DETECTION OF STEALTHY FALSE DATA INJECTION ATTACKS IN TRANSMISSION SYSTEMS USING KALMAN FILTERS

A Thesis in
Electrical Engineering
by
Alberto Miguez Dominguez

© 2021 Alberto Miguez Dominguez

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2021
The thesis of Alberto Miguez Dominguez was reviewed and approved by the following:

Peter B. Idowu
Assistant Dean for Graduate Studies, Graduate Studies
Professor of Electrical Engineering, School of Science, Engineering and Technology
Thesis Advisor

Aldo W. Morales
Professor of Electrical Engineering, School of Science, Engineering, and Technology
Co-Director, Center for Signal Integrity

Seth Wolpert
Associate Professor of Electrical Engineering, School of Science, Engineering, and Technology

Rafic A. Bachnak
Professor of Electrical Engineering, School of Science, Engineering, and Technology
Professor-in-Charge, Master in Engineering Management
Professor-in-Charge, Master of Science in Electrical Engineering
Abstract

A smart grid is an electricity grid that allows two-way flow of electricity in its network, enabling consumers to have better control over their electricity usage while reducing the operations and management costs for utilities. The communication devices in smart grids have increased the integration of renewable energy systems, such as wind and solar, and have proven to be very effective at helping restore power faster when a power disturbance occurs. In recent years, the integration of more communication devices in the power grid has opened the opportunity for more Data Integrity cyber-physical attacks. Smart grids can be a prime target for these types of attacks which can lead to cascading failures in a transmission system. False Data Injection attacks, a type of Data Integrity cyber-physical attack, can manipulate the system’s measurements, and therefore, the power dispatch, in a way that can make the lines in the system overflow. This type of attack could theoretically be performed without the operator ever knowing that there was an attack, and it can cause power outages and even system blackouts.

The purpose of this thesis is to implement a False Data Injection attack strategy on targeted buses that bypass DC state estimation and develop a new algorithm that can detect them using AC state estimation with Kalman Filters. Possible attacks on the system will be considered and Kalman Filters will be used to aid in the detection of bad data injections in the system that would allow the operator to know if there is an attack currently happening. The proposed novel algorithm was developed in MATLAB and tested using a modified IEEE 14 bus-system with a fixed power flow between lines of 25 MW.

Index Terms – Cyber-Physical Attack, Attack Strategy, False Data Injection, Kalman Filters, IEEE 14 bus-system
# Table of Contents

List of Figures vi  
List of Tables vii  
List of Symbols ix  
Acknowledgments xi  

Chapter 1  
Introduction 1  
1.1 Definition and Overview ........................................... 1  
1.2 Literature Review .................................................. 4  
1.2.1 Contributions ................................................... 5  
1.2.2 Thesis Objectives and Organization .......................... 6  

Chapter 2  
Power Flow 8  
2.1 Introduction ......................................................... 8  
2.2 Newton-Raphson Power Flow Method ........................... 8  
2.3 Gauss-Seidel Method ............................................... 11  

Chapter 3  
Attack model and results 14  
3.1 FDI attacks on DC state estimation .............................. 14  
3.2 Case Studies .......................................................... 16  
3.2.1 Available lines to attack in DC state estimation ......... 16  
3.2.2 Overflowing a single line with multiple targeted buses .... 17  
3.2.3 Overflowing Multiple Lines .................................... 18  
3.3 Conclusion ............................................................. 19  

Chapter 4  
State estimation 20  
4.1 Weighted Least Squares (WLS) ................................. 20  
4.1.1 Weighted Least Squares Estimation Results ............... 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Discrete Kalman Filter (DKF)</td>
<td>23</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Introduction</td>
<td>23</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Implementing the algorithm</td>
<td>24</td>
</tr>
<tr>
<td>4.2.3</td>
<td>DKF Results</td>
<td>28</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>30</td>
</tr>
<tr>
<td>5.2</td>
<td>Reconstructed Good Data</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>Reconstructed Bad Data</td>
<td>34</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Reconstructed Bad Data: Case 1</td>
<td>35</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Reconstructed Bad Data: Case 2</td>
<td>36</td>
</tr>
<tr>
<td>5.4</td>
<td>Conclusion</td>
<td>38</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Matlab Code</td>
<td>40</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Power Flow Algorithms</td>
<td>72</td>
</tr>
<tr>
<td>Appendix C</td>
<td>State estimation algorithms</td>
<td>74</td>
</tr>
<tr>
<td>Appendix D</td>
<td>Reconstructed measurements raw data</td>
<td>76</td>
</tr>
<tr>
<td>Bibliography</td>
<td>81</td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

