimator. It refers to the ability of the estimator to perform estimation with minimum number of measurements. In this regard, PSOs and GA perform better than the CCN. These can work with any observable network. But for CCN, local observability is needed for every cell. However, in case of insufficient local measurements, the cells can merge to form a bigger observable *supercell* and it can eliminate the necessity of extra measurements. Another way to eliminate the extra measurements is to use the gossip-based distributed methods [3, 64] which do not require any local observability.

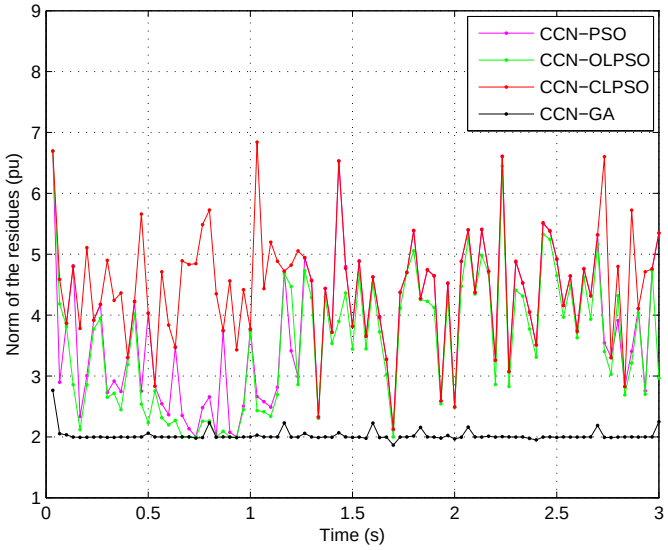

Figure 6.12: The norm of the residues with uniform distribution of the measurement errors. The proposed CCN-GA performs better than other hybrid estimators.

6.4.6 Qualitative Analysis of the Simulation Results

From Figs. 6.7, 7.4, and 6.12, it is visible that CCN-GA is performing better than other hybrid methods most of the times. CCN is the common part of all methods. The differences in performance come from the heuristic parts. Besides the acuumulation of errors, two major problems are found with the PSOs in power system state estimation, tuning of a large number of parameters, and the low number of fitness evaluations. PSOs have a high number of variable to choose (around 15), and these work quite well within a limited range. With the change of the states, these need to be changed. State estimation is a real-time continuous process and it is not possible to test different parameters of PSOs. As a result, the CCN led PSOs cannot keep high performance over time. Moreover, due to the higher execution time, it cannot evaluate the fitness function for a large number of times. This leads to a less

accurate result for the PSOs.

On the other hand, mutations of GA is a finer tool to find better positions. The amount of change in mutation can be controlled directly to dig deeper in the solution space. As an iteration of GA takes significantly less time than that of OLPSO, it can run more iterations with less time to find the lowest norms. This makes CCN-GA a better choice.

There are different qualitative measures to compare the performances of the hybrid estimators. For the following analysis, instead of taking the norms, the mean squared errors (MSE) are shown as the measure of accuracy. Two metrics are used for convergence, the accuracy of convergence, and the probability of convergence. Besides these two, the required time to convergence, and the number of fitness evaluations are also shown in Table 6.1. 25 trials are taken for both *case I* (with 90 samples) and *case II* (with 150 samples). Unlike previous setups, the estimators are run for an unbounded time till their convergence. The metrics used are given below,

i) Convergence: If the method does not improve the norm of the residues by 1% *in* 100 *consecutive iterations, it is considered to have converged.*

ii) Mean Square Error (MSE): The mean of the square of the difference of the actual measurements and the estimated measurements. If the number of samples is N_s,

$$MSE = \frac{1}{mN_s} \sum_{j=1}^{N_s} \sum_{i=1}^{m} (z_{ij} - \hat{z_{ij}})^2 \tag{6.10}$$

iii) Time to converge: The required time to reach convergence (as described in (i)) for each sample.

iv) Number of fitness evaluations: The average number of times the norms of the residues (fitness function) are calculated for each sample to reach convergence.

v) Probability of convergence (POC): The percentage of the states whose estimated values fall within a defined range of the actual values. The range is taken 0.01 for the experiments.

Table 6.1: Performance of the hybrid estimators with different metrics

	Features	**CCN-PSO**		**CCN-OLPSO**		**CCN-CLPSO**		**CCN-GA**	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
Case I	MSE (pu)	0.045	3.24e-4	0.0367	3.1e-4	0.0454	6.9e-4	0.0143	6.92e-5
	Time to converge (s)	8.75	4.5	46.88	16.28	6.25	0.87	6.65	1.63
	Number of fitness evaluations	7289	246	5413	59.4	6623	111.9	11238	42.1
	Probability of convergence	89.93%	0.056%	89.75%	0.071%	90.09%	0.04%	90.12%	0.065%
Case II	MSE (pu)	0.074	4.92e-4	0.0686	3.38e-4	0.1145	1.6e-4	0.0246	4.35e-5
	Time to converge (s)	25.38	16.15	73.80	21.91	6.84	1.395	18.84	6.08
	Number of fitness evaluations	15539	557.93	7707.9	73.55	7416.9	127.94	28701	114.24
	Probability of convergence	82.01%	0.086%	81.67%	0.056%	82.18%	0.027%	82.05%	0.019%

Though CCN-CLSPO may require less time to converge, it is due to the premature convergence. The probability of convergence is also higher for CCN-CLPSO, but that is not the objective of the optimization problem. The objective function is to minimize the MSE and CCN-GA performed better than others in that.

The differences between the CCN-PSO, CCN-OLPSO, CCN-CLPSO and CCN-GA are summarized below in Table 6.2.

6.4.7 Comparison of Static and Semi-dynamic Estimator

The performance of the static and the semi-dynamic estimators are shown in Fig. 6.13. It can be seen that the semi-dynamic estimator works better than the static estimator. It is due to the fact that the semi-dynamic estimator can improve

Table 6.2: Summary of the performance of the hybrid methods

Qualities	**CCN-PSO**	**CCN-OLPSO**	**CCN-CLPSO**	**CCN-GA**
Accuracy	Medium	High	Low	High
Rate of convergence	Medium fast	Slow	Fast	Fast
Number of fitness evaluations	Medium	Low	Low	High
Probability of convergence	Medium	Medium	Medium	Medium
Dependency on initialization	Largely dependent	Largely dependent	Largely dependent	Less, dependent
Real-time applications	Not suitable	Not suitable	Not suitable	Suitable
Number of parameters	High	High	High	Low
Online tuning of parameters	Needed	Needed	Needed	Not Needed

its result with multiple iterations of k. On the other hand, the static estimator does not have the option for such a loop.

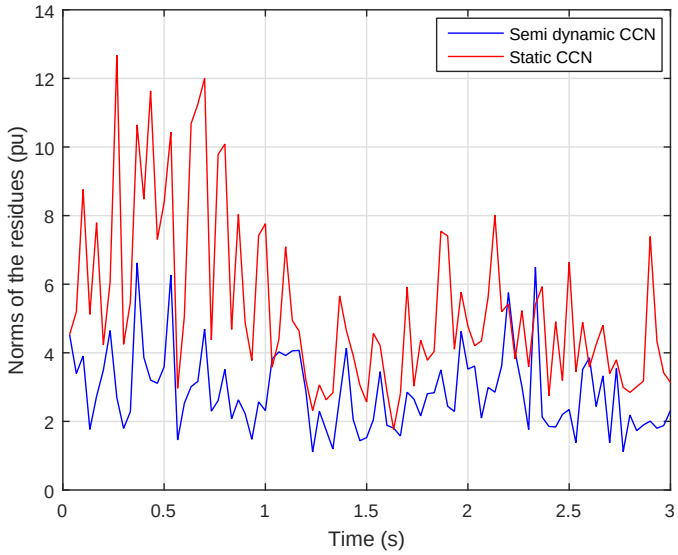

Figure 6.13: The performances of the static and semi-dynamic CCN based estimator.

6.5 Summary

State estimation of large power systems is a challenging task for heuristic optimization methods. In this paper, a hybrid state estimator based on CCN and GA is developed to perform state estimation accurately and in a fast manner. Two parts of the CCN method, estimation, and exchange and update, are investigated analytically. CCN takes the values close to the optimal solution and prevents the heuristic methods to search in non-related regions. The CCN-GA method is implemented for state estimation on IEEE 16-machine 68-bus power system and improved results compared to other heuristic method based hybrid estimators, CCN-PSO, CCN-CLPSO, and CCN-OLPSO, are obtained.

Future work will involve parallel processing of the CCN-GA to realize a real-time hybrid state estimator. Moreover, the distributability of the methods needs to be explored. Different hybridization methods can also be investigated to find a better solution. To reduce the requirement for local observability, the supercells also need further analysis.

Chapter 7

CCN Based Distributed State Prediction

7.1 Introduction

State prediction can become an important concept in power systems operation. With the advent of the PMU measurements, the dynamics of the system can be tracked. As a result, it is possible to predict the states of the system and they can be used in multiple parts of the power systems.

One of the major applications of the predictor can be its use in contingency analysis (CA). In the traditional CA, the states are considered to remain the same at the time contingency. However, the states change over time and the predicted values can give a better accuracy for the CA. Moreover, with the predicted values, some possible near-future conditions can be simulated as part of the CA.

The predictor can also play an important role in detecting large changes in the system. It always assumes a normal operating condition and predicts the behavior of the states according to that. If the system undergoes a rapid change, the estimated

states and the predicted states will differ significantly. Based on their difference, it will be possible to take some preventive measures.

The prediction can be of different step size. With the increase of step size, the accuracy of the prediction reduces. As a result, a trade-off exists in between the accuracy and prediction steps. In this chapter, both the single-step and multi-step ahead predictors are developed and their performances are shown through simulation.

7.2 Background

Though the concept of state estimation is not new, prediction is not used in power system. As mentioned earlier, it has a special use in Kalman filtering which is neither distributable nor extensible to multi-step.

7.2.1 Elman Recurrent Neural Network

One of the most well known variants of NN is Elman Recurrent Neural Network (RNN). In general, the RNNs are useful for tracking the dynamic states. Beside the input, output and the hidden layer, Elman network has a context layer as shown in Fig. 7.1. The output of the hidden layer are fed in itself before being multiplied with the output weights. Let the main input be denoted with $\mathbf{x_i}$, and the context layer input be denoted with $\mathbf{x_c}$.

The full input $\mathbf{x_F}(t)(= [\mathbf{x_i^T x_c^T}]^T)$ is multiplied with the input layer weight matrix $\mathbf{w}$ and reaches the hidden layer. A bias is added with the input layer. Let the hidden layer include N neurons. The output of the hidden layers, $\mathbf{a}$ are passed through normalizing units and produces $\mathbf{d}$. The normalized values $\mathbf{d}$ are saved in the context layer for using in the next time step. The next input $\mathbf{x_F}(t+1)$ is formed with input and context layer neurons. The values of $\mathbf{d}$ are also multiplied with the output

layer weights $\mathbf{v}$ to form $\hat{\mathbf{y}}$ for the current step t.

$$\begin{aligned} \mathbf{a} &= \mathbf{w}\mathbf{x}_\mathbf{F} \\ d_n &= \frac{1}{1+e^{-a_n}} \quad \text{for } n = 1...N \\ \hat{\mathbf{y}} &= \mathbf{v}^\mathbf{T}\mathbf{d} \end{aligned} \tag{7.1}$$

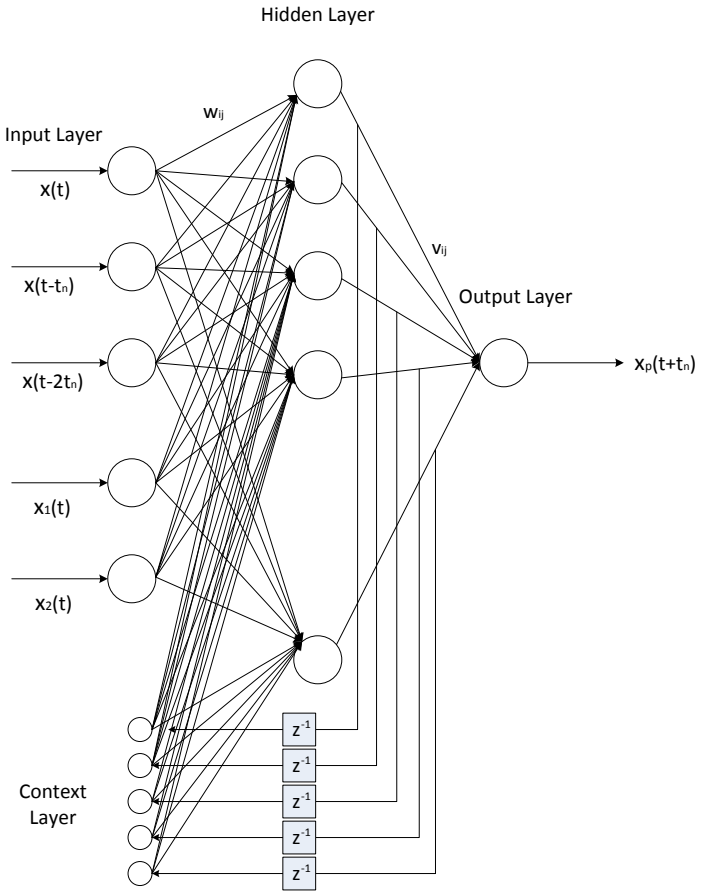

Figure 7.1: The structure of the Elman recurrent neural network. The output of the hidden layer is fed back in the next time step.

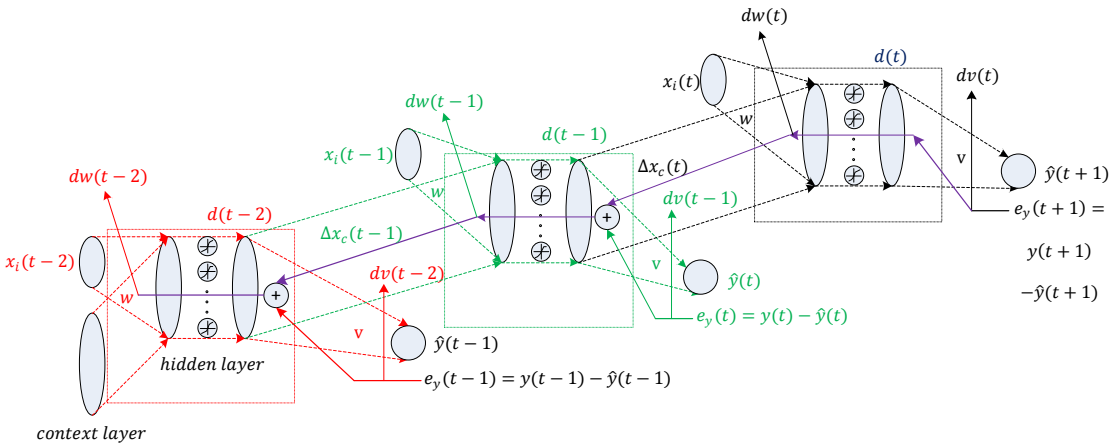

Figure 7.2: The working principle of the back-propagation through time. A two step unfolded network showing back propagation of error at time t.

7.2.2 Training of the Elman Network

Before using, the network needs to be trained with a known sequence of input and its corresponding output. The training of the Elman network is different from the training of the canonical NN. It requires unfolding of the recurrent network to basic networks and back-propagation is applied on those networks. This special method is known as back-propagation through time (BPTT) [65].

7.2.2.1 Basic back-propagation

Back-propagation is a very simple and effective tool for training a non-recursive network. The known sequence is fed in the network and an output is found. The output is compared with the expected output and the error $\mathbf{e_y}$ is determined. The error is fed back to the network and the contribution of different parts i.e. $\mathbf{e_d}$, and $\mathbf{e_a}$ are calculated. Then the weights are updated according to the contributions. It can

be done in batch processing, i.e. the errors are summarized for all training samples and the weights are updated with that. The process is described in the following set of equations.

$$
\begin{aligned}
&\mathbf{e_y} = \mathbf{y} - \hat{\mathbf{y}} \\
&\mathbf{e_d} = \mathbf{v^T e_y} \\
&e_{an} = d_n(1 - d_n)e_{dn} \qquad \text{for } n = 1...N \\
&\mathbf{\Delta v} = \gamma_m \mathbf{\Delta v} + \gamma_g \mathbf{e_y d^T} \\
&\mathbf{\Delta w} = \gamma_m \mathbf{\Delta w} + \gamma_g \mathbf{e_a x_F^T} \\
&\mathbf{v} = \mathbf{v} + \mathbf{\Delta v} \\
&\mathbf{w} = \mathbf{w} + \mathbf{\Delta w}
\end{aligned}
\tag{7.2}
$$

Here, γ_g, and γ_m are the learning gain and the momentum gain respectively. γ_g determines how fast the network should learn from one sample of the training sequence. On the other hand, γ_m determines the rigidity of the network to changes.

7.2.2.2 Back-propagation through time

Due to the recurrent nature of the Elman network, the training is different. BPTT is an offline process of training the recurrent networks. A simple training method is shown in Fig. 7.2.

To integrate the effects of previous input, the network is unfolded upto a certain times, h. This is the depth of the network. In Fig. 7.2, the process of BPTT is shown for $h = 3$. For a single time t, the inputs are fed for $\mathbf{x_i}(t-2)$, $\mathbf{x_i}(t-1)$, and $\mathbf{x_i}(t)$. The weights $\mathbf{w}$, and $\mathbf{v}$ remain the same throughout the forward process. The error for prediction of the last stage $\mathbf{e_y}(t+1)$ is fed back to update $\mathbf{v}$, and $\mathbf{w}$. For

$\mathbf{v}$, the update is similar to (7.2). However, as $\mathbf{w}$ is affected by the values of $\mathbf{d}$, the values of $\mathbf{e_d}(t-1)$ will be affected with both $\mathbf{\Delta d}(t)$, and $\mathbf{e_y}(t)$. If $\mathbf{\Delta x_F}(t)$ represents the corresponding change of input, from the gradient based analysis, the expression of it is found as follows,

$$\mathbf{\Delta x_F}(t) = \mathbf{e_a^T}(t)\mathbf{w} \tag{7.3}$$

$\mathbf{\Delta x_c}(t)$ is the part of $\mathbf{\Delta x_F}(t)$ corresponding to the context layer. It gets added with the effect of $\mathbf{e_y}$ to form the complete effect on $\mathbf{d}(t-1)$,

$$\mathbf{e_d}(t-1) = \mathbf{\Delta x_c}(t) + \mathbf{v^T e_y}(t) \tag{7.4}$$

The process runs till the starting network and the necessary corrections are determined. Then the last one of them are used to update the values of $\mathbf{w}$ and $\mathbf{v}$.

7.2.3 Application of the State Predictor

Any time ahead prediction can be helpful in the operation of the power system. The number of time steps depends on the time needed for the operator to take any action. Different actions require different amount of time to be taken. As a result, multiple predictor can run with different time steps to serve different purposes. It is important to remember that the accuracy of the prediction depends on the time step. If the prediction time is longer, the accuracy reduces.

The main application of the predictor can be in state estimation, contingency analysis, load forecasting, automatic voltage regulation, load frequency control, automatic generation control, economic dispatch etc.

7.3 Test Power System

The efficiency of the proposed predictor is tested in IEEE 16-machine 68-bus NY-NE test system. It has a total of 135 states with 83 transmission lines. For each bus, there are two computation cells for estimating the magnitude and the angle as shown in Fig. 7.3. As a result, there exists a total of 135 ERNN cells which are separate from each other. They take the states of all neighboring cells. The states are taken directly from the simulation of the 68-bus system in Real Time Digital Simulator. In practice, they can be taken from the state estimator.

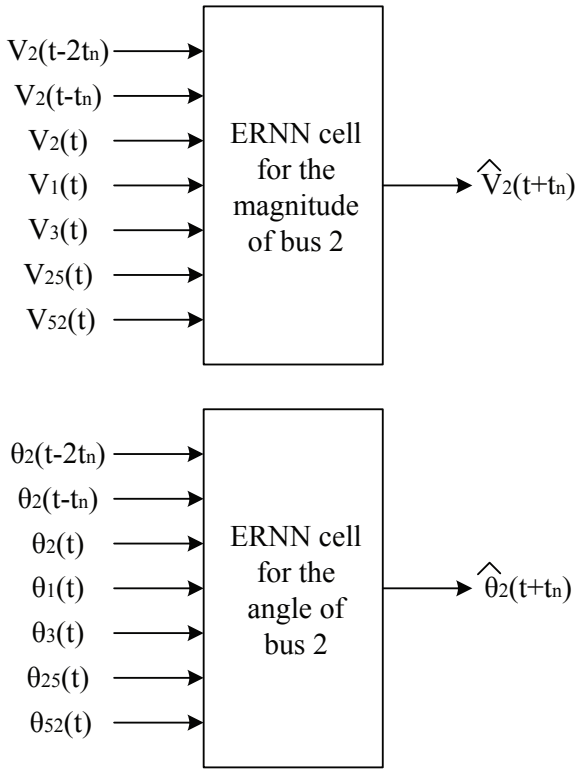

Figure 7.3: Inputs and outputs for the two cells of bus 2.

7.3.1 Training Data

For both the single-step and the multi-step predictions, the training is done with 20000 data sampled at 30 Hz from the simulation of the system. The samples are iterated over 100-500 epochs to fit the training signals properly. To ensure a good amount of disturbance in the system, pseudo random binary signals (PRBS) are applied in the excitation control of the generators while collecting the measurements. A part of the 16 PRBS signals for 16 generators is shown in Fig. 7.4. The random changes take randomness in the training signals. The corresponding changes in the outputs of the generators are shown in Fig. 7.5.

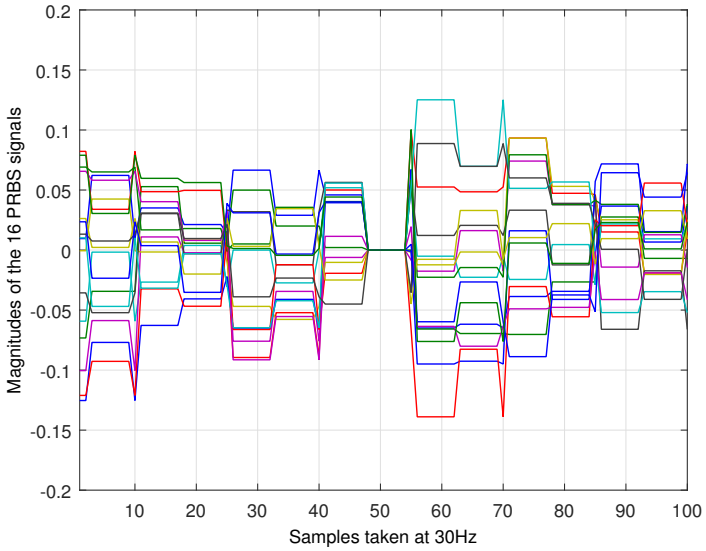

Figure 7.4: PRBS signals used to perturb the generators in the power system.

The voltage magnitudes and the angles are separated. So, for each cell, there are two separate networks at each cell. Each cell uses the previous values of the corresponding state as well as the current value of it to predict the next value. If it predicts for n step ahead result, it uses the previous values of $t - t_n$ and $t - 2t_n$. For the neighbors, it only uses the data of current values.

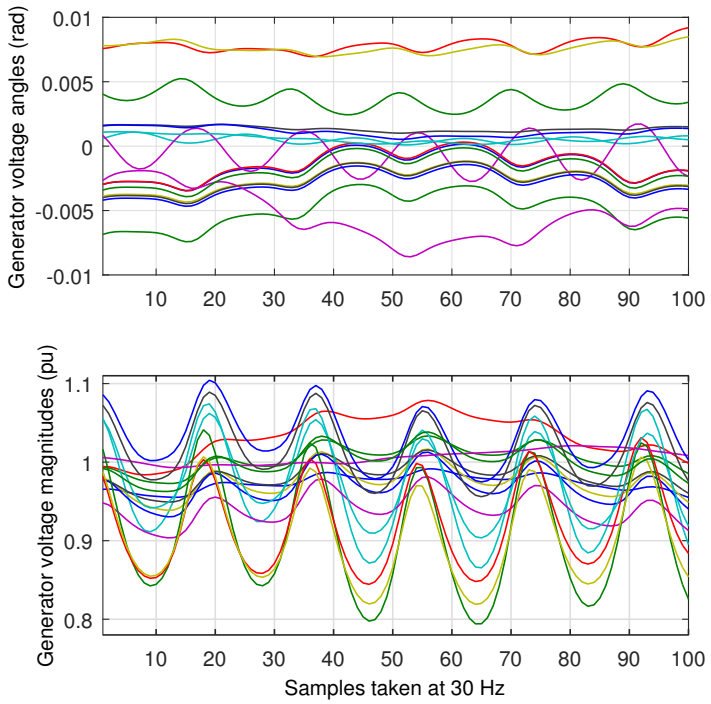

Figure 7.5: The output of the generator voltages due to the PRBS signals.

The number of hidden layer neurons are taken to be double of the input layer. The number of the neurons of the context layer is equal to that of the hidden layers. As the network of each cell is producing either the magnitude or the angle, there is only one output neuron for each network.

The context layer is initialized with random weights. The starting inputs for this layer are zeros. As shown in Fig. 7.1, there is a one step delay between the output and the input part of the context layer.

7.3.2 Testing Data

The testing is done with 15428 samples of data taken under the same condition as of the training data. It is also initiated with zero values in the context layer. Though it gives some incorrect results in the beginning, the effects wear out very soon.

7.4 Simulation Results

Accuracy is the most important quality of any predictor. Though the accuracy of the voltage magnitude is well enough, the angles suffer a lot. The reason is the very low variation of phase angles over time. It makes the tracking difficult for the predictor.

To measure the accuracies of the predictors, Mean Absolute Percentage Error (MAPE) is taken for a single state over a specific number of time samples. MAPE is defined as,

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} |\frac{V_{jt} - \hat{V}_{jt}}{V_{jt}} \times 100\%| \tag{7.5}$$

Here, MAPE is taken over n time steps for bus j. V_{jt}, and $\hat{V}_{jt}$ represents the true and predicted value of the corresponding state V.

7.4.1 Single-step Prediction

The accuracy is important for both the training session as well as for the testing session. With a repetition of 100-500 epochs, the training session usually gets an acceptable accuracy as shown in Fig. 7.6. For the testing session, the accuracies of the phase angles and the voltage magnitudes of bus 8, 25, and 52 are shown in Figs. 7.7, and 7.8. These are carried out for a single step prediction.

7.4.2 Multi-step Prediction

As the single step prediction is not suitable for some operations, multi-step predictions are needed. The training is done for multiple steps and the testing is shifted accordingly. The inputs are shifted according to the step size. For example,

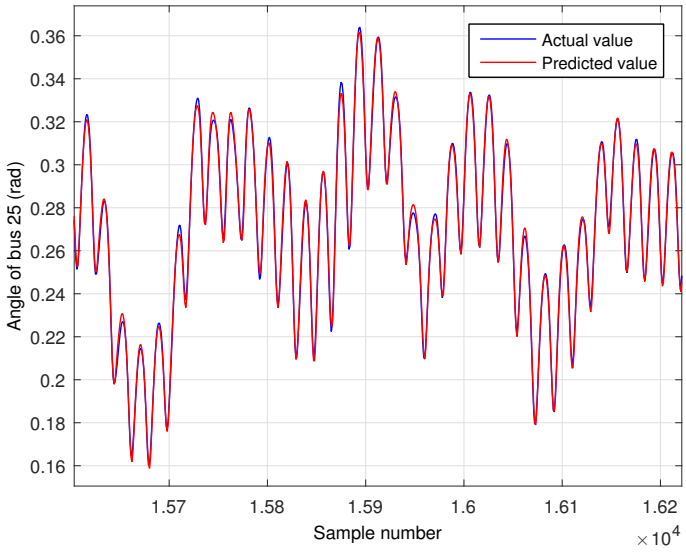

Figure 7.6: A part of the final training values and the predicted training values.

if the step size is six, the previous values are taken from $t-6$ and $t-12$. It is found that the multi-step prediction is less accurate than the single step prediction. A six step ahead prediction results are shown in Figs. 7.9, and 7.10.

The MAPEs and the standard deviation of the absolute prediction error (STD) for the single and the multi-step predictions are shown in Table 7.1.

Table 7.1: MAPE±STD for Predictions

Bus	Single-step		Six-step	
	|V|	θ	|V|	θ
8	0.77±0.61	3.48±2.29	1.84±1.21	16.5±15.4
25	0.85±0.57	3.81±2.42	2.41±1.52	15.5±12.9
52	0.55±0.39	6.81±4.37	2.19±1.27	24.4±31.4

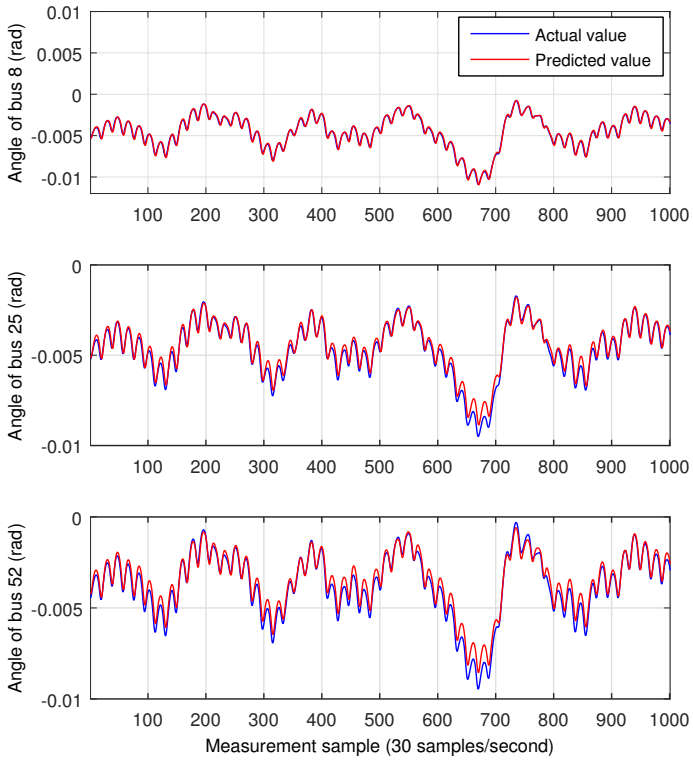

Figure 7.7: A part of the testing values and the predicted testing values for a single step prediction of phase angle.

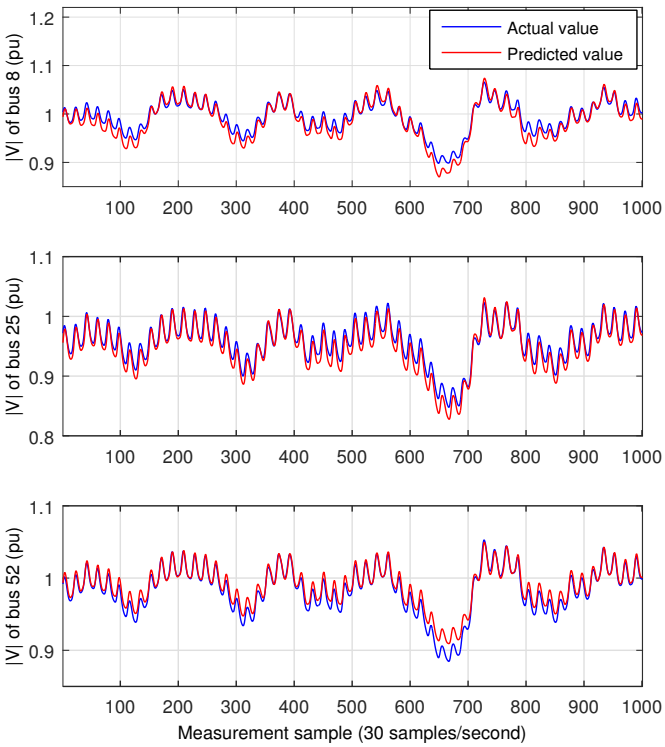

Figure 7.8: A part of the testing values and the predicted testing values for a single step prediction of voltage magnitude.

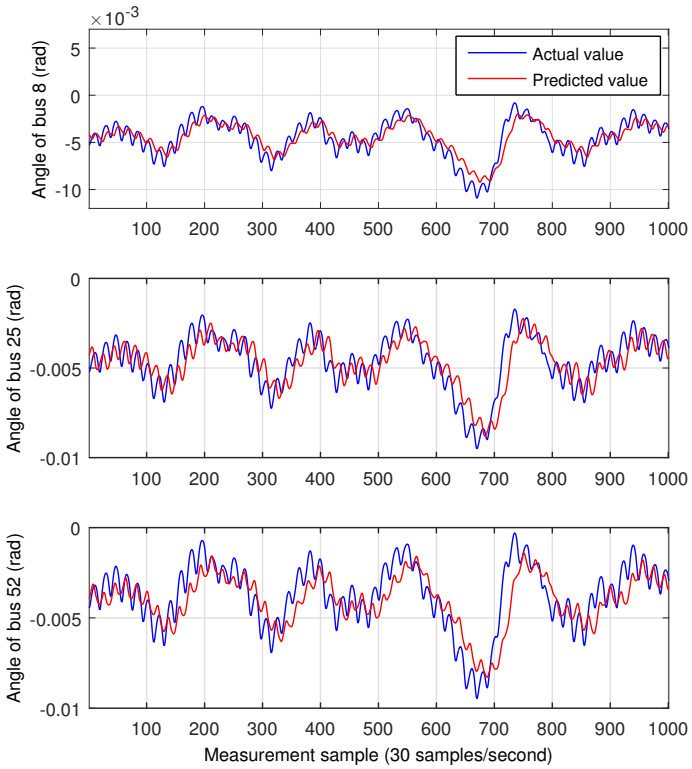

Figure 7.9: A part of the testing values and the predicted testing values for a six step prediction of phase angle.

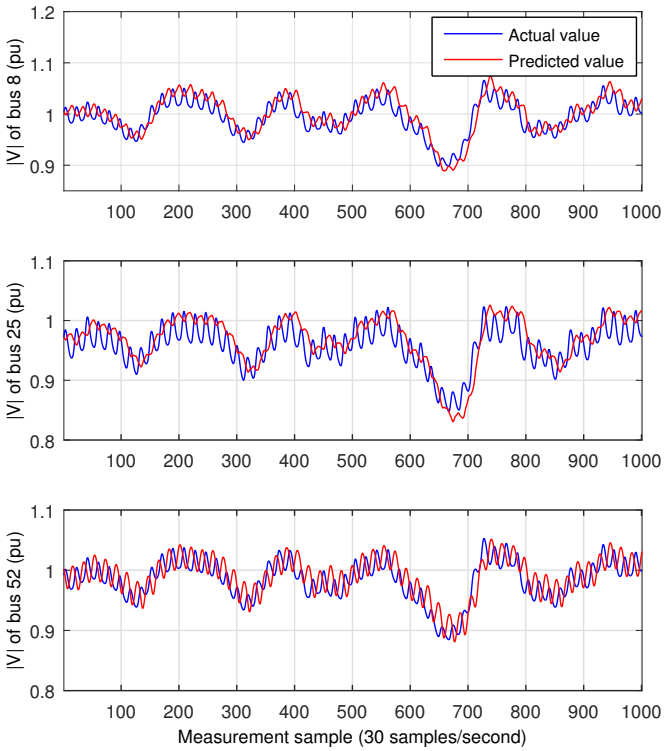

Figure 7.10: A part of the testing values and the predicted testing values for a six step prediction of voltage magnitude.

7.5 Summary

A CCN inspired ERNN based dynamic state predictor is proposed for power systems and the accuracy of the predictor are shown through simulation in this chapter. The results show that the network works well for the single step prediction. The accuracy of the multi-step predictor needs further analysis.

It should be noted that the same method of predictions can be applied for predicting the renewable energy sources like the solar or wind power. The renewable energies can change rapidly which is one of the major concerns. A proper prediction of them can increase their contribution to the main grid significantly.