1.1 Smart grid [1] ................................................................. 2
1.2 Types of Data Integrity Attacks [2] ................................. 3
1.3 Chapter organization ...................................................... 7
2.1 Gauss-Seidel Algorithm for power flow ............................... 13
3.1 Flowchart of model proposed in [3] ................................. 16
5.1 Novel algorithm for detecting FDI attacks ............................ 31
5.2 Original (blue) vs Reconstructed Good Data measurements (orange) . . 34
5.3 Original (blue) vs Reconstructed Bad Data 1 measurements (orange) . . 36
5.4 Original (blue) vs Reconstructed Bad Data 2 measurements (orange) . . 37
List of Tables

2.1 Gauss-Seidel Algorithm for power flow ........................................... 12

3.1 False Data Injections to overflow only Line 7 ........................................ 17
3.2 False Data Injections to overflow only Line 15 ......................................... 18
3.3 False Data Injections to overflow Lines 7 and 15 ....................................... 19

4.1 WLS Estimation Results ................................................................. 23
4.2 Discrete Kalman Filter Estimation Results ............................................. 29

5.1 Line Flow Measurements ................................................................. 32
5.2 Bus Injection Measurements ............................................................. 32
5.3 Voltage Measurements ................................................................. 32
5.4 False Data Injections to overflow only Line 7 ........................................ 35

D.1 Reconstructed Line Flow Measurements ............................................. 76
D.2 Reconstructed Bus Injection measurements ............................................ 76
D.3 Reconstructed Voltages ................................................................. 77
D.4 Results ................................................................. 77
D.5 Reconstructed Line Flow Measurements ............................................. 78
D.6 Reconstructed Bus Injection measurements ............... 78
D.7 Reconstructed Voltages .................................................. 78
D.8 Results ........................................................................... 79
D.9 Reconstructed Line Flow Measurements ....................... 79
D.10 Reconstructed Bus Injection measurements ..................... 80
D.11 Reconstructed Voltages .................................................. 80
List of Symbols

- $V$: Voltage magnitude
- $\theta$: Voltage angle
- $P$: Active Power
- $Q$: Reactive power
- $J$: Jacobian matrix
- $\tau$: Percentage of nominal load power
- $\Delta D$: False Data Injection matrix
- $\alpha_l$: Weighting factor
- $G$: Gain matrix
- $R$: Error covariance matrix
- $H$: Measurement matrix
- $z$: Measurement vector
- $x$: State vector
- $G$: Real part of bus admittance matrix
- $B$: Imaginary part of bus admittance matrix
- $Y_{bus}$: Admittance matrix
- $g$: Real part of line impedance matrix
- $b$: Imaginary part of line impedance matrix
- $P_G$: Generated active power
- $Q_G$: Generated reactive power
\( P_L \) Load active power

\( Q_L \) Load reactive power

\( \epsilon \) Small threshold number
Acknowledgments

I would like to first thank my thesis advisor Dr. Peter Idowu for his consistent support throughout the last stages of this thesis, as well as for helping me improve its quality. Your constant help has helped me finish this thesis. Your efforts don’t go unnoticed.

I also would like to thank Dr. Javad Khazaei at The Pennsylvania State University. He has motivated me to always go the extra mile in every homework and project, and has steered me in the right the direction whenever he thought I needed it. He is an excellent professor and the best mentor I have ever had.

I would like to express my most profound gratitude to my parents and wife for supporting me throughout my studies, and particularly during this past year. You have made this possible for me.

Thank you
Chapter 1  
Introduction

1.1 Definition and Overview

A smart grid is an electricity grid that allows two-way flow of electricity in its network, enabling consumers to have better control over their electricity usage while reducing the operations and management costs for utilities, ultimately lowering power costs for consumers. The communication devices in smart grids have increased the integration of renewable energy systems, such as wind and solar, and have proven to be very effective at helping restore power faster when a power disturbance occurs. They also allow consumers to sell their own power generated by renewable energies back to the grid so it can be used by other consumers [4].

The electric infrastructure in the US is rapidly aging and it is being pushed to do more than it was originally designed to do. The average total annual power disruption duration in the US has increased by nearly 71% from 2013 to 2018, from 3.5 hours to nearly 6 hours, respectively. Major events, mainly climate-related events like flooding or extreme temperatures, are the lead problem for the increase in power disruptions from 2013 to 2018 (U.S. Energy Information Administration). Updating the grid to make it more resilient by incorporating more communication devices and improved equipment can reduce the frequency and duration of power outages, reduce the impact of flooding or storm, and restore service faster when outages occur. This transformation of the nation’s electric grid creates both challenges and opportunities to advance its capabilities. A simple smart grid diagram is shown in Figure 1.1 representing the different power providers and consumers in a modern smart grid, as well as the use of renewable energies that provide more flexibility to the power grid.
Despite the several advantages of having more control and data over the power network, smart grids are vulnerable to cyber-attacks due to the increased number of Advanced Metering Infrastructure (AMI) devices used in a system to determine the optimal power generation [5]. There are several types of data integrity cyber-attacks that a power network can be vulnerable to. First, in Replay attacks the attacker records sensor measurements and then uses those recorded measurements to substitute the actual ones at a different point in time. Second, in False Data Injection (FDI) attacks the attacker targets the measurement and control channels of the system and typically corrupts the data received by the measurement devices to trick the system to make the network fail. And lastly, in Covert attacks, the attacker calculates the output response of a system and subtracts it from the measurement readings to be undetected [2]. A simple table with the types of Data Integrity Attacks in power systems is shown in Figure 1.2.

More than ever, power suppliers are opening to renewable energies and adding smart meters to measure the added power supplied to the grid. Over the past decades, power plants and substations have been connected to public and also privately-owned networks to access information remotely which leaves them exposed to attacks. Producers and distributors have been reluctant to spend on protecting themselves against low-probability attacks. But in the past 6 years, the world has seen many cyber-attacks on different power grids across the globe. In December 2015, hackers got into the system of a western Ukrainian power company, cutting power to 225,000 households. In December of 2016, Ukraine experienced another cyber-attack on their power grid, where hackers left customers in parts of Kyiv without electricity for an hour, after disabling an electricity
substation. In August 2017, Saudi Aramco became the target of cyber-attacks when hackers targeted the safety system in one of the company’s petrochemical plants, forcing the plant to shut down [6].

More research in power systems and cyber-security is needed to ensure a more resilient, cleaner, and affordable power grid for the future.

In this thesis, we will focus on False Data Injection attacks. To understand the effects of FDI attacks on corrupting the data gathered by the AMI devices, it is imperative to understand how the power generation is calculated by the network using state estimation [7].

![Figure 1.2. Types of Data Integrity Attacks](image)

Smart grids are composed of many interconnected power lines and may also include several components such as loads and generators in a power system. A bus is a node where a power line or several power lines are connected. In today’s world, it is impossible to gather data from all buses at the same time and calculate the exact power that needs to be generated for the power network at all times. Therefore, state estimation is mainly used to approximate the voltage magnitude and angle in parts of the power system which are not directly metered, based on other measurements in the same network. If some measurements in the system are corrupted by an FDI attack, the power requirements
predicted by the state estimation algorithm can cause the system to send too much power through some power lines and make them overflow and fail [8,9]. In this thesis, we will explore the contributions made to FDI attack models by different authors and my contribution towards the detection of such attacks in a power grid by testing my methods on an IEEE 14-bus system.

1.2 Literature Review

There has been extensive work on modeling FDI attacks in recent years. In [10] a novel algorithm is developed for optimal sensor placement to improve the security of the distribution network to ensure it is observable. Moreover, in [11] another novel mathematical model that can consider weather events while calculating the reliability of the energy grid. False data injections attacks have been studied in this paper from the state estimation perspective in [12]- [13] and introduced the detection in DC state estimation, but not in AC state estimation. Liu et al. [12] developed a model for false data injection. Kosut et al proposed a heuristic method that can detect the worst-case attack scenario. In [14], a detection method to identify false data injection is proposed. In previous studies the authors in [14] addressed false data injection attacks, however, they have not presented the attacker’s strategy and scenario for such a grid network. In [13], the attack model to overflow a transmission line is introduced. However, the author of this study considered only one single transmission line to be attacked. Also, the congestion problem of the multiple transmission lines is considered, too. In [15], the attack is classified as a Data attack where the attacker illegally inserts, alters, or deletes data in the communication network traffic to confuse the smart grid to trigger mistakes. A recent study [16] it was mentioned that the power industry is targeted the most, up to 54%, in cyber-attacks based on attacks recorded in the US. Thus security is a major challenge in the deployment and running of IoT-based smart grid networks. In [17], the risk assessment methodology is described for any cyber-physical system, where smart metering is considered to be likely hacked compared to other cyber components like ECC, which is the main component that controls the power generation, that is considered rarely hacked.

In [18], a novel algorithm is developed to model an efficient attack model to maximize the total failure impacts on users in smart grids, although this paper considers a limited budget which may reduce the efficacy of the algorithm since it limits the power distribution options in the system. The need for false data detection to measure potential and physical
damaging effects on power systems is highlighted in [19]. In [20], the authors demonstrate the effect of GPS spoofing on meters in a system, and they develop an attack strategy to test these kinds of attack on a real system. In [21], a summary and comparison of different false data injection attack detection algorithms is found. Each attack strategy is rated depending on what percentage of FDI attacks is able to detect. The paper in [22] considers the most important cyber-physical security algorithms applied for power systems. Some defense mechanisms are shown in order to prevent potential cyber-attackers from obtaining important system data and prevent harm in the system. Moreover, in [23], a simplified version of an attack model is considered. The model selects the easiest target in the system with the least cost but fails to consider the selection of a specific line in the system that would make the system go into a potential cascading failure.