The fully connected CCN is left as the future work. The single step prediction

can be effectively used with distributed dynamic state estimation. It will be shown in the next chapter. The relation between the step size and the accuracy of the predictor is also an important area of research. Though not explored, the accuracy may get better with some periods of steps.

Chapter 8

Distributed Dynamic State Estimator

The application of a dynamic estimator depends on the nature of the states. If the states change randomly over time, they can be considered as static. On the other hand, if they show a gradual change which can be tracked properly, it is dynamic. Before deciding what to use, it is important to investigate more about the true nature of the states.

8.1 Nature of the State Variable

In the traditional SCADA system, measurements are taken at a very slow rate of around 1 sample per 2-4 seconds. Under this rate, the collected samples miss some important changes and they may look random. So, the use of the static WLS estimator is rational for this rate.

However, the estimator is getting faster day by day, and it requires a faster rate of collection of measurements. In recent time, PMUs are serving the purpose. With the slowest PMU rate, i.e., 30 samples per second, the measurements show a complete dynamic nature. It can be easily observed from the actual values of

the previous simulation results that except during a fault, the states never change abruptly. The actual values of the voltage magnitude of bus 25 is shown in Figure 8.1.

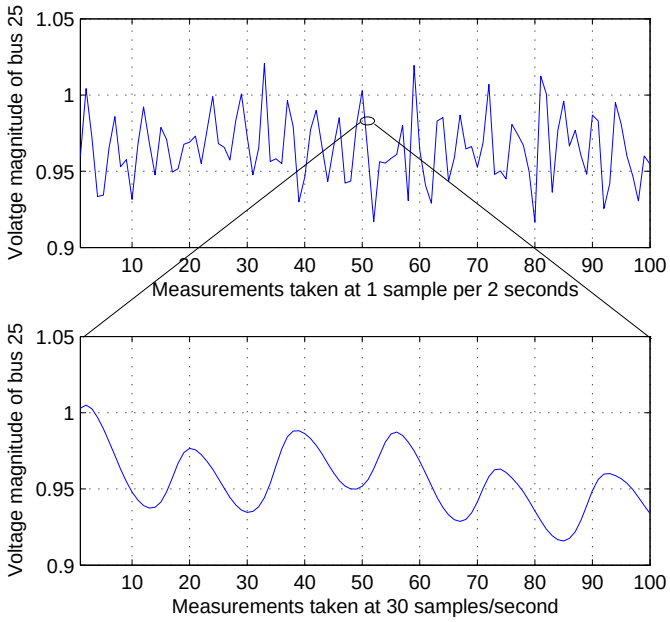

Figure 8.1: The trend of one state variable under two different sampling rates. The upper one is taken at SCADA rate where the lower one shows at the PMU rate.

Using a static estimator can be inefficient in dynamic system. Though the WLS estimator is the most efficient static estimator, it tries to optimize the states for one sample only and does not take the advantage of the dynamic nature. As a result, it cannot yield a better accuracy than the dynamic estimators.

8.2 CCN Based Predictive State Estimation

A distributed dynamic estimator can be the future of the state estimation. Distributed estimation will preserve the privacy of the local market. At the same

time it will be fast and scalable.

The effect of CCN is already shown in Chapter 5, and 6. It is an effective method to distribute the responsibilities of a centralized process. On the other hand, using the dishonest method yields a very fast and accurate result. Implementing the dishonest method with CCN framework can be a good option to take the advantage of both of them.

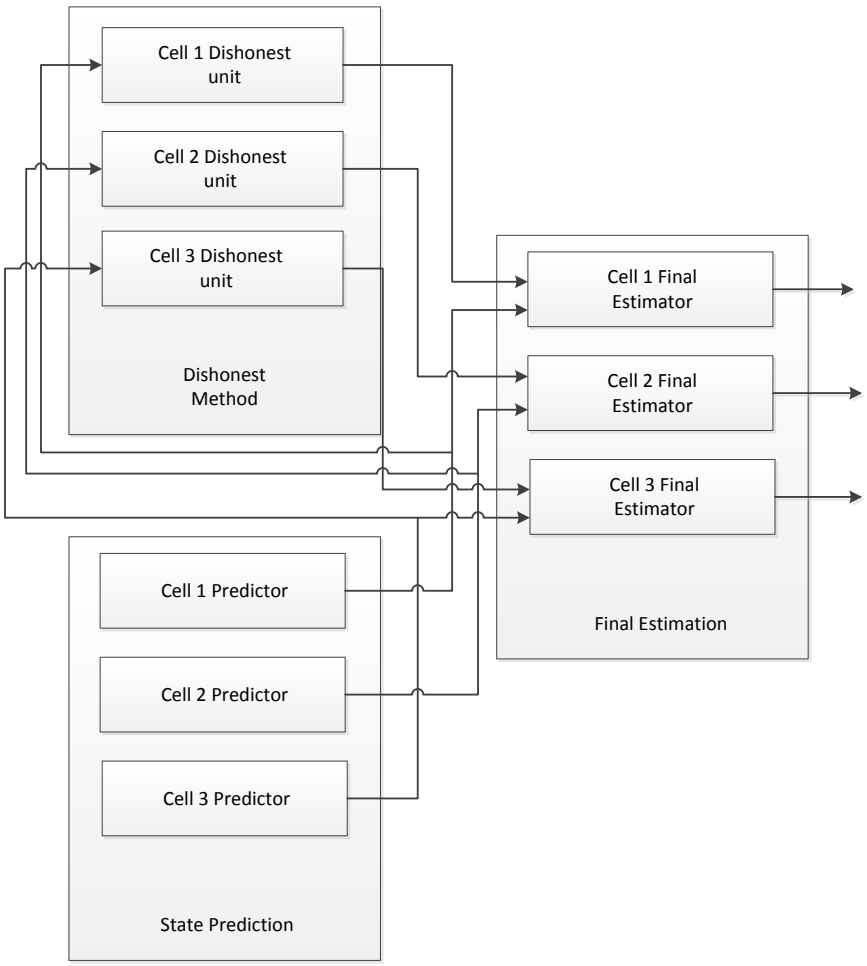

Figure 8.2: Block diagram of the planned dynamic estimator. It preserves the distributed architecture of the CCN method.

Building a fast, distributed, and dynamic estimator is the main objective of the dissertation. Taking the advantage of the predicted value from the CCN based distributed predictor, the CCN based dishonest method can give us the final product as shown in Figure 8.2. The application of the cellular method for IEEE 14-bus system is shown in Fig. 8.3, and 8.4.

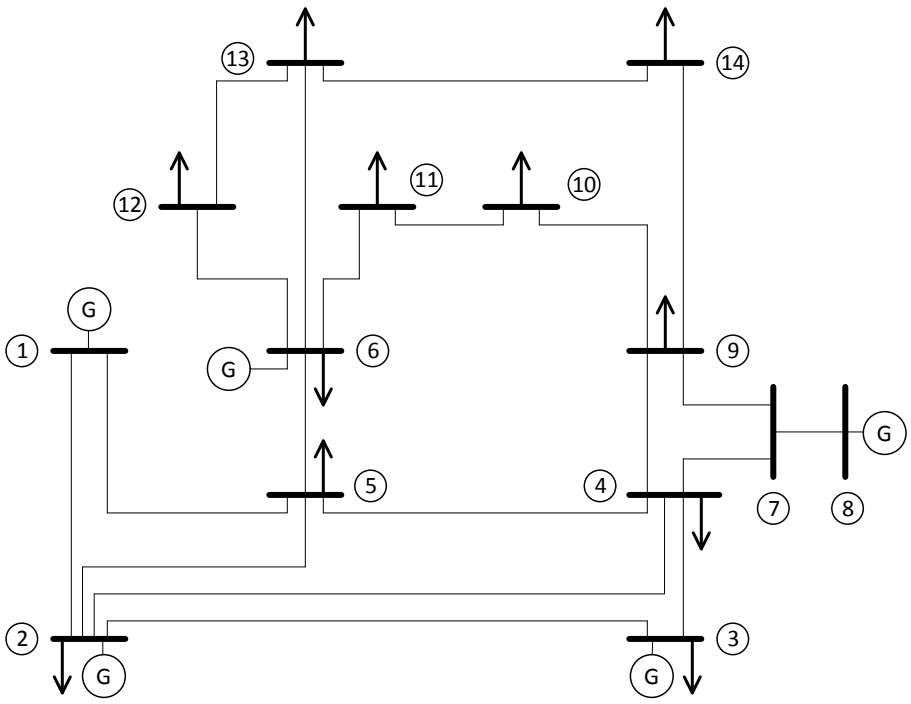

Figure 8.3: The basic diagram of the IEEE 14 bus test system.

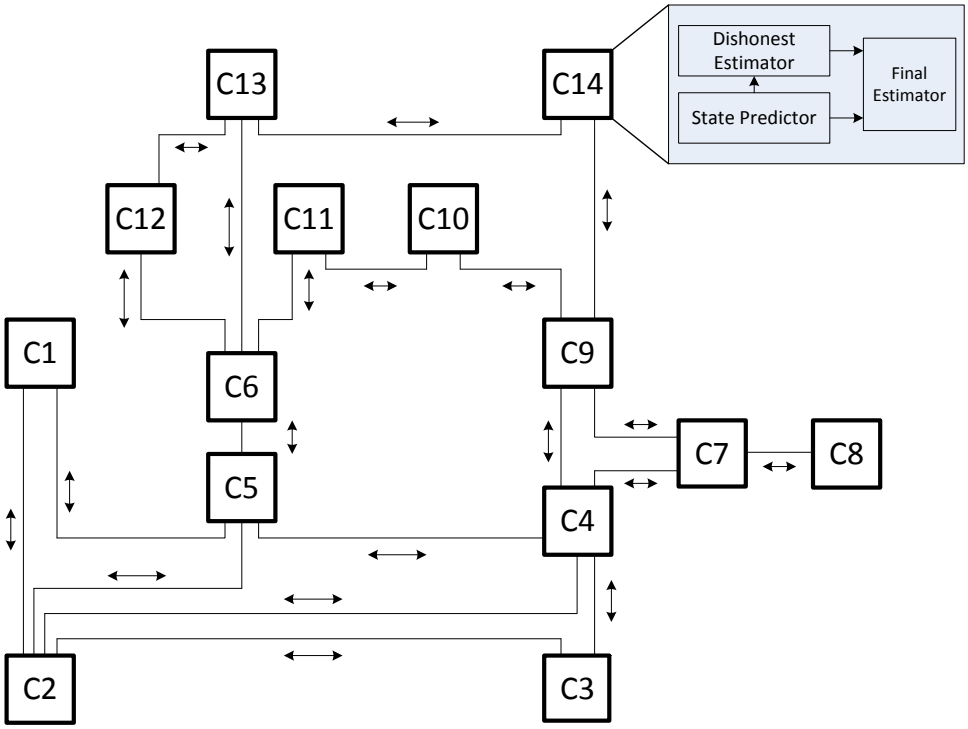

Figure 8.4: The planned estimator for IEEE 14 bus test system.

8.2.1 Distributed Dishonest Method

In Chapter 6, the canonical WLS estimator is used with CCN to make a semi-dynamic estimator. In this chapter, the dishonest method is distributed with CCN. The CCN based dishonest method produces an accurate estimation, and it can also use the previous estimation to normalize the power measurements. Instead of using the previous estimation, in this dissertation, the predicted values are used.

8.2.1.1 Description

The structure of the distributed estimator is shown in details in Fig. 8.5. The output of the predictor is sent to both the CCN based dishonest estimator and to

the final estimator. The cellular dishonest estimator normalizes the measurements using the output of the predictor and produces the estimated value. The output of the dishonest estimator is sent to the final estimator.

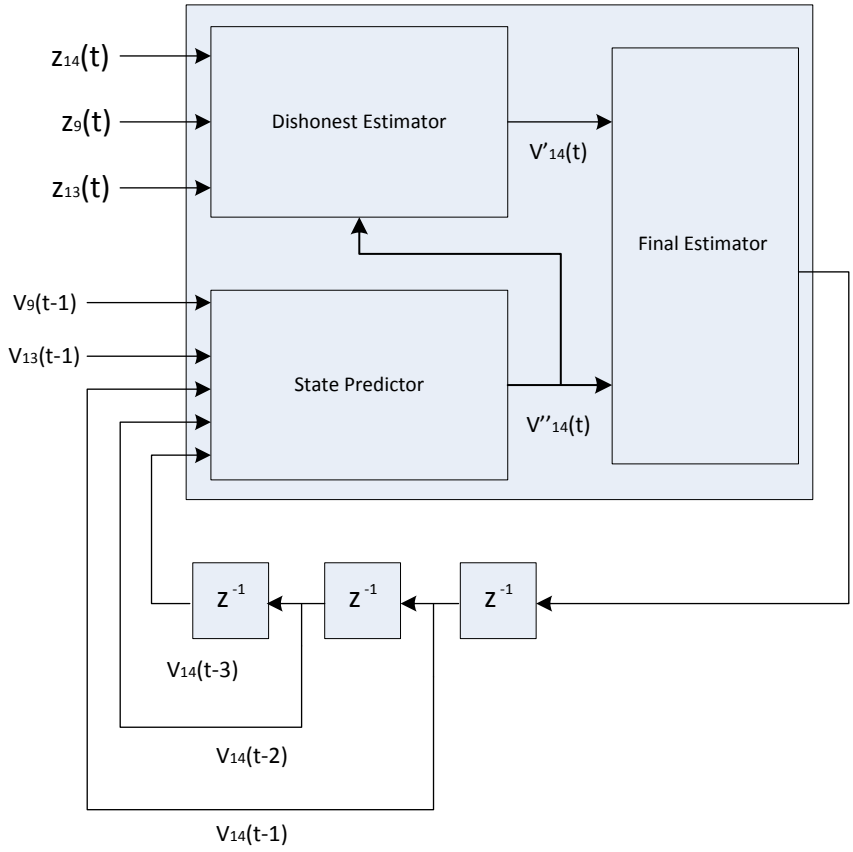

Figure 8.5: The flow of information of cell 14 is shown in details. The neighboring cells are 13, and 9.

8.2.1.2 Significance for Practical Systems

From Figs. 8.2, 8.4, and 8.5, it can be seen that the method is completely distributed to the cell level. With the distributed nature of the estimator, the computational complexity reduces significantly and the privacy is maintained. Due to

this nature, the computation process runs faster than the PMU rate with the cellular dishonest method. It is expected that the method will be able to meet the needs of the real-time operations with the PMU data.

Another importance enhancement is the integration of the predictor in the estimation. It takes very less time compared to the dishonest unit. With the predicted values, it is possible to detect any large change in the system. For example, when there is a large disturbance in the system, the traditional estimator gives a result following the existing model of the system. But, the model looses its validity under the large disturbance. In this estimator, with two outputs from two different units, it is easy to detect any big change in the system.

8.2.2 Finalizing Unit

The final estimator units collect the estimation result from the dishonest units and the predicted values and it compares the values. If there is no significant difference between the two, then it is assumed that the system is running under a normal operating condition. With this case, a weighted average is taken of the two using the following equation,

$$x_{final} = \alpha x_{dishonest} + (1 - \alpha) x_{predicted} \tag{8.1}$$

After getting the final values, the finalizing units send them back to the prediction unit for handling the next sample. The output value of t is used in the prediction of $t + 1$, and $t + 2$.

In case, a significant difference is found between the two, then it is concluded that the system is undergoing a large disturbance, and the presented system model is no longer valid. Under this situation, the correct model needs to be formed based

on the nature of the disturbance. The estimator hands the job over to fault analysis. This is beyond the scope of the dissertation.

8.3 Simulation Results

The simulation results are shown for IEEE 68-bus NY-NE test system. The value of α is varied from 0.4 to 1.0. The governors and the excitation systems of all generators are disturbed with Pseudo-Random Binary Signals (PRBS) to emulate the random changes of the load. Both the magnitudes and the frequencies of the PRBS signals are varied randomly using the method delineated in Fig. 8.6. A part of the PSBS signals are shown in Fig. 8.7.

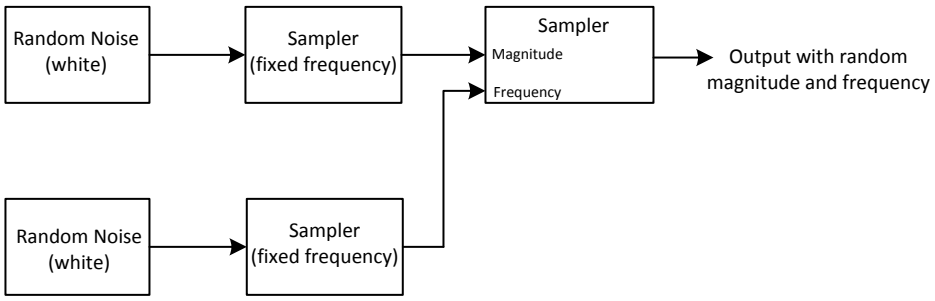

Figure 8.6: The PRBS signal generator. Both the magnitude and the frequency inputs of the sampler is fed with two random generators.