An attack model by [3] was developed to find the minimum power injections into transmission lines that would cause the grid to overflow one or multiple lines. The attack model was formulated using a bi-level mixed-integer linear programming (BMILP) problem with an upper-level and lower-level problem. The upper-level problem models the attack problem and provides the false data injections to be used to overflow transmission lines. The lower-level problem solves the economic dispatch problem using DC power flow with injected data.

1.2.1 Contributions

Based on the fore-mentioned contributions from different authors, FDI attacks can still bypass the current detection algorithms based on DC state estimation [3]. In this thesis, a novel algorithm will be developed to establish a threshold between normal operation in the IEEE 14-bus system and during an FDI attack. This algorithm will allow the power network to detect FDI attacks, which would open the opportunity to build defenses to prevent harm in the system from such attacks. The damage caused by an FDI attack, if left unchecked, could vary from the failure of one power line to a cascading effect that could make the entire grid fail.

The novel algorithm will receive the state estimation of the system by a Discrete Kalman Filter which will be key in approximating the voltage magnitude and angles in the system, and will then reconstruct the original measurements based on those estimated values. Based on the reconstructed measurements, disparities between the original measurements and the reconstructed ones will arise, where a similarity threshold will determine if the system is suffering an FDI attack or not. This model will incorporate
for the first time AC state estimation in the complex attack model by [3] which would allow the system to react to an FDI attack to prevent potential failures in the system.

In this thesis, I will also test unique FDI attacks on the IEEE 14-bus system and prove that the novel algorithm is able to detect such attacks. This contribution will open the door to developing methods to protect a system once an FDI attack has been detected.

1.2.2 Thesis Objectives and Organization

Based on the previous work presented in the area of FDI attacks and in combination with the use of Kalman Filters, the objectives for this thesis are summarized as follows:

1. Determine if an algorithm can be developed to detect FDI attacks that bypass DC state estimation, using AC state estimation.
2. Develop a state estimation algorithm using Discrete Kalman Filters (DKF) for power systems, and find the estimated values for the magnitude and angle of buses that are not directly metered in the system.
3. Test the developed state estimation algorithm using simulation software and record the margin by which the system identifies an FDI attack.

Regarding the thesis structure, it will feature 4 chapters, in addition to the introductory chapter, where several key components will be presented and discussed. Figure 1.3 shows a diagram of how the thesis is organized.

In chapter 2, two methods to calculate the power flow in a power network are presented. Both methods will be used when calculating the power flow in the rest of the chapters. These methods for calculating power flow provide an essential theoretical framework to find voltage magnitudes and angles in different buses, as well as active and reactive power at each bus. These power flow methods are described from a high-level perspective, and a more comprehensive description can be found in Appendix B.

Subsequently, Chapter 3 will present the attack model by [3] used to determine the weaknesses and the amount of power that needs to be injected in the IEEE 14-bus system to perpetrate an FDI attack. A total of 3 different attacks are considered, in which different sets of buses are manipulated to trick the power network from detecting an overflowing power line or power lines in the system. The first attack presented in this chapter is further investigated in Chapter 5 to demonstrate how the detection algorithm I developed would detect an FDI attack on this system.

Chapter 4 focuses on two methods for state estimation. First, the Weighted Least
Squares (WLS) algorithm is used to establish a baseline for the voltage magnitude and angles in the system. Second, the Discrete Kalman Filter (DKF) algorithm will be introduced and the result estimates for the IEEE 14-bus system under normal operation will be presented. Because it is challenging to directly use Discrete Kalman Filter (DKF) due to its need for precision in the range of error considered, the WLS also serves as a baseline to tune the DKF.

Lastly, all the information from the previous chapters will be combined in Chapter 5 to develop a detection algorithm for FDI attacks. The equations from Chapter 2 will be used to obtain the results for the IEEE 14-bus system under normal operations and during an FDI attack. The attack results from Chapter 3 will be used to model the FDI attack in the IEEE 14-bus system. The Discrete Kalman Filter algorithm from Chapter 3 will be used to determine the estimated voltage magnitudes and angles in the system during normal operation, which will be compared to the results seen during an FDI attack. Finally, the estimated and actual measurements will then be compared and a detection algorithm based on the Euclidean length between the two will determine if there is an FDI attack in progress or not.