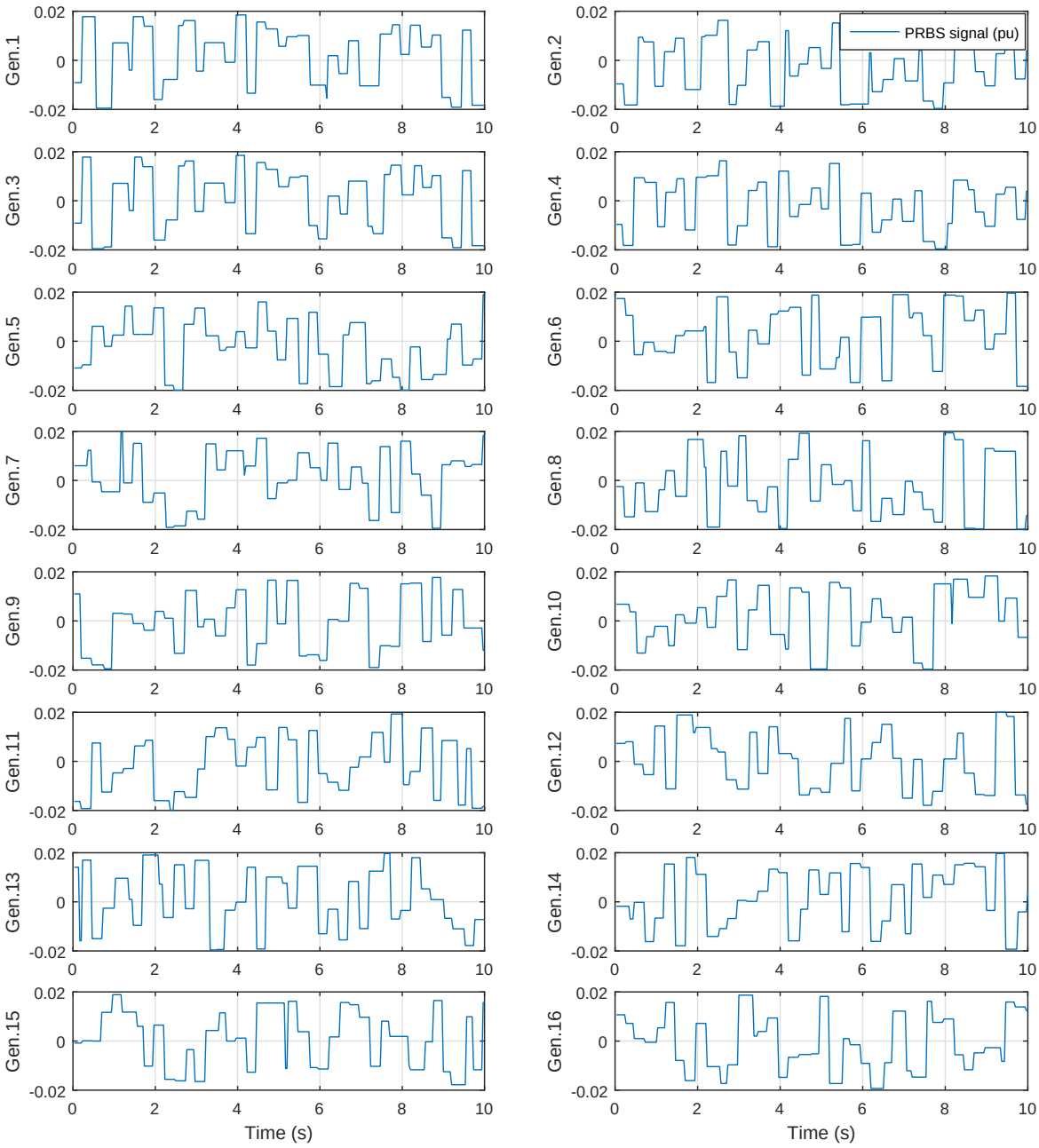

Figure 8.7: The PRBS signals with random magnitudes and frequencies that are injected at the 16 generators of the 68-bus system.

There is a specific difference between the PRBS signals of Chapter 7 and the signals used here. As can be seen from Fig. 7.4, the frequency of the PRBS signals are constant which may create a specific frequency of the voltage magnitude and the phase angle. Having a specific frequency can make it easier to predict. As the loads can be completely random, the states are expected to change with changing frequency. To emulate the randomness, the frequencies are made random in this chapter.

8.3.1 Accuracy

The accuracy of the estimator are shown for six buses (bus 2, 8, 21, 32, 51, 60) with $\alpha = 0.7$ in Fig. 8.8. The left column shows the angles and the right column shows the magnitudes. Except a few samples, the overall estimation result is quite satisfactory. It proves that the estimation can be executed at the cell level.

8.3.2 Smoothness

In order to measure smoothness of the output, a metric is defined in this dissertation based on the definition of the total deviation described in [66]. It is the mean change of the estimated values over one second of data. If the PMU rate is N_{pmu} samples/second in the process, the smoothness for state i is defined as,

$$Smoothness = [\frac{1}{N_{pmu} - 1} \sum_{t=2}^{N_{pmu}} |(\hat{x}_i(t) - \hat{x}_i(t-1))|]^{-1} \tag{8.2}$$

8.3.3 Required Time

The run-time is an influential factor behind the distributed method. As the distribution is made to the cell level, the required time is also low. The experiments are run on an Intel(R) Xeon(R) CPU (E5-2609) with 2.4 GHz core and 48GB of

memory. The average required time by the whole unit is around 1.8955 ms which is much faster than the PMU data collection rate. The dishonest unit takes around 1.7122 ms, the prediction unit takes around 0.19 ms. The finalizing unit takes a negligible amount of time. The communication time is excluded in all these results.

8.3.4 Individual Contribution

The outputs of the dishonest unit, the predictor, and the final unit are shown for bus 25 in Fig. 8.9, and 8.10. It is observable that the dishonest unit runs closer to the actual value. On the other hand, the predictor unit yields a smoother result.

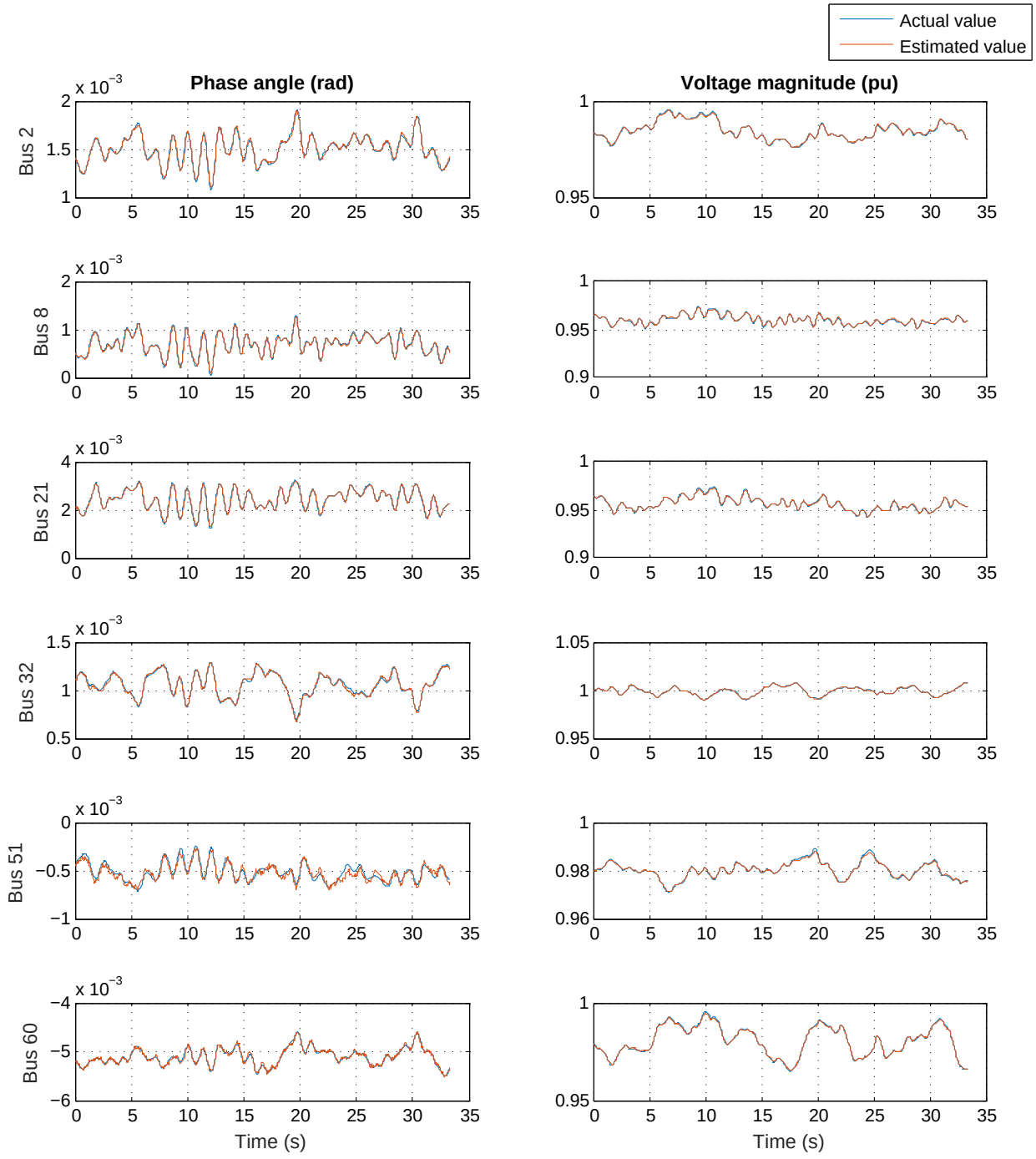

Figure 8.8: The actual and the estimated values of the voltage magnitude and anglers of six buses.

The relative performances are shown in Table 8.1.

Table 8.1: Performance of different parts of the distributed estimator.

Features	**Dishonest unit**		**Prediction unit**		**Over-all**	
	Mean	Std	Mean	Std	Mean	Std
Absolute error (θ)	2.37e-5	2.12e-5	3.39e-5	3.36e-5	2.42e-5	2.2e-5
Absolute error ($\|V\|$)	1.51e-4	1.5e-4	8.02e-4	9.39e-4	3.09e-4	3.29e-4
Time (ms)	1.7	0.102	0.19	0.096	1.9	0.159

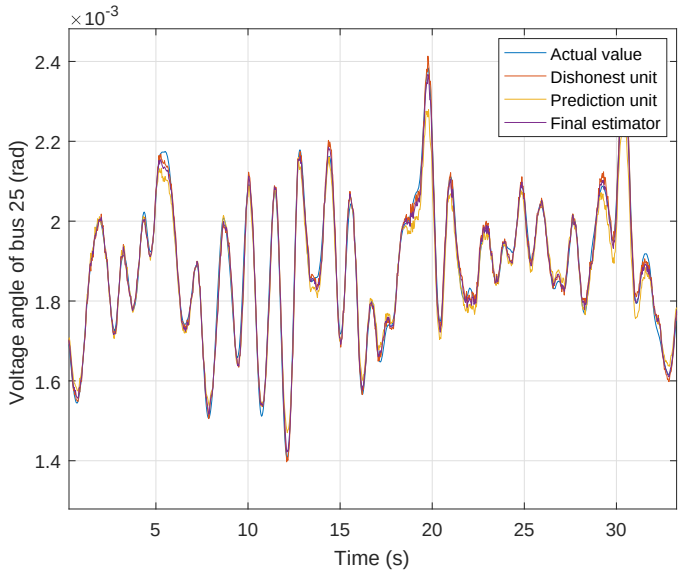

Figure 8.9: The outputs of different parts of the distributed dynamic estimator with the actual values of phase angle of bus 25.

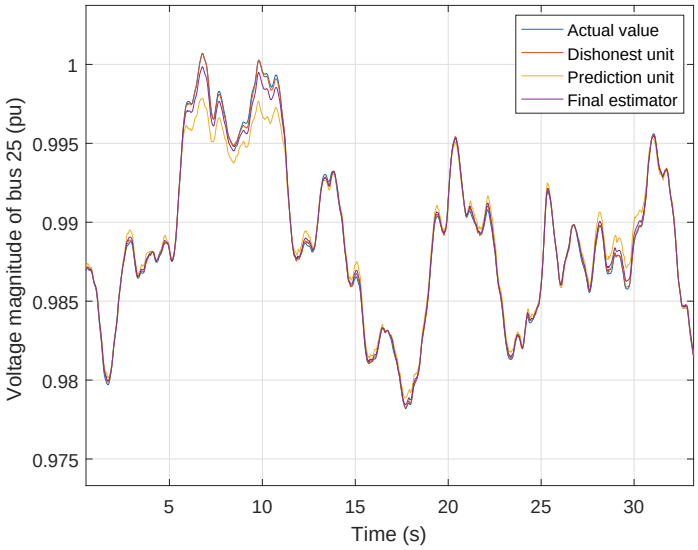

Figure 8.10: The outputs of different parts of the distributed dynamic estimator with the actual values of voltage magnitude of bus 25.

8.3.5 Effects of α

The effects of the weight, α on the estimation are shown for two different PMU rates in Figs. 8.11 and 8.12. From the figures it can be seen that with the increase of α, the estimated values go close to the actual values. Like the previous results, it also indicates that the output of the dishonest unit is more accurate than the prediction unit. The absolute error and the absolute percentage error for different values of α are summarized in Tables 8.2 - 8.9.

But, the predictor units can make the output smoother. The effect of the values of α on the smoothness can be seen from Tables 8.10 - 8.13. It can be seen that the mean smoothness decreases with α.

For detailed performance analysis, both the accuracy and the smoothness for one second of window are shown in Figs. 8.13 - 8.44. It is important to remember

that for the 30 Hz data rate, there are 30 samples in a window and for the 60 Hz, there are 60 samples.

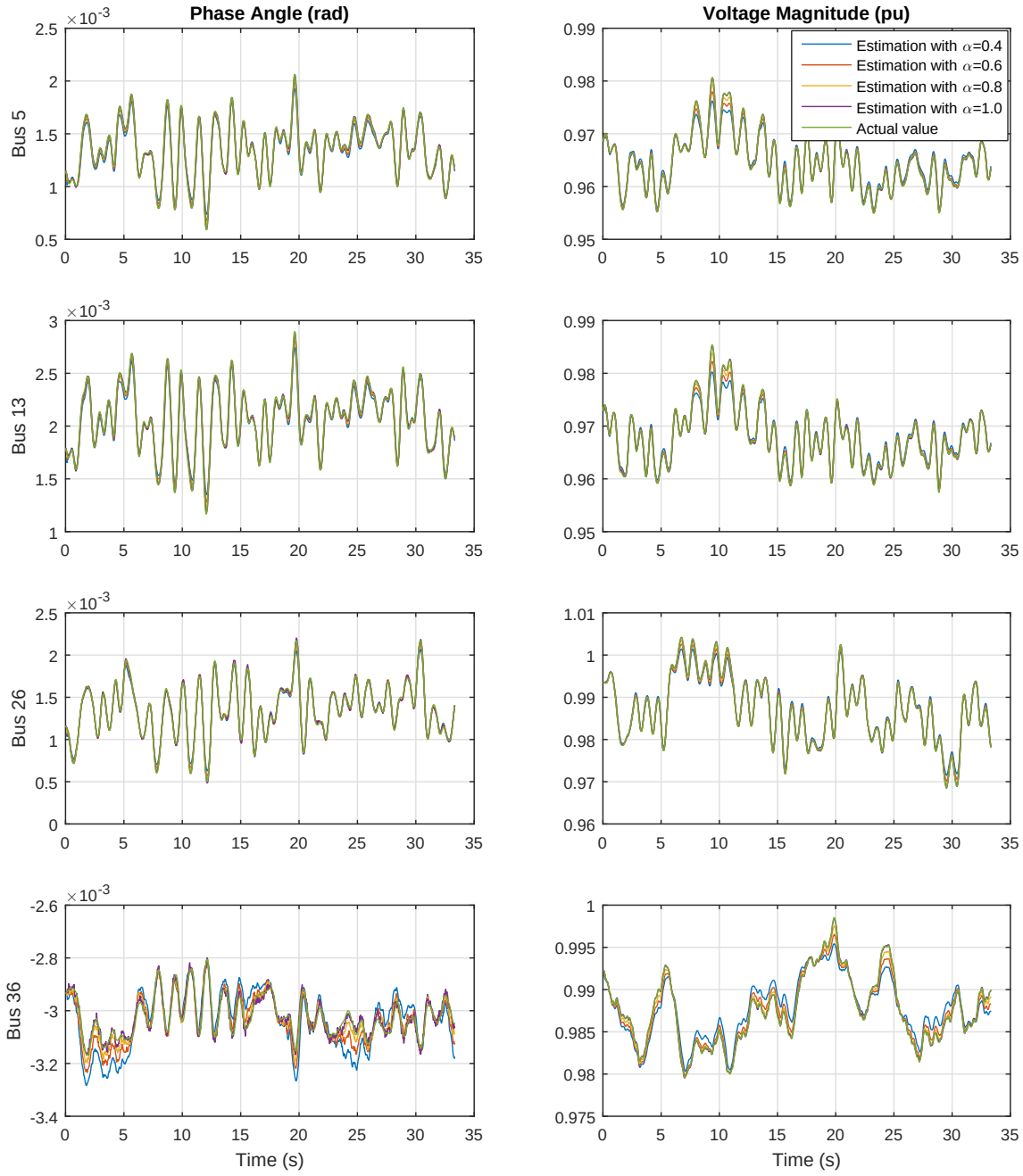

Figure 8.11: The actual and the estimated values of the voltage magnitude and anglers of four buses for different values of α for a PMU rate of 30 Hz.