Figure 1.3. Chapter organization
Chapter 2  
Power Flow

2.1 Introduction

Power flow is widely used in power systems to analyze the flow of power between different transmission lines. The power flow model of a power system can be constructed using key components such as the load and generation data for that network. Once the power flow model is developed, we may obtain the voltage magnitude and angles at each bus, line flows for each transmission line, and the power losses in the system. These results are obtained by setting up equations at each bus and solving the power balance equations. Due to the nonlinearity of these nodal equations, iterative methods such as the Newton-Raphson or the Gauss-Seidel methods are typically used to solve these equations.

The objective of a power flow study is to calculate the voltage magnitude and angle for a specific power generation, load requirements, and a given power network. Losses in the system, as well as line flows, can be calculated once all the voltages are known. To begin solving these power flow equations we need to determine the known and unknown variables in the system [24].

2.2 Newton-Raphson Power Flow Method

The Newton-Raphson technique is based on Taylor’s expansion approximation. By initiating the estimate with \( x[0] \), the deviation from the accurate solution is \( \Delta x[0] \). This algorithm offers fast convergence as long as the initial guess is not too different from the actual solution. Some disadvantages are that each iteration takes much longer than a Gauss-Seidel iteration and it is also more complicated to code. The Newton-Raphson
algorithm is very commonly used in power flow analysis [25].

When using the Newton-Raphson method, we have to establish the state variables in the system. In a power system, those state variables typically are the voltage magnitudes and angles for each of the buses in the system. Some angles and voltages can be assumed as given when assuming a specific bus to be the slack bus (typically the first bus).

The Newton-Raphson algorithm is summarized below in 5 steps. This iterative method approximates the state variables and reaches stable results if it can find a solution for the power flow problem.

**Newton-Raphson Algorithm for Power Flow**

1. Establish state variables in \( x(i) \) with voltage magnitude and angles, \( x(i) = \begin{bmatrix} \theta(i) \\ V(i) \end{bmatrix} \)

2. Compute the difference in active and reactive powers, \( \begin{bmatrix} \Delta P(i) \\ \Delta Q(i) \end{bmatrix} = \begin{bmatrix} P - P[x(i)] \\ Q - Q[x(i)] \end{bmatrix} \)

3. Calculate the Jacobian matrix elements as shown in Equation 2.4, \( J = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \)

4. Solve for \( \Delta x \) using, \( \begin{bmatrix} \Delta \theta(i) \\ \Delta V(i) \end{bmatrix} = \begin{bmatrix} \Delta P(i) \\ \Delta Q(i) \end{bmatrix} = \begin{bmatrix} J \end{bmatrix}^{-1} \begin{bmatrix} P - P[x(i)] \\ Q - Q[x(i)] \end{bmatrix} \)

5. Update \( x \) for the next time step \( (i+1) \), \( x(i+1) = \begin{bmatrix} \theta(i+1) \\ V(i+1) \end{bmatrix} = \begin{bmatrix} \theta(i) \\ V(i) \end{bmatrix} + \begin{bmatrix} \Delta \theta(i) \\ \Delta V(i) \end{bmatrix} \)

A mathematical framework will be provided below to do the calculations from the algorithm. Note that \( V_1 \) is not considered since it is the 'Slack bus' and is always assumed to be 1.
\[
x = \begin{bmatrix} \theta \\ V \end{bmatrix} = \begin{bmatrix} \theta_2 \\ \vdots \\ \theta_N \\ V_2 \\ \vdots \\ V_N \end{bmatrix} \tag{2.1}
\]

The equations for determining the active and reactive power in each iteration are given by Equations 2.2 and 2.3.

\[
P_k(x) = |V_k| \sum_{n=1}^{N} |Y_{kn}| |V_n| \cos(\theta_k - \theta_n - \theta_{kn}) \tag{2.2}
\]

\[
Q_k(x) = |V_k| \sum_{n=1}^{N} |Y_{kn}| |V_n| \sin(\theta_k - \theta_n - \theta_{kn}) \tag{2.3}
\]

Using the equations above, we can solve for the Jacobian matrix, J, by taking the partial derivatives of each active and reactive power measurement with respect to each state variable as shown in Equation 2.4.