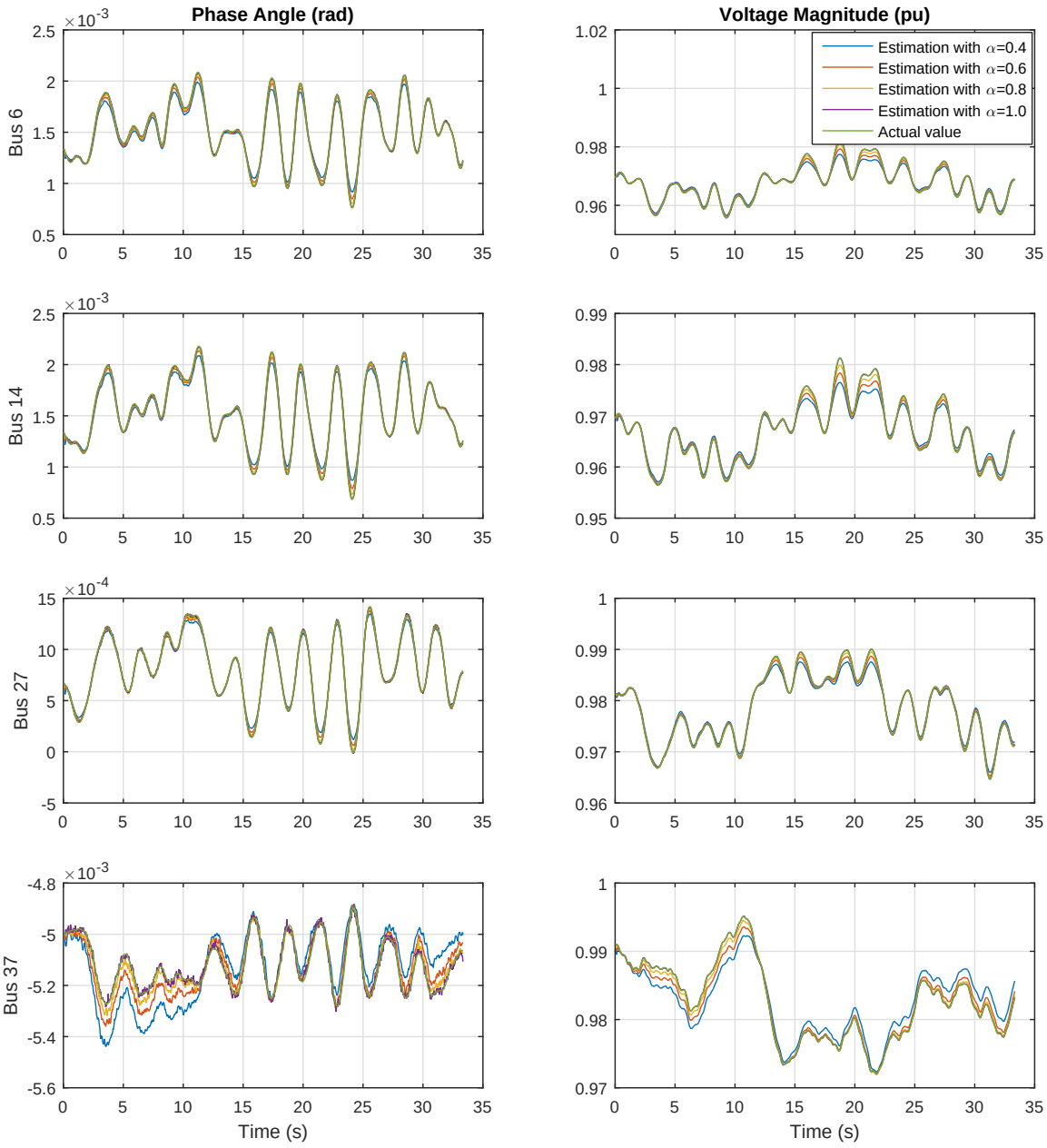

Figure 8.12: The actual and the estimated values of the voltage magnitude and anglers of four buses for different values of α for a PMU rate of 60 Hz.

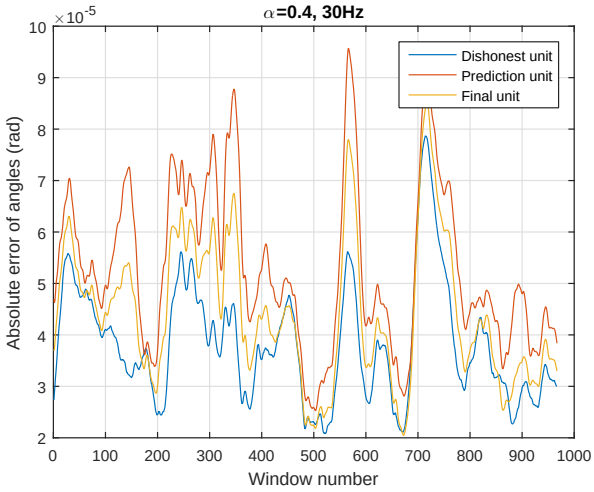

Figure 8.13: Absolute error of phase angle for $\alpha = 0.4$, and PMU rate 30 Hz.

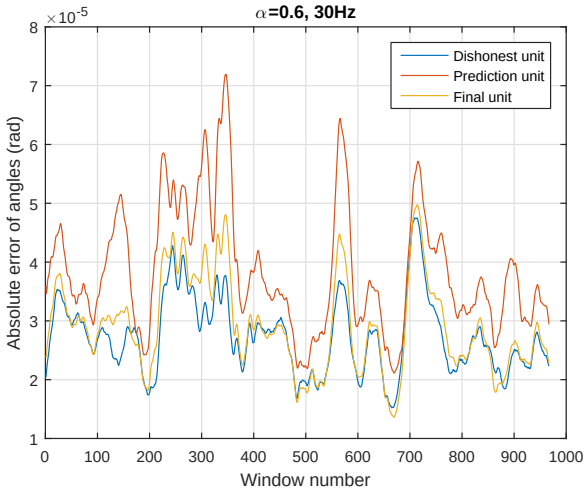

Figure 8.14: Absolute error of phase angle for $\alpha = 0.6$, and PMU rate 30 Hz.

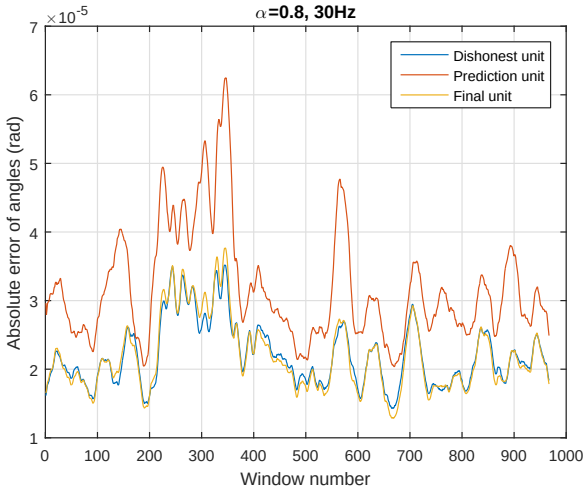

Figure 8.15: Absolute error of phase angle for $\alpha = 0.8$, and PMU rate 30 Hz.

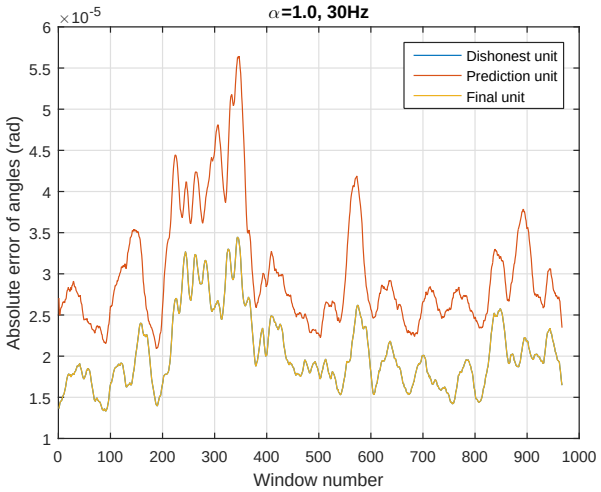

Figure 8.16: Absolute error of phase angle for $\alpha = 1.0$, and PMU rate 30 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

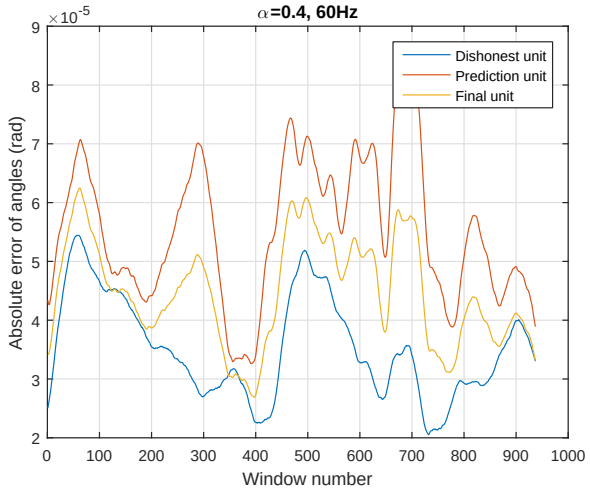

Figure 8.17: Absolute error of phase angle for $\alpha = 0.4$, and PMU rate 60 Hz.

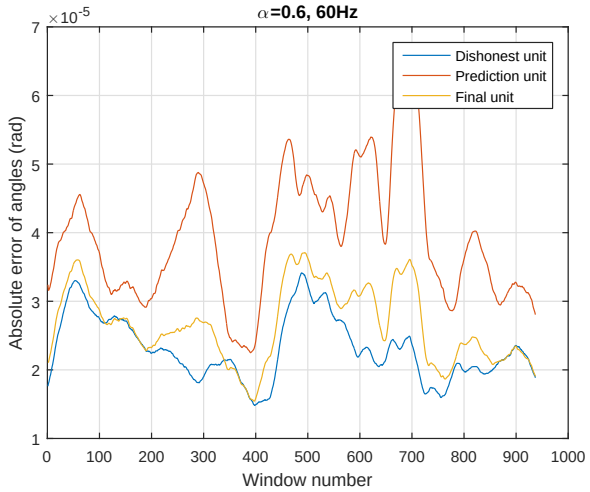

Figure 8.18: Absolute error of phase angle for $\alpha = 0.6$, and PMU rate 60 Hz.

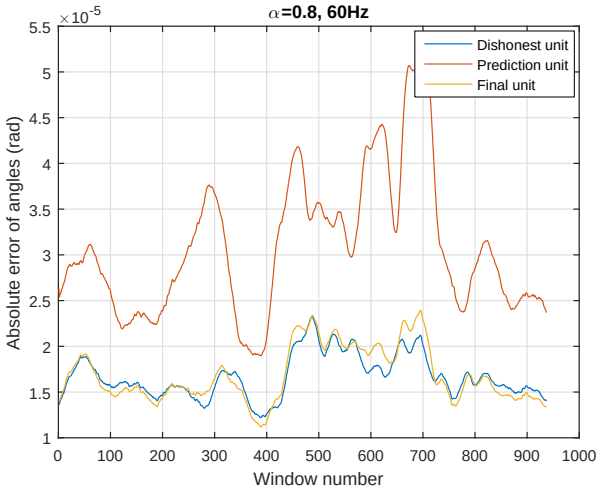

Figure 8.19: Absolute error of phase angle for $\alpha = 0.8$, and PMU rate 60 Hz.

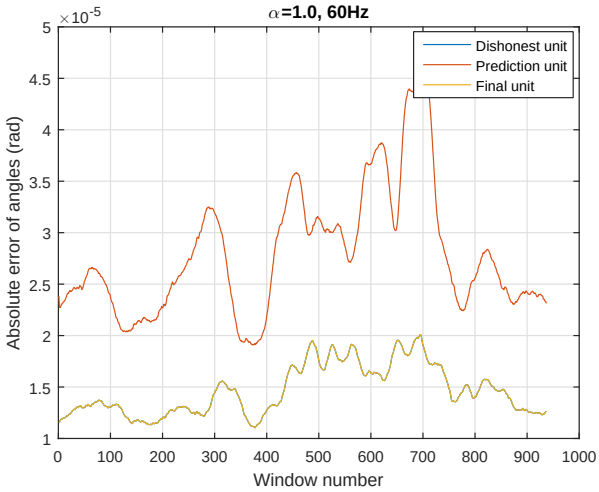

Figure 8.20: Absolute error of phase angle for $\alpha = 1.0$, and PMU rate 60 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

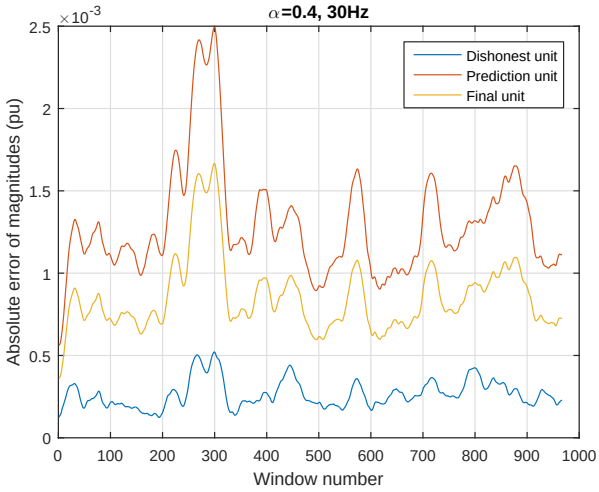

Figure 8.21: Absolute error of voltage magnitude for $\alpha = 0.4$, and PMU rate 30 Hz.

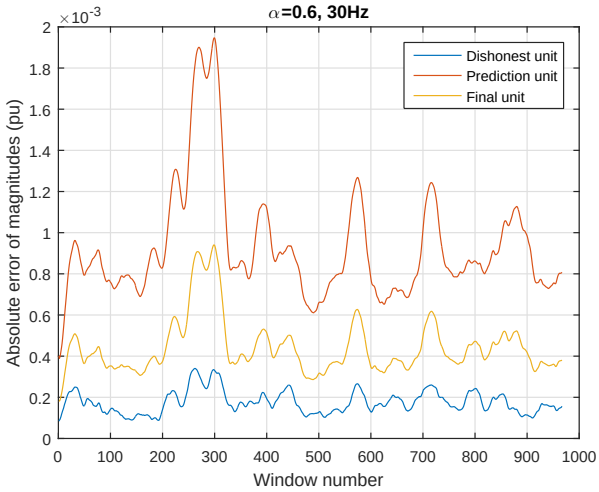

Figure 8.22: Absolute error of voltage magnitude for $\alpha = 0.6$, and PMU rate 30 Hz.

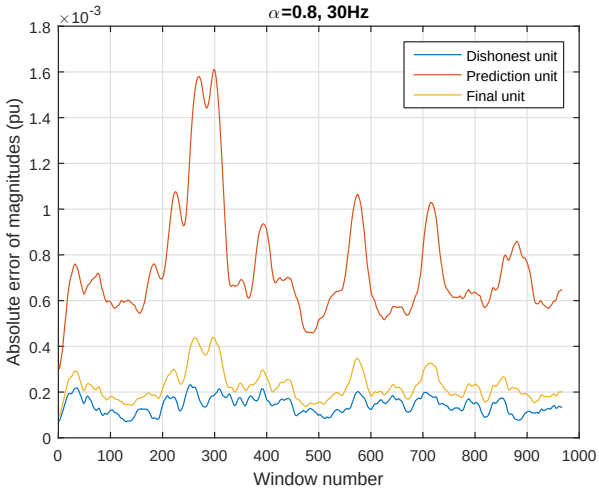

Figure 8.23: Absolute error of voltage magnitude for $\alpha = 0.8$, and PMU rate 30 Hz.

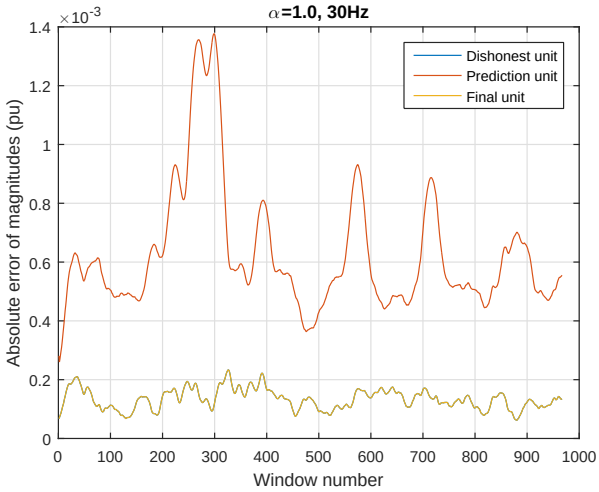

Figure 8.24: Absolute error of voltage magnitude for $\alpha = 1.0$, and PMU rate 30 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

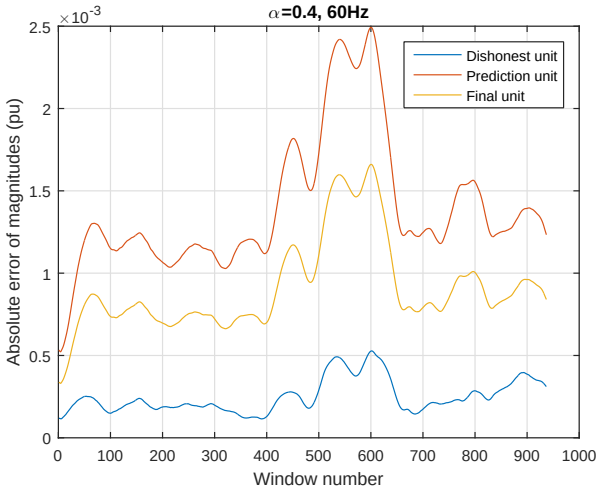

Figure 8.25: Absolute error of voltage magnitude for $\alpha = 0.4$, and PMU rate 60 Hz.

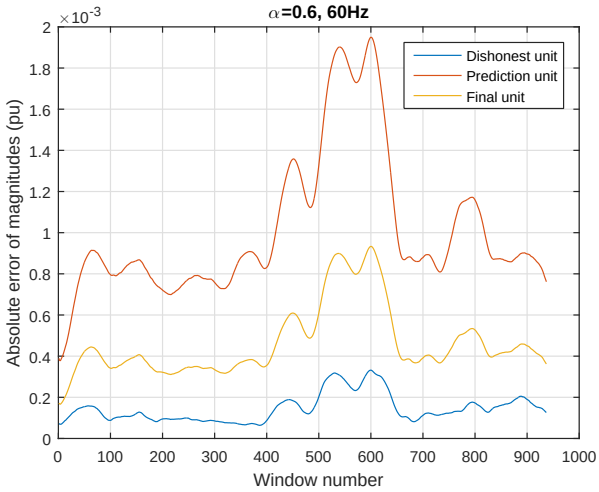

Figure 8.26: Absolute error of voltage magnitude for $\alpha = 0.6$, and PMU rate 60 Hz.