\[
J = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} = \begin{bmatrix}
\frac{\partial P_2}{\partial \theta_2} & \cdots & \frac{\partial P_2}{\partial \theta_N} & \frac{\partial P_2}{\partial V_2} & \cdots & \frac{\partial P_2}{\partial V_N} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial P_N}{\partial \theta_2} & \cdots & \frac{\partial P_N}{\partial \theta_N} & \frac{\partial P_N}{\partial V_2} & \cdots & \frac{\partial P_N}{\partial V_N} \\
\frac{\partial Q_2}{\partial \theta_2} & \cdots & \frac{\partial Q_2}{\partial \theta_N} & \frac{\partial Q_2}{\partial V_2} & \cdots & \frac{\partial Q_2}{\partial V_N} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial Q_N}{\partial \theta_2} & \cdots & \frac{\partial Q_N}{\partial \theta_N} & \frac{\partial Q_N}{\partial V_2} & \cdots & \frac{\partial Q_N}{\partial V_N} \\
\frac{\partial \theta_2}{\partial \theta_2} & \cdots & \frac{\partial \theta_2}{\partial \theta_N} & \frac{\partial \theta_2}{\partial V_2} & \cdots & \frac{\partial \theta_2}{\partial V_N} \\
\frac{\partial \theta_N}{\partial \theta_2} & \cdots & \frac{\partial \theta_N}{\partial \theta_N} & \frac{\partial \theta_N}{\partial V_2} & \cdots & \frac{\partial \theta_N}{\partial V_N} \\
\end{bmatrix} \tag{2.4}
\]

Each term in the Jacobian matrix is calculated differently, depending if the partial derivative of the active or reactive power at that bus is with respect to the state variable at that same bus or not. If the bus number of the partial derivative of the active or reactive power with respect to the state variable is not equal, or \( n \neq k \), that element of the Jacobian matrix will be calculated following Equations 2.5-2.8. If the bus number is the same, or \( n = k \), that element of the Jacobian matrix will be calculated using Equations 2.9-2.12.
When $n \neq k$

\[
J_{1}^{kn} = \frac{P_{k}}{\partial \theta_{n}} = V_{k}Y_{kn}V_{n} \sin(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.5}
\]

\[
J_{2}^{kn} = \frac{P_{k}}{\partial V_{n}} = V_{k}Y_{kn} \cos(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.6}
\]

\[
J_{3}^{kn} = \frac{Q_{k}}{\partial \theta_{n}} = -V_{k}Y_{kn}V_{n} \cos(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.7}
\]

\[
J_{4}^{kn} = \frac{Q_{k}}{\partial V_{n}} = V_{k}Y_{kn} \sin(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.8}
\]

When $n = k$

\[
J_{1}^{kk} = \frac{P_{k}}{\partial \theta_{k}} = -V_{k} \sum_{n=1, n \neq k}^{N} Y_{kn}V_{n} \sin(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.9}
\]

\[
J_{2}^{kk} = \frac{P_{k}}{\partial V_{k}} = V_{k}Y_{kk} \cos(\theta_{kk}) + \sum_{n=1}^{N} V_{n}Y_{kn} \cos(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.10}
\]

\[
J_{3}^{kk} = \frac{Q_{k}}{\partial \theta_{k}} = -V_{k} \sum_{n=1, n \neq k}^{N} Y_{kn}V_{n} \cos(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.11}
\]

\[
J_{4}^{kk} = \frac{Q_{k}}{\partial V_{k}} = -V_{k}Y_{kk} \sin(\theta_{kk}) + \sum_{n=1}^{N} V_{n}Y_{kn} \sin(\theta_{k} - \theta_{n} - \theta_{kn}) \tag{2.12}
\]

### 2.3 Gauss-Seidel Method

The Gauss-Seidel method is an iterative algorithm for solving a system of non-linear equations. To begin this iterative method, a vector containing estimated results based on previous experience is assumed. An update to the estimated results is calculated and is substituted on the solution vector that we introduced in the previous step for this variable. We then repeat this iterative process for all the variables, completing one iteration step. The algorithm is then required to go back to the first step and run several iterations until the solution vector converges to a specific value. Depending on the set of initial parameters in the solution vector that we assumed in the first step, the speed at which the algorithm will converge changes.

In a power system, each bus is labeled as a Slack (or Swing), PV, or PQ bus, depending on the parameters known for each bus. The slack bus is typically considered as the reference bus because both voltage magnitude and angles are known. For PV buses, or generator buses, the net real power as well as the voltage known for that bus. The
majority of buses in a power system are PQ buses, or load buses, where both the active and reactive powers are known.

The objective of calculating the power flow of a power system is to find the voltage magnitude and angles at each bus. For the slack bus, the voltage magnitude and angle are already specified, so there are no variables to solve for. For PV buses, the voltage magnitude is known while the voltage angle is unknown. And lastly, for PQ buses both the voltage magnitude and angles are unknown. In a system with $n$ buses and $g$ generators, there are $2(n-1)-(g-1)$ unknowns. A summary of the different bus categories can be found below in Table 2.1 based on the known variables in that specific bus.