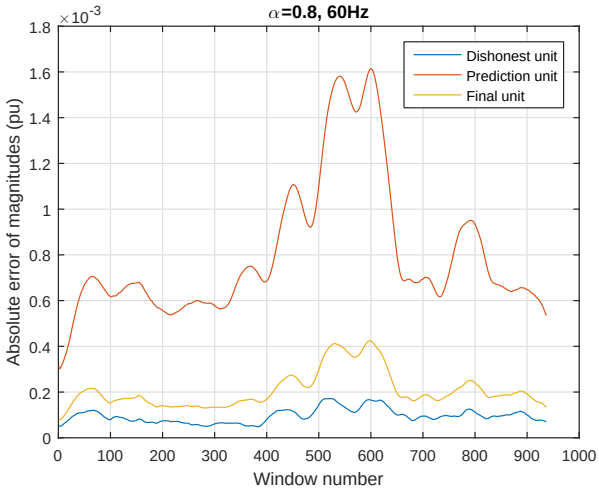

Figure 8.27: Absolute error of voltage magnitude for $\alpha = 0.8$, and PMU rate 60 Hz.

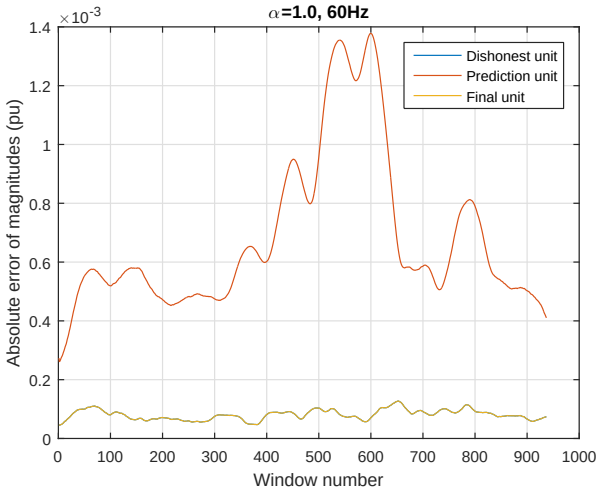

Figure 8.28: Absolute error of voltage magnitude for $\alpha = 1.0$, and PMU rate 60 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

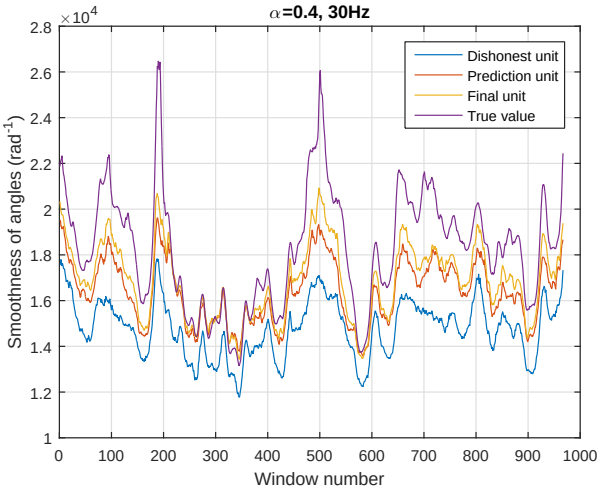

Figure 8.29: The smoothness of phase angle for $\alpha = 0.4$, and PMU rate 30 Hz.

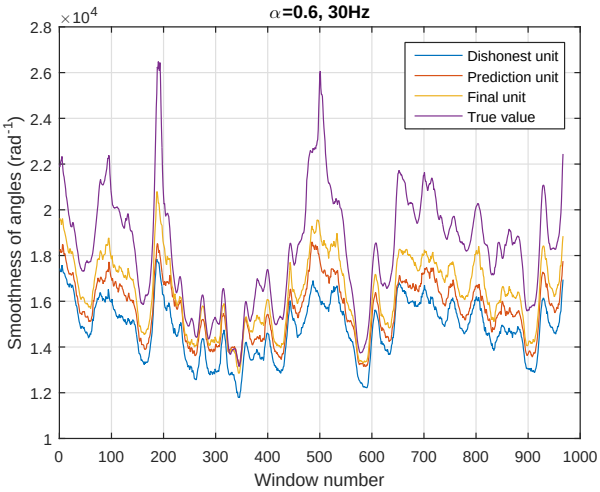

Figure 8.30: The smoothness of phase angle for $\alpha = 0.6$, and PMU rate 30 Hz.

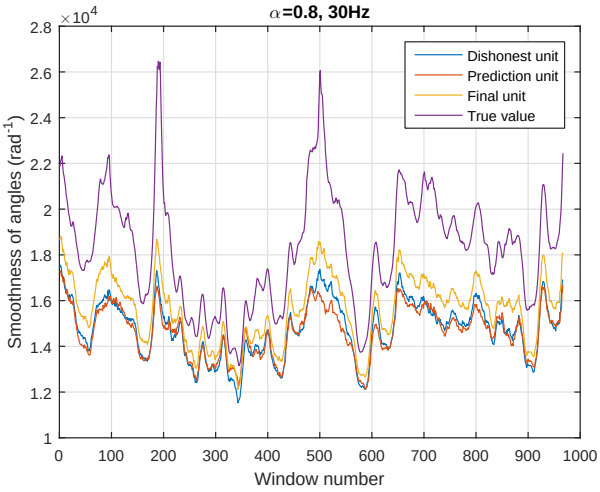

Figure 8.31: The smoothness of phase angle for $\alpha = 0.8$, and PMU rate 30 Hz.

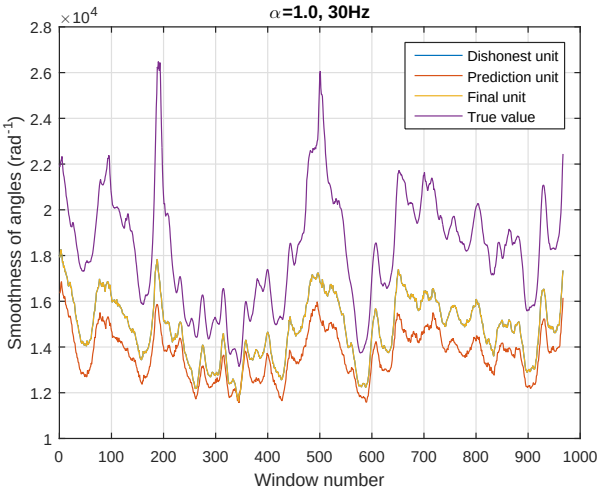

Figure 8.32: The smoothness of phase angle for $\alpha = 1.0$, and PMU rate 30 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

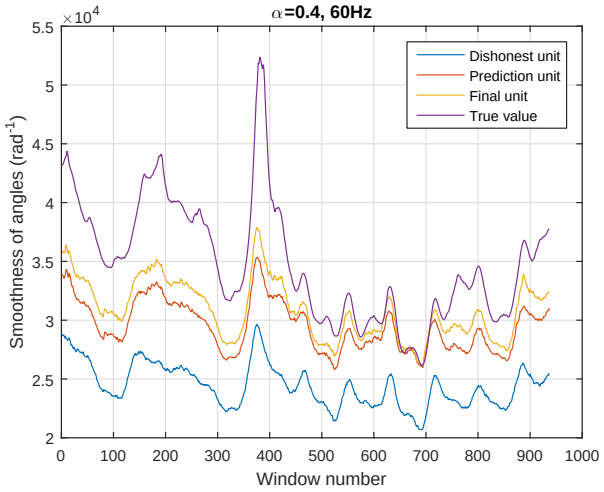

Figure 8.33: The smoothness of phase angle for $\alpha = 0.4$, and PMU rate 60 Hz.

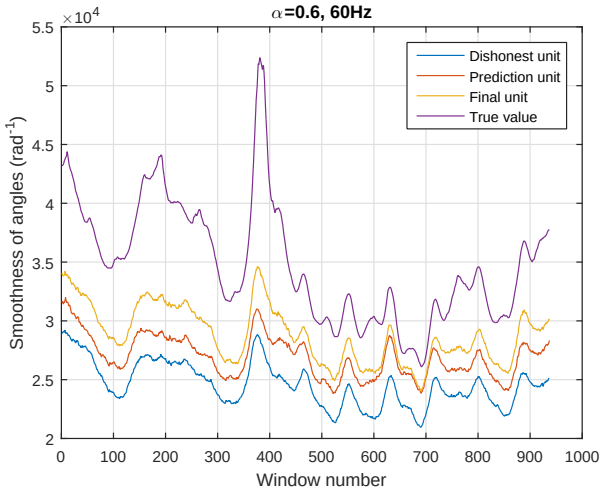

Figure 8.34: The smoothness of phase angle for $\alpha = 0.6$, and PMU rate 60 Hz.

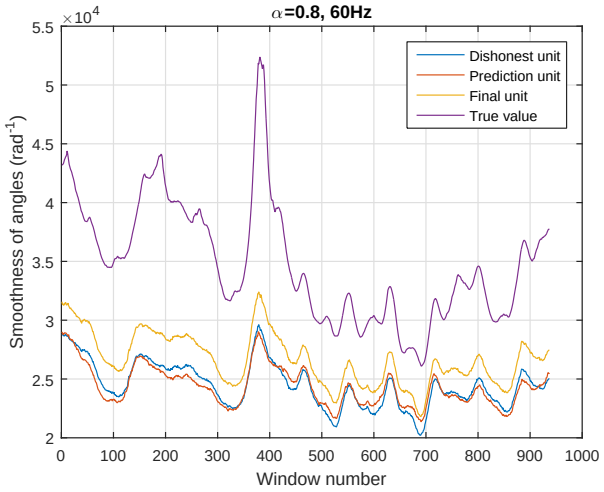

Figure 8.35: The smoothness of phase angle for $\alpha = 0.8$, and PMU rate 60 Hz.

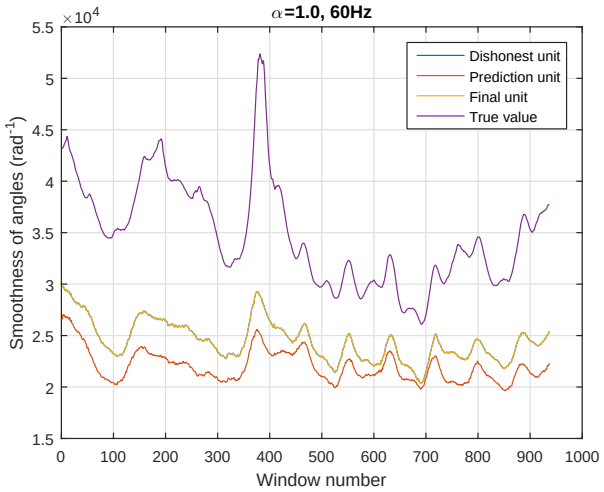

Figure 8.36: The smoothness of phase angle for $\alpha = 1.0$, and PMU rate 60 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

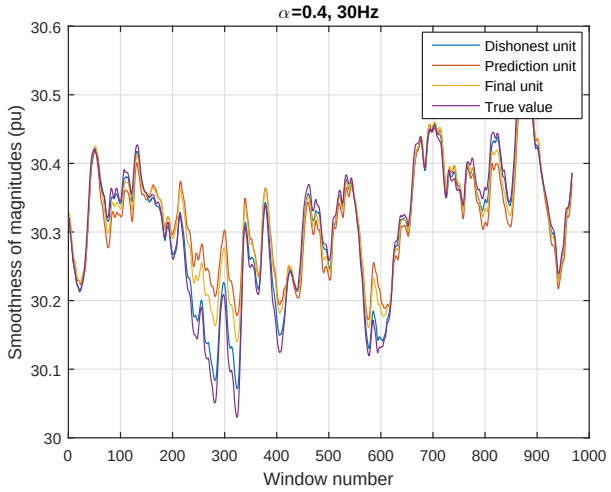

Figure 8.37: The smoothness of voltage magnitude for $\alpha = 0.4$, and PMU rate 30 Hz.

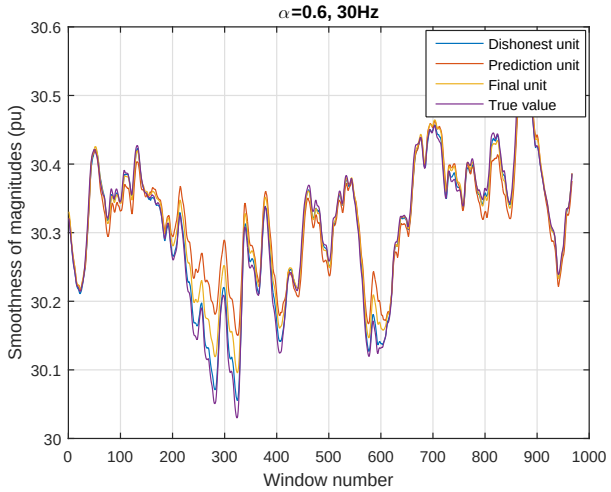

Figure 8.38: The smoothness of voltage magnitude for $\alpha = 0.6$, and PMU rate 30 Hz.

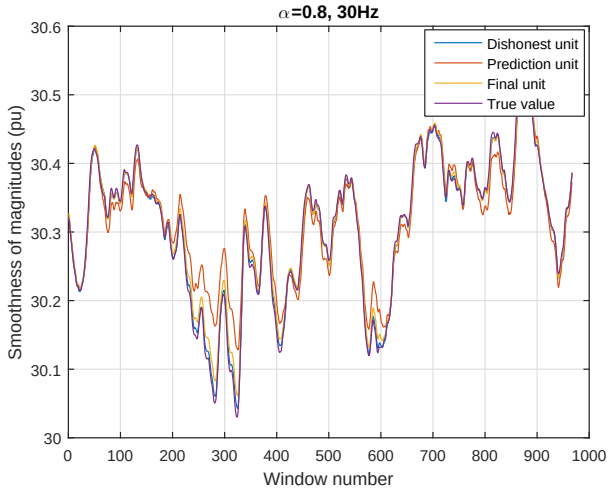

Figure 8.39: The smoothness of voltage magnitude for $\alpha = 0.8$, and PMU rate 30 Hz.

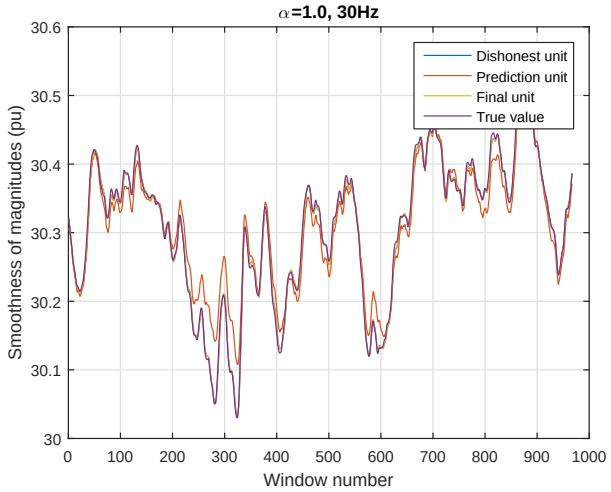

Figure 8.40: The smoothness of voltage magnitude for $\alpha = 1.0$, and PMU rate 30 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

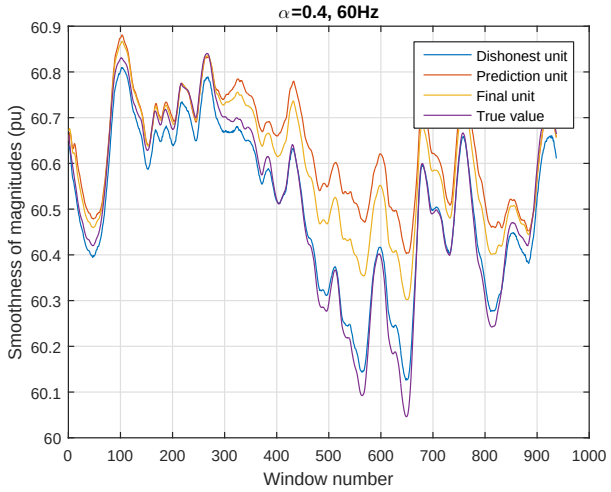

Figure 8.41: The smoothness of voltage magnitude for $\alpha = 0.4$, and PMU rate 60 Hz.

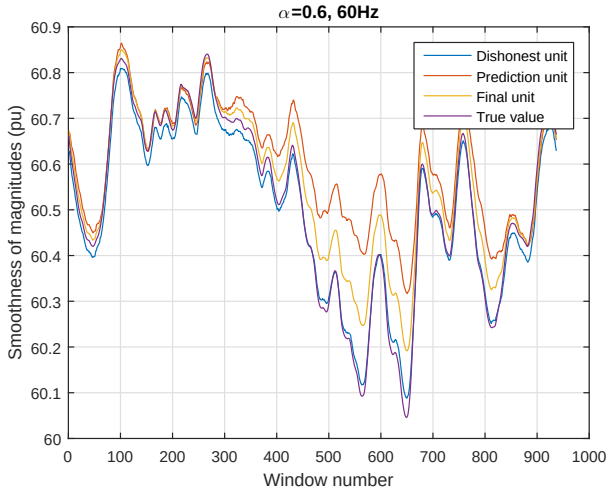

Figure 8.42: The smoothness of voltage magnitude for $\alpha = 0.6$, and PMU rate 60 Hz.

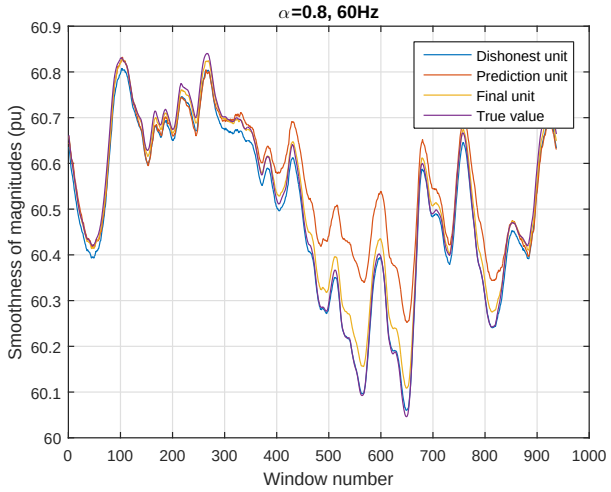

Figure 8.43: The smoothness of voltage magnitude for $\alpha = 0.8$, and PMU rate 60 Hz.