<table>
<thead>
<tr>
<th>Known</th>
<th>Slack bus</th>
<th>PQ buses</th>
<th>PV buses</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_k, \theta_k$</td>
<td></td>
<td>$P_k, Q_k$</td>
<td>$P_k, V_k$</td>
</tr>
<tr>
<td>Unknown</td>
<td>$P_k, Q_k$</td>
<td>$V_k, \theta_k$</td>
<td>$Q_k, \theta_k$</td>
</tr>
</tbody>
</table>

Table 2.1. Gauss-Seidel Algorithm for power flow

The algorithm to solve for the Gauss-Seidel method can be found in Figure 2.1 below. A step-by-step explanation for this algorithm can be found in Appendix B, under the Gauss-Seidel Method in-depth section.

In Chapter 3, both methods will be applied in realistic attack conditions from [3] to determine the minimum number of false data injections to overload a power line in the system. These methods from Chapter 2 will also be used to determine the power flow at each bus during an FDI attack and the effect of targeting multiple power lines in the rest of the system. Multiple case studies, based on targeting different power lines, will be presented.
Figure 2.1. Gauss-Seidel Algorithm for power flow
Chapter 3  
Attack model and results

3.1 FDI attacks on DC state estimation

The main purpose of this chapter is to implement the authors’ model in [3] for the IEEE 14-bus benchmark to develop a detection algorithm for FDI attacks. The authors of the original paper in [3], modeled an attack based on false data injections in multiple transmission lines simultaneously. Their model proposes the following:

- A list of transmission lines can be focused to be overflowed bypassing the bad data detection in the DC state estimation algorithm.
- The attacker may have limited access to only certain buses in a system to inject false data.
- Multiple lines can be attacked simultaneously by injecting minimum false data in selected buses to overflow the targeted lines.
- The transmission line overflow is limited to appear realistic from the operator’s point of view.

The attacker can access smart meters and inject false data that can lead to lines overflowing and potential cascading failures. They would need information about the system in order to perform a successful attack, including the reactance of the transmission lines. They must also ensure that the target lines to be overflowed are not detected by the bad data detection in the DC state estimation algorithm [12]- [13] [26] [3].

In the paper, they considered the following physical limitations that an attacker would encounter when injecting false data [27]:
1. The attacker would only have access to specific buses and would not have access to
generators due to their higher security measures
2. Load injections in the system should not surpass a percentage of the nominal load
3. Power balance must be achieved by ensuring that the sum of all false data injections is zero.
4. Load and generation should always be balanced.

Their main objective is to create a mathematical model that mimics false data injection attacks in a real system on a set of targeted buses to overflow one or more transmission lines without triggering any bad data detection algorithms in DC state estimation.

Their model was formulated using a bi-level mixed-integer linear programming (BMILP) problem with two layers. The first layer of the problem (or upper-level problem) oversees finding the false data injections, targets the line that needs to be overflowed, as well as the buses that the attacker has access to. The second layer (or lower-level problem) oversees the economic dispatch with the false data injections from the upper-level and ensures that it bypasses the DC state estimation algorithm [3]. A more in-depth explanation of the model can be found in Appendix E.

The proposed model in [3] can be summarized as a flowchart illustrated in Figure 3.1. Prior to the attack, the attacker obtains information about the system regarding the power flow between lines and the system characteristics (such as loads for each bus and reactance of the transmission lines). The attacker then runs the upper-level problem for the buses that have accessible and selects a transmission line that wants to attack. The attacker then runs the lower-level problem to ensure that the DC state estimation algorithms get bypassed and do not detect the attacker. The operator then will make the decision based on the economic dispatch solution.

Regarding the profile of the attacker, they would need to be well versed in the operation of power flow in a system. Potentially with a background in Electrical and Computer Engineering, they would be able to steal information regarding the system’s lines impedance and run a model that would yield a successful attack vector to overflow one or more lines. This attacker would require to be an expert in economic dispatch from the operator point-of-view, as well as at injecting power at different buses locations. The attacker would potentially need several accomplices to coordinate the attack.
In the case studies below, we explored different scenarios that an attacker could consider when attacking this system. Single-line attacks will be considered first since they are easier to approach from an attacker’s point of view, and a multi-line attack will also be considered. The results of the case studies will allow studying a detection algorithm to detect these attacks.

### 3.2 Case Studies

#### 3.2.1 Available lines to attack in DC state estimation

Based on this model, not every line can be targeted due to cyber-physical limitations. The first step of the attack model will consider all transmission lines depending on the parameters of the system. The parameters considered are $\tau_1$ as 1.2, $\tau_2$ as 0.2 (or 20%), and $\alpha_L$ as 1.05. A line that is already transmitting close to the maximum power flow allowed for that line is typically a very good candidate to be overflowed since minimal injections in specific buses should overflow the line slightly.
Our model differs from the one considered in [3]. For our model, we considered that all lines have a maximum flow of +/- 25 MW, so the results shown in the case studies below will not match the original paper as it considered different maximum power flows for each line.