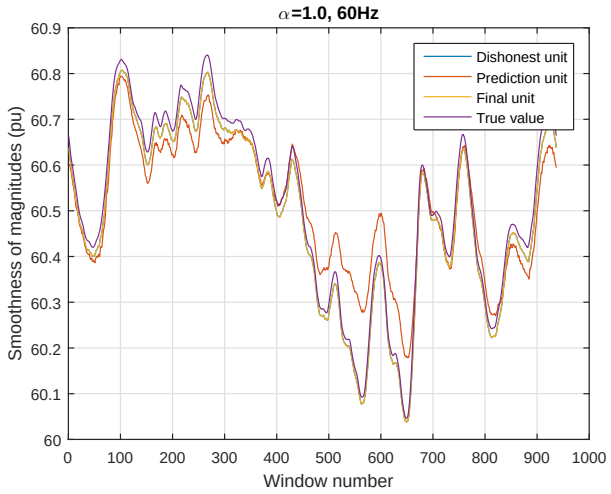

Figure 8.44: The smoothness of voltage magnitude for $\alpha = 1.0$, and PMU rate 60 Hz. As $\alpha = 1.0$, the output of the dishonest unit is the same as the final unit and they completely overlap.

8.3.6 Analysis of the Results

The simulation results for different values of α can be grouped for phase angles and voltage magnitudes, and for accuracy and smoothness. All four combinations are

discussed below,

- Accuracy of phase angles: This part is related to Tables 8.2, 8.3, 8.6, 8.7 and Figs. 8.13 - 8.20. For almost all the values of α it can be seen that the prediction units have the maximum errors, and the dishonest unit have the minimum errors. Being an average, the final unit has a value in between these two for most of the cases. In a few cases it has less error compared to both the dishonest and the prediction units. This happens when the output of the two stays on two opposite sides of the true value and their weighted average is closer.

- Accuracy of voltage magnitudes: From the Tables 8.4, 8.5, 8.8, 8.9 and Figs. 8.21 - 8.28, it can be seen that the dishonest units have the minimum error and the prediction units have the maximum. For all cases, the final units have a value in between the two. Unlike phase angles, there is no exceptional improvements in voltage magnitudes.

- Smoothness of phase angles: The results are summarized in Tables 8.10 - 8.11, and in Figs. 8.29 - 8.36. From the tables, it can be easily seen that the smoothness of the final output increases with the value of α. It proves that with the increase of the contribution of the prediction unit, the smoothness increases.

- Smoothness pf voltage magnitudes: Similar trends are found for voltage magnitudes from Tables 8.12, 8.13 and Figs. 8.37 - 8.44. It can be noted that the improvement is lower for voltage magnitudes compared to phase angles. It is due to the different variation of the two types.

8.3.7 Necessity of Cooperation

From the simulation results, it can be seen that the estimator as a whole pursuits the actual value quite well. In a few cases, the prediction units deviate a bit, but it gets back to the track with the help of the dishonest units. On the other hand, the dishonest units which have rough outputs get smoothed by the prediction units. Thus these units help each other to improve the final output.

8.3.8 Distributability

It can be noted that, the whole system runs as cells on each bus. The dishonest, and the predictor units are clearly visible as cells. The final units require the input from only the corresponding bus's dishonest and prediction units. These do not even require the neighbors' outputs. This preserves the distributability of the proposed estimator.

8.3.9 Importance of the Predictor

From Tables 8.2 and 8.9, it can be seen that the errors get reduced with the increase of α. The maximum possible value of α is 1.0. With this value, the weighted average reduces to the following equation and the output of the dishonest unit becomes the final output.

$$x_{final} = x_{dishonest} \tag{8.3}$$

However, the predictor unit has some unique contributions that made its place in the estimation process,

- With the predicted values, it is possible to compare the output of the dishonest

estimator to detect any large disturbance in the system. Under that, the system model becomes invalid and the estimation results lose their significance.

- Though the normalization of the power flows of each cell can be done with previous estimated values, it is better to use the predicted values.
- The outputs of the dishonest method is not smooth and it usually reflects the effects of the errors. With the integration of the predictors, it gets smoother.

8.4 Summary

Due to the dynamic nature of the system, it is desired to have a dynamic estimator for the state estimation. The existing WLS estimator is a static estimator, and it cannot reduce the roughness of the result. Though the Kalman filter based dynamic estimator can reduce the roughness, it is still not distributed to the cell level. The proposed work combines the concept of the dynamic estimator with the distributed systems at the cell level.

Table 8.2: Mean and standard deviation of the absolute errors of the phase angles for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	3.83e-05	1.15e-05	3.53e-05	8.53e-06	5.31e-05	1.57e-05	5.55e-05	1.19e-05	4.44e-05	1.38e-05	4.48e-05	8.95e-06
0.6	2.74e-05	6.43e-06	2.31e-05	4.61e-06	3.87e-05	1.06e-05	3.93e-05	9.27e-06	2.92e-05	7.86e-06	2.67e-05	5.43e-06
0.8	2.20e-05	4.58e-06	1.67e-05	2.36e-06	3.19e-05	8.25e-06	3.09e-05	7.68e-06	2.19e-05	5.10e-06	1.69e-05	2.98e-06
1.0	2.04e-05	4.56e-06	1.46e-05	2.42e-06	3.02e-05	7.02e-06	2.79e-05	6.20e-06	2.04e-05	4.56e-06	1.46e-05	2.42e-06

Table 8.3: Maximum and minimum of the absolute errors of the phase angles for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
0.4	7.87e-05	2.08e-05	5.44e-05	2.05e-05	9.57e-05	2.53e-05	8.20e-05	3.26e-05	8.51e-05	2.05e-05	6.25e-05	2.69e-05
0.6	4.75e-05	1.52e-05	3.41e-05	1.48e-05	7.20e-05	2.11e-05	6.20e-05	2.25e-05	4.98e-05	1.36e-05	3.71e-05	1.54e-05
0.8	3.52e-05	1.43e-05	2.33e-05	1.22e-05	6.25e-05	2.04e-05	5.20e-05	1.90e-05	3.77e-05	1.28e-05	2.39e-05	1.12e-05
1.0	3.45e-05	1.33e-05	2.01e-05	1.10e-05	5.64e-05	2.09e-05	4.59e-05	1.91e-05	3.45e-05	1.33e-05	2.01e-05	1.10e-05

Table 8.4: Mean and standard deviation of the absolute errors of the voltage magnitudes for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	2.65e-04	8.35e-05	2.51e-04	1.01e-04	1.30e-03	3.36e-04	1.40e-03	4.12e-04	8.57e-04	2.24e-04	9.13e-04	2.75e-04
0.6	1.76e-04	5.50e-05	1.45e-04	6.74e-05	9.32e-04	2.81e-04	1.01e-03	3.51e-04	4.43e-04	1.35e-04	4.64e-04	1.70e-04
0.8	1.42e-04	3.57e-05	9.55e-05	3.04e-05	7.40e-04	2.45e-04	8.08e-04	3.06e-04	2.26e-04	6.51e-05	2.08e-04	7.98e-05
1.0	1.34e-04	3.42e-05	8.07e-05	1.70e-05	6.27e-04	2.16e-04	6.82e-04	2.68e-04	1.34e-04	3.42e-05	8.07e-05	1.70e-05

Table 8.5: Maximum and minimum of the absolute errors of the voltage magnitudes for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
0.4	5.21e-04	1.23e-04	5.28e-04	1.15e-04	2.49e-03	5.62e-04	2.49e-03	5.22e-04	1.67e-03	3.62e-04	1.66e-03	3.31e-04
0.6	3.40e-04	8.38e-05	3.32e-04	6.39e-05	1.95e-03	3.89e-04	1.95e-03	3.78e-04	9.41e-04	1.81e-04	9.33e-04	1.67e-04
0.8	2.32e-04	7.03e-05	1.71e-04	4.83e-05	1.61e-03	3.02e-04	1.61e-03	3.01e-04	4.39e-04	9.62e-05	4.24e-04	7.92e-05
1.0	2.33e-04	6.21e-05	1.27e-04	4.56e-05	1.38e-03	2.60e-04	1.38e-03	2.61e-04	2.33e-04	6.21e-05	1.27e-04	4.56e-05

Table 8.6: Mean and standard deviation of the absolute percentage errors of the phase angles for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	3.13e+00	1.34e+00	2.91e+00	1.25e+00	3.13e+00	1.34e+00	2.91e+00	1.25e+00	3.13e+00	1.34e+00	2.91e+00	1.25e+00 *
0.6	2.11e+00	7.36e-01	1.82e+00	6.71e-01	2.11e+00	7.36e-01	1.82e+00	6.71e-01	2.11e+00	7.36e-01	1.82e+00	6.71e-01
0.8	1.59e+00	5.62e-01	1.18e+00	3.39e-01	1.59e+00	5.62e-01	1.18e+00	3.39e-01	1.59e+00	5.62e-01	1.18e+00	3.39e-01
1.0	1.46e+00	5.73e-01	9.38e-01	2.70e-01	1.46e+00	5.73e-01	9.38e-01	2.70e-01	1.46e+00	5.73e-01	9.38e-01	2.70e-01

The values of θ which are close to 0 are excluded for this table.

Table 8.7: Maximum and minimum of the absolute percentage errors of the phase angles for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
0.4	∞	1.22e+00	∞	1.12e+00	∞	1.59e+00	∞	2.06e+00	∞	1.14e+00	∞	1.60e+00
0.6	∞	8.53e-01	∞	8.69e-01	∞	1.24e+00	∞	1.51e+00	∞	7.14e-01	∞	9.71e-01
0.8	∞	7.52e-01	∞	6.76e-01	∞	1.13e+00	∞	1.29e+00	∞	6.63e-01	∞	7.21e-01
1.0	∞	7.65e-01	∞	6.22e-01	∞	1.29e+00	∞	1.21e+00	∞	7.65e-01	∞	6.22e-01

Table 8.8: Mean and standard deviation of the absolute percentage errors of the voltage magnitudes for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	2.72e-02	8.53e-03	2.57e-02	1.03e-02	1.33e-01	3.41e-02	1.43e-01	4.16e-02	8.76e-02	2.27e-02	9.31e-02	2.78e-02
0.6	1.81e-02	5.61e-03	1.49e-02	6.85e-03	9.51e-02	2.84e-02	1.03e-01	3.56e-02	4.52e-02	1.37e-02	4.73e-02	1.73e-02
0.8	1.46e-02	3.66e-03	9.78e-03	3.08e-03	7.55e-02	2.48e-02	8.23e-02	3.10e-02	2.31e-02	6.60e-03	2.12e-02	8.10e-03
1.0	1.37e-02	3.51e-03	8.28e-03	1.74e-03	6.40e-02	2.18e-02	6.94e-02	2.72e-02	1.37e-02	3.51e-03	8.28e-03	1.74e-03

Table 8.9: Maximum and minimum of the absolute percentage errors of the voltage magnitudes for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
0.4	5.31e-02	1.26e-02	5.37e-02	1.18e-02	2.53e-01	5.74e-02	2.53e-01	5.33e-02	1.69e-01	3.70e-02	1.69e-01	3.38e-02
0.6	3.46e-02	8.59e-03	3.38e-02	6.55e-03	1.98e-01	3.97e-02	1.98e-01	3.86e-02	9.56e-02	1.86e-02	9.49e-02	1.71e-02
0.8	2.37e-02	7.23e-03	1.75e-02	4.95e-03	1.64e-01	3.09e-02	1.64e-01	3.08e-02	4.46e-02	9.85e-03	4.31e-02	8.10e-03
1.0	2.39e-02	6.40e-03	1.30e-02	4.66e-03	1.40e-01	2.66e-02	1.40e-01	2.66e-02	2.39e-02	6.40e-03	1.30e-02	4.66e-03

Table 8.10: Mean and standard deviation of the smoothness of the phase angles for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	1.48e+04	1.24e+03	2.46e+04	1.90e+03	1.64e+04	1.38e+03	2.95e+04	2.12e+03	1.69e+04	1.64e+03	3.09e+04	2.54e+03
0.6	1.48e+04	1.27e+03	2.45e+04	1.96e+03	1.56e+04	1.25e+03	2.69e+04	1.97e+03	1.64e+04	1.61e+03	2.88e+04	2.52e+03
0.8	1.49e+04	1.29e+03	2.45e+04	1.99e+03	1.47e+04	1.12e+03	2.44e+04	1.74e+03	1.57e+04	1.48e+03	2.67e+04	2.27e+03
1.0	1.49e+04	1.30e+03	2.44e+04	2.00e+03	1.37e+04	9.74e+02	2.20e+04	1.51e+03	1.49e+04	1.30e+03	2.44e+04	2.00e+03

Table 8.11: Maximum and minimum of the smoothness of the phase angles for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
0.4	1.78e+04	1.18e+04	2.96e+04	2.02e+04	1.96e+04	1.36e+04	3.54e+04	2.58e+04	2.09e+04	1.34e+04	3.71e+04	2.57e+04
0.6	1.81e+04	1.18e+04	2.97e+04	2.01e+04	1.86e+04	1.31e+04	3.24e+04	2.34e+04	2.09e+04	1.30e+04	3.50e+04	2.36e+04
0.8	1.81e+04	1.18e+04	2.97e+04	2.01e+04	1.79e+04	1.24e+04	2.97e+04	2.13e+04	1.97e+04	1.24e+04	3.24e+04	2.17e+04
1.0	1.83e+04	1.18e+04	2.97e+04	2.00e+04	1.71e+04	1.17e+04	2.69e+04	1.95e+04	1.83e+04	1.18e+04	2.97e+04	2.00e+04

Table 8.12: Mean and standard deviation of the smoothness of the voltage magnitudes for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.4	3.03e+01	9.46e-02	6.05e+01	1.70e-01	3.03e+01	7.27e-02	6.06e+01	1.15e-01	3.03e+01	8.03e-02	6.06e+01	1.35e-01
0.6	3.03e+01	9.83e-02	6.05e+01	1.80e-01	3.03e+01	7.93e-02	6.06e+01	1.28e-01	3.03e+01	9.02e-02	6.06e+01	1.59e-01
0.8	3.03e+01	1.01e-01	6.05e+01	1.87e-01	3.03e+01	8.30e-02	6.06e+01	1.36e-01	3.03e+01	9.72e-02	6.05e+01	1.78e-01
1.0	3.03e+01	1.03e-01	6.05e+01	1.93e-01	3.03e+01	8.55e-02	6.05e+01	1.41e-01	3.03e+01	1.03e-01	6.05e+01	1.93e-01

Table 8.13: Maximum and minimum of the smoothness of the voltage magnitudes for different values of α

Value of α	**Dishonest**				**Predictor**				**Final**			
	30 Hz		**60 Hz**		**30 Hz**		**60 Hz**		**30 Hz**		**60 Hz**	
	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
0.4	3.05e+01	3.01e+01	6.08e+01	6.01e+01	3.05e+01	3.02e+01	6.09e+01	6.04e+01	3.05e+01	3.01e+01	6.09e+01	6.03e+01
0.6	3.05e+01	3.01e+01	6.08e+01	6.01e+01	3.05e+01	3.02e+01	6.09e+01	6.03e+01	3.05e+01	3.01e+01	6.08e+01	6.02e+01
0.8	3.05e+01	3.00e+01	6.08e+01	6.01e+01	3.05e+01	3.01e+01	6.08e+01	6.03e+01	3.05e+01	3.01e+01	6.08e+01	6.01e+01
1.0	3.05e+01	3.00e+01	6.08e+01	6.00e+01	3.05e+01	3.01e+01	6.08e+01	6.02e+01	3.05e+01	3.00e+01	6.08e+01	6.00e+01

Chapter 9

Conclusion

9.1 Introduction

State estimation is considered to be an essential part of power system. The traditional estimators are not suitable for the upcoming smart grid operations. A fast and distributed estimator is needed with the exploitation of dynamic nature. It should be able to use the data collected by the PMUs effectively and it will help in avoiding big interruptions in the operation.

9.2 Summary of Research

In the first part of this dissertation, a fast and scalable estimator is developed with dishonest Gauss Newton method. Using the constant slope for the Jacobian, it runs very fast to meet the demand of the speed. Though, with current method, the Jacobian needs to be updated every time the network topology changes, future research may remove the necessity for minor changes.

The second part shows two CCN based distributed estimator - a static and

a semi-dynamic hybrid estimator. The static estimator is a layer-based sequential approach and the required time increases with the number of layers. On the other hand, the semi-dynamic method runs all cells simultaneously and keeps a contant time. With the help of GA, the semi-dynamic hybrid estimator gives an acceptable accuracy. Though CCN is fully distributed, the hybrid one needs a centralized processing for evaluating the fitness function. But it does not violet the privacy of the deregulated market as the fitness function is distributable. Thus the complete CCN-GA can be considered to be distributed.

However, due to the slow nature of the heuristic methods, it does not serve the purpose of fast estimation. To keep the speed, the dishonest method is integrated with the CCN framework. The power flows of the dishonest method are normalized with a predictor. The predictor is also based on the CCN framework to keep the distributability of the process. In a separate chapter of the dissertation, it is shown that the cellular Elman Recurrent Neural Network (ERNN) works quite well for the single step ahead prediction.

In the end of the dissertation, the CCN based dishonest method is presented as a distributed solution to power system dynamic state estimation. This should be able to meet most practical requirements of the future smart grid. From the simulation results on the 68-bus and 118-bus test system, it is expected that the planned estimator will work efficiently in real power systems.

9.3 Values for Practical Systems

The proposed distributed dynamic method has three major advantages as mentioned in Chapter 1. Each of them is important for the future smart grid.

- Privacy of information: In the deregulated energy market, the participants do

not want to share their information with others. It is very important for policy making and it makes a distributed method inevitable. The method developed in this dissertation is distributed to the smallest cells that demonstrates the utmost level of security of information.

- Speed: Being fully distributed, the method runs faster than the PMU rate of data collection. As the modern power systems are growing fast to cover new geographical areas, the computational load is also increasing. The method can distribute the load to small cells and keep an expected rate of throughput.

- Reduction of losses: Large disturbances are not very common in power systems. But, these can cause a huge amount of losses. In most of the cases, these do not occur instantaneously. Detecting the disturbance in the beginning can reduce the losses significantly. This is made possible in this estimator with the predictors.

9.4 Future Work

A series of improvements can be incorporated in the existing estimation methods and it can be extended in a number of ways. The prediction can be integrated with the prediction stage of the Kalman filter. Instead of using the dishonest method, the Kalman filter can be set as the computation method with the CCN framework. Thus a cellular Kalman filter may be achieved.

The prediction of the state variables should be applied to other applications of the power system like the contingency analysis or stability analysis. The inputs of the predictors can be enriched with some other relevant factors like the angular deviation of the rotating machines, the internal coherency of the generators etc. Instead of

using the PRBS signals, the randomness can be included in the system with real data from residential and industrial loads. Another practical way to investigate the randomness is to integrate the real data from solar or wind plants.

There are some systems where the basic WLS or the dishonest methods do not work due to the calculation of the gradients. These are known as the ill-conditioned systems. The heuristic methods such as the PSO or GA can be applied to these systems. The robustness of these heuristic methods against bad data also needs further investigation.

Appendix

Table 9.1: Values of the state variables, $|V|$, and θ for the failed case of 68-bus system as shown in Fig. 4.8

Bus	|V|	θ	Bus	|V|	θ
1	1.000000	0.000000	35	0.903346	0.294854
2	1.123044	-0.290585	36	0.660723	0.412601
3	1.077397	-0.297812	37	1.011343	0.550404
4	1.004883	-0.318400	38	1.042428	0.030962
5	0.958901	-0.312663	39	1.008511	0.554983
6	0.965078	-0.327882	40	1.225189	-0.058275
7	0.912044	-0.262654	41	1.240158	-0.393123
8	0.895098	-0.239621	42	1.339963	-0.143990
9	0.758673	0.152988	43	1.009167	0.549785
10	1.019755	-0.397631	44	1.009798	0.549518
11	0.998357	-0.374395	45	1.071776	0.347421
12	0.943231	-0.378499	46	1.123650	0.084432
13	1.017465	-0.386901	47	1.103357	0.023470
14	1.021613	-0.363019	48	1.161692	0.012454
15	1.052775	-0.386312	49	1.208029	0.091866
16	1.098565	-0.424251	50	1.363774	0.152734
17	1.097677	-0.367931	51	1.190591	0.304066
18	1.085835	-0.333407	52	1.504477	-0.049914
19	1.190758	-0.553913	53	1.125044	-0.289585
20	1.074964	-0.548420	54	0.967078	-0.326882
21	1.099046	-0.493611	55	1.021755	-0.396631
22	1.147669	-0.600822	56	1.192758	-0.552913
23	1.136976	-0.596297	57	1.076964	-0.547420
24	1.109760	-0.433404	58	1.149669	-0.599822
25	1.181595	-0.332462	59	1.138976	-0.595297
26	1.167035	-0.355902	60	1.183595	-0.331462
27	1.120066	-0.331762	61	1.204773	-0.488767
28	1.190985	-0.434027	62	1.043612	0.007477
29	1.202773	-0.489767	63	0.911234	-0.153819
30	0.916075	0.031617	64	0.662723	0.413601
31	1.041612	0.006477	65	1.013343	0.551404
32	0.909234	-0.154819	66	1.242158	-0.392123
33	0.863357	0.014014	67	1.341963	-0.142990
34	0.822119	0.271059	68	1.506477	-0.048914

Table 9.2: Values of the state variables $|V|$, and θ for the failed case of 118-bus system as shown in Fig. 4.9

Bus	|V|	θ	Bus	|V|	θ
1	1.000000	0.000000	60	1.308048	0.217817
2	1.254920	0.009599	61	1.337683	0.233351
3	1.164455	0.015533	62	1.269382	0.222704
4	1.001670	0.080460	63	1.370614	0.210836
5	1.179872	0.088314	64	1.233405	0.241728
6	1.295964	0.040666	65	1.320579	0.296357
7	1.415903	0.032987	66	1.356102	0.293390
8	1.227277	0.176278	67	1.313056	0.247313
9	1.459855	0.302815	68	1.410197	0.294612
10	1.494270	0.435285	69	1.468543	0.337372
11	1.164272	0.035779	70	1.094830	0.207869
12	1.118547	0.026704	71	0.980916	0.200364
13	0.958780	0.011868	72	1.329444	0.179943
14	1.352468	0.014486	73	1.279064	0.196699
15	1.122560	0.009774	74	0.989765	0.191463
16	1.309046	0.021642	75	1.153070	0.213628
17	1.218902	0.053582	76	1.357104	0.193732
18	1.156968	0.015010	77	1.022439	0.280125
19	1.041672	0.006632	78	1.280089	0.274889
20	0.970780	0.021991	79	1.053564	0.280125
21	0.943039	0.049742	80	1.302247	0.319221
22	1.339004	0.094422	81	1.254347	0.304211
23	1.115642	0.180293	82	1.156091	0.289201
24	1.275740	0.178373	83	1.260887	0.309796
25	1.051693	0.301244	84	1.149754	0.353953
26	1.256510	0.332311	85	1.357216	0.381180
27	0.971014	0.081681	86	1.287170	0.357269
28	1.032003	0.051487	87	1.177100	0.361807
29	1.115463	0.034208	88	1.243606	0.435809
30	1.271346	0.141721	89	1.272952	0.506495
31	1.347401	0.036303	90	1.375289	0.394793
32	1.317138	0.072082	91	1.422980	0.395143
33	1.034026	-0.000698	92	1.303967	0.403695
34	1.105628	0.010996	93	1.186419	0.351160
35	1.407531	0.003491	94	0.962161	0.313636
36	1.076834	0.003491	95	0.981334	0.296706
37	1.015842	0.019199	96	1.368939	0.293913
38	1.279548	0.108909	97	1.416611	0.300371
39	1.057154	-0.039444	98	1.308081	0.291994
40	1.080127	-0.057945	99	1.188759	0.285710
41	0.913231	-0.065450	100	1.445616	0.302989
42	1.213370	-0.037350	101	0.979749	0.330565
43	1.146151	0.010647	102	1.130653	0.377515
44	1.012095	0.054978	103	0.956386	0.240332
45	1.285239	0.087266	104	1.212708	0.192335
46	1.224372	0.136485	105	1.194589	0.172788
47	1.092231	0.175580	106	0.913465	0.168424
48	1.165665	0.161617	107	1.191761	0.119730
49	1.095608	0.179245	108	1.092674	0.152018
50	1.239512	0.143641	109	0.925283	0.144164
51	0.938937	0.097913	110	1.408138	0.129503
52	1.203241	0.081158	111	0.992275	0.158301
53	0.910561	0.064228	112	1.196231	0.075398
54	1.331205	0.080111	113	1.371510	0.053582
55	1.068414	0.075049	114	1.246506	0.066148
56	0.955070	0.078365	115	0.922720	0.066148
57	1.213467	0.099309	116	1.172455	0.287107
58	1.219227	0.084474	117	1.139809	0.000000
59	1.322104	0.151844	118	1.276773	0.196350

Bibliography

[1] A. Monticelli, *State estimation in electric power systems: A generalized approach.* Norwell, MA: Kluwer Academic Publishers, 1999.

[2] V. Kekatos and G. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. Power Systems*, vol. 28, no. 2, pp. 1617–1626, May 2013.

[3] L. Xie, D. H. Choi, S. Kar, and H. Poor, "Fully distributed state estimation for wide-area monitoring systems," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1154–1169, Sept 2012.

[4] A. G. Mutambara, *Decentralized estimation and control for multisensor systems.* CRC press, 1998.

[5] J. Stewart, T. Maufer, R. Smith, C. Anderson, and E. Ersonmez. (2016, January) Synchrophasor security practices. [Online]. Available: https://cdn.selinc.com/assets/Literature/Publications/Technical%20Papers/6449_SynchrophasorSecurity_EE_20100913_Web.pdf

[6] B. Luitel and G. K. Venayagamoorthy, "Cellular computational networks—A scalable architecture for learning the dynamics of large networked systems," *Neural Networks*, vol. 50, pp. 120–123, 2014.

[7] A. Semlyen and F. De León, "Quasi-Newton power flow using partial jacobian updates," *IEEE Trans. Power Systems*, vol. 16, no. 3, pp. 332–339, Aug. 2001.

[8] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, Mar. 2010.

[9] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008.

[10] M. A. Rahman and G. K. Venayagamoorthy, "Scalable cellular computational network based wls state estimator for power systems," in *Power Systems Conference (PSC), 2015 Clemson University.* IEEE, 2015, pp. 1–6.

[11] ——, "A hybrid state estimator for power systems using cellular computational network," *Engineering Application of Artificial Intelligence*, revised version under review.

[12] ——, "Dishonest gauss newton method based power system state estimation on a gpu," in *Power Systems Conference (PSC), 2016 Clemson University.* IEEE, 2016, p. to be published.

[13] ——, "Convergence of the fast state estimation for power systems," *SAIEE Research Journal*, accepted for publication.

[14] B. Otto, *Linear Algebra with Applications.* Upper Saddle River, NJ: Prentice Hall, 2005.

[15] F. C. Schweppe and J. Wildes, "Power system static-state estimation, Part I: Exact model," *IEEE Trans. Power Apparatus and Systems*, vol. PAS-89, no. 1, pp. 120–125, Jan. 1970.

[16] F. C. Schweppe and D. B. Rom, "Power system static-state estimation, Part II: Approximate model," *IEEE Trans. Power Apparatus and Systems*, vol. PAS-89, no. 1, pp. 125–130, Jan. 1970.

[17] F. C. Schweppe, "Power system static-state estimation, Part III: Implementation," *Power Apparatus and Systems, IEEE Transactions on*, vol. PAS-89, no. 1, pp. 130–135, Jan. 1970.

[18] A. S. Debs and R. E. Larson, "A dynamic estimator for tracking the state of a power system," *Power Apparatus and Systems, IEEE Transactions on*, no. 7, pp. 1670–1678, 1970.

[19] W. F. Tinney and C. E. Hart, "Power flow solution by newton's method," *Power Apparatus and Systems, IEEE Transactions on*, no. 11, pp. 1449–1460, 1967.

[20] W. L. Miller and J. B. Lewis, "Dynamic state estimation in power systems," *Automatic Control, IEEE Transactions on*, vol. 16, no. 6, pp. 841–846, 1971.

[21] M. F. Hassan, G. Salut, M. G. Singh, and A. Titli, "A decentralized computational algorithm for the global kalman filter," *Automatic Control, IEEE Transactions on*, vol. 23, no. 2, pp. 262–268, 1978.

[22] Y. Wallach, E. Handschin, and C. Bongers, "An efficient parallel processing method for power system state estimation," *Power Apparatus and Systems, IEEE Transactions on*, no. 11, pp. 4402–4406, 1981.

[23] D. M. Falcao, F. F. Wu, and L. Murphy, "Parallel and distributed state estimation," *Power Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 724–730, 1995.

[24] H. Karimipour and V. Dinavahi, "Extended kalman filter-based parallel dynamic state estimation," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1539–1549, May 2015.

[25] K. Maheshwari, M. Lim, L. Wang, K. Birman, and R. van Renesse, "Toward a reliable, secure and fault tolerant smart grid state estimation in the cloud," in *Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES*. IEEE, 2013, pp. 1–6.

[26] Y. Liu, W. Jiang, S. Jin, M. Rice, and Y. Chen, "Distributing power grid state estimation on hpc clusters-a system architecture prototype," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 2012, pp. 1467–1476.

[27] Y. Chen, Z. Huang, Y. Liu, M. J. Rice, and S. Jin, "Computational challenges for power system operation," in *System Science (HICSS), 2012 45th Hawaii International Conference on*. IEEE, 2012, pp. 2141–2150.

[28] R. Ebrahimian and R. Baldick, "State estimation distributed processing," *IEEE Trans. Power Syst*, vol. 15, no. 4, pp. 1240–1246, 2000.

[29] G. N. Korres, "A distributed multiarea state estimation," *IEEE Trans. Power Systems*, vol. 26, no. 1, pp. 73–84, Feb. 2011.

[30] M. M. Nordman and M. Lehtonen, "Distributed agent-based state estimation for electrical distribution networks," *Power Systems, IEEE Transactions on*, vol. 20, no. 2, pp. 652–658, 2005.

[31] W. Jiang, V. Vittal, and G. T. Heydt, "A distributed state estimator utilizing synchronized phasor measurements," *IEEE Trans. Power Systems*, vol. 22, no. 2, pp. 563–571, May 2007.

[32] A. K. Singh and B. C. Pal, "Decentralized dynamic state estimation in power systems using unscented transformation," *Power Systems, IEEE Transactions on*, vol. 29, no. 2, pp. 794–804, 2014.

[33] D. E. Marelli and M. Fu, "Distributed weighted least-squares estimation with fast convergence for large-scale systems," *Automatica*, vol. 51, pp. 27–39, 2015.

[34] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, no. 6, pp. 601–611, 1976.

[35] K.-I. Nishiya, H. Takagi, J. Hasegawa, and T. Koike, "Dynamic state estimation for electric power systemsâĂŤintroduction of a trend factor and detection of innovation processes," *Electrical Engineering in Japan*, vol. 96, no. 5, pp. 79–87, 1976.

[36] K. Nishiya, J. Hasegawa, and T. Koike, "Dynamic state estimation including anomaly detection and identification for power systems," in *Generation, Transmission and Distribution, IEE Proceedings C*, vol. 129, no. 5. IET, 1982, pp. 192–198.

[37] A. L. Da Silva, D. C. M B Filho, and J. Cantera, "An efficient dynamic state estimation algorithm including bad data processing," *Power Systems, IEEE Transactions on*, vol. 2, no. 4, pp. 1050–1058, 1987.

[38] F. L. Chernousko, *State estimation for dynamic systems*. CRC Press, 1993.

[39] R. Kandepu, B. Foss, and L. Imsland, "Applying the unscented kalman filter for nonlinear state estimation," *Journal of Process Control*, vol. 18, no. 7, pp. 753–768, 2008.

[40] Z. Huang, K. Schneider, and J. Nieplocha, "Feasibility studies of applying kalman filter techniques to power system dynamic state estimation," in *Power Engineering Conference, 2007. IPEC 2007. International*. IEEE, 2007, pp. 376–382.

[41] E. Ghahremani and I. Kamwa, "Dynamic state estimation in power system by applying the extended kalman filter with unknown inputs to phasor measurements," *Power Systems, IEEE Transactions on*, vol. 26, no. 4, pp. 2556–2566, 2011.

[42] G. Valverde and V. Terzija, "Unscented kalman filter for power system dynamic state estimation," *Generation, Transmission & Distribution, IET*, vol. 5, no. 1, pp. 29–37, 2011.

[43] J. L. Crassidis and J. L. Junkins, *Optimal estimation of dynamic systems*. CRC press, 2011.

[44] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*. International Society for Optics and Photonics, 1997, pp. 182–193.

[45] A. Abur and A. Exposito, *Power System State Estimation: Theory and Implementation*. New York, NY: Marcel Dekker Inc., 2004.

[46] A. E. Eiben, P. E. Raue, and Z. Ruttkay, "Genetic algorithms with multi-parent recombination," in *Parallel Problem Solving from Nature - PPSN III*. Springer, 1994, pp. 78–87.

[47] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel computing*, vol. 17, no. 6, pp. 619–632, 1991.

[48] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992.

[49] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1. New York, NY, 1995, pp. 39–43.

[50] Y. Del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: basic concepts, variants and applications in power systems," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 2, pp. 171–195, 2008.

[51] F. A. Zotes and M. S. Peñas, "Particle swarm optimisation of interplanetary trajectories from Earth to Jupiter and Saturn," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 1, pp. 189–199, 2012.

[52] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.

[53] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[54] Ş. Gülcü and H. Kodaz, "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 33–45, 2015.

[55] Z. Zhan, J. Zhang, Y. Li, and Y. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[56] R. Christie. (2016, March) Power system test case archive. [Online]. Available: https://www.ee.washington.edu/research/pstca/pf118/pg_tca118bus.htm

[57] I. P. Group. (2016, March) One-line diagram of ieee 118-bus test system. [Online]. Available: http://motor.ece.iit.edu/data/IEEE118bus_inf/IEEE118bus_figure.pdf

[58] B. Chaudhuri, R. Majumder, and B. C. Pal, "Wide-area measurement-based stabilizing control of power system considering signal transmission delay," *IEEE Trans. Power Systems*, vol. 19, no. 4, pp. 1971–1979, Nov 2004.

[59] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA tesla: A unified graphics and computing architecture," *IEEE Micro*, vol. 28, no. 2, pp. 39–55, Mar. 2008.

[60] M. Harris. (2016, Jan.) Optimizing parallel reduction in CUDA. [Online]. Available: http://vuduc.org/teaching/cse6230-hpcta-fa12/slides/cse6230-fa12--05b-reduction-notes.pdf

[61] D. Gervini and V. J. Yohai, "A class of robust and fully efficient regression estimators," *Annals of Statistics*, vol. 30, no. 2, pp. 583–616, Apr. 2002.

[62] L. Xie, D.-H. Choi, S. Kar, and H. V. Poor, "Fully distributed state estimation for wide-area monitoring systems," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1154–1169, Sep. 2012.

[63] H. Hassani and E. S. Silva, "A Kolmogorov-Smirnov based test for comparing the predictive accuracy of two sets of forecasts," *Econometrics*, vol. 3, no. 3, pp. 590–609, 2015.

[64] X. Li and A. Scaglione, "Robust decentralized state estimation and tracking for power systems via network gossiping," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 7, pp. 1184–1194, 2013.

[65] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[66] R. Holopainen, "Smoothness under parameter changes: derivatives and total variation," in *Proceedings of the Sound and Music Computing Conference*. KTH, 2013, pp. 646–653.

Biography

Md Ashfaqur Rahman received his Master of Science from the Dept. of Electrical Engineering at Texas Tech University in 2012. He completed his Bachelor of Science in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2009. He enrolled into the PhD program in Electrical Engineering at Clemson University in August 2014. His research area is power system state estimation with emphasis on fast and distributed state estimation using cellular computational network. He has been to date a research assistant with Real-Time Power and Intelligent Systems Laboratory and a teaching assistant in the dept. of Electrical and Computer Engineering at Clemson University.