In every case study considered below, the targeted buses and false data injections ($\theta_D$), total cost before and after the attack, and line power flow after the attack will be displayed in the corresponding Tables.

### 3.2.2 Overflowing a single line with multiple targeted buses

In this case study, the objective is to overflow one transmission line by injecting false data on certain buses.

In the first scenario, the attacker targets line 7 which connects bus 4 to 5 as seen in Figure 3.1. For this specific line, two different sets of buses can be chosen to overflow line 7. In case 1, buses 6, 9, and 13 can be injected with false data to overflow this line (the injections are equal in value but opposite so load and generation are equal). And buses 3, 5, and 6 could also be targeted in case 2. Results are shown in Table 3.1. It is observed that line 7 in both cases will be -26.25 MW, exceeding by 5% the maximum power flow allowed of 25 MW. Therefore, this line will be overflowed and will be considered a successful attack.

<table>
<thead>
<tr>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_2 = 0.2, \alpha_L = 1.05$</td>
<td>$\tau_2 = 0.2, \alpha_L = 1.05$</td>
</tr>
<tr>
<td>Targeted Buses</td>
<td>$\Delta D$</td>
</tr>
<tr>
<td>6</td>
<td>-2.24 MW</td>
</tr>
<tr>
<td>9</td>
<td>4.31 MW</td>
</tr>
<tr>
<td>13</td>
<td>-2.07 MW</td>
</tr>
<tr>
<td>P7</td>
<td>-26.25 MW</td>
</tr>
<tr>
<td>Cost (no attack)</td>
<td>$15,041.43$</td>
</tr>
<tr>
<td>Cost (with attack)</td>
<td>$15,330.97$</td>
</tr>
</tbody>
</table>

In the second scenario, the attacker could also choose to target line 15 which connects buses 7 and 9. The attacker could accomplish this in two different ways, as shown in
Case 1 and 2 in Table 3.2. The attacker could overflow line 15 by targeting buses 3, and 9, or by targeting buses 4, 6, and 10. Line 15 in both cases would have a power flow of -26.25 MW, which is 5% higher than the allowable 25 MW power flow (since \( \tau_L \) is 1.05). Hence, Line 15 would be overflowed, and the attack would be successful.

**Table 3.2.** False Data Injections to overflow only Line 15

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targeted Buses</td>
<td>( \tau_2 = 0.2, \alpha_L = 1.05 )</td>
<td>( \tau_2 = 0.2, \alpha_L = 1.05 )</td>
</tr>
<tr>
<td>Targeted Buses</td>
<td>( \Delta D )</td>
<td>( \Delta D )</td>
</tr>
<tr>
<td>3</td>
<td>-2.70 MW</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>2.70 MW</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>P15</td>
<td>26.25 MW</td>
<td>P15</td>
</tr>
<tr>
<td>Cost (no attack)</td>
<td>$15,041.43</td>
<td>Cost (no attack)</td>
</tr>
<tr>
<td>Cost (with attack)</td>
<td>$15,169.20</td>
<td>Cost (with attack)</td>
</tr>
</tbody>
</table>

It is also noted that the cost of the system after the attack increases by a maximum of 1.92% in scenario 1 for the first case, compared to the cost before the attack. In the second scenario in Table 3.2, the second case has a cheaper cost with the attack. This case is particularly dangerous since this scenario will be more likely to be chosen by the operator during the economic dispatch.

### 3.2.3 Overflowing Multiple Lines

This case explores the possibility of overflowing multiple lines at the same time by targeting selected buses. The possibilities to target several lines at the same time are more limited than targeting just one line, and the potential damage to the system is far greater than overflowing one line. If successful, overflowing two or more lines can lead to even more fatal consequences than only overflowing one line, and are considered a severe cyber-attack. Only one scenario will be considered for overflowing lines 7 and 15. Results are illustrated in Table 3.3. By targeting buses 3, 5, 6, and 9 in the system, both lines 7 and 15 would be overflowed by 5% of the maximum allowable power flow for those lines. This case verifies that the BMILP model used could successfully be used to attack and overflow the system.
Table 3.3. False Data Injections to overflow Lines 7 and 15

<table>
<thead>
<tr>
<th>Case 1</th>
<th>(\tau_2 = 0.2, \alpha_L = 1.05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targeted Buses</td>
<td>(\Delta D)</td>
</tr>
<tr>
<td>3</td>
<td>-0.64 MW</td>
</tr>
<tr>
<td>5</td>
<td>-0.97 MW</td>
</tr>
<tr>
<td>6</td>
<td>-2.24 MW</td>
</tr>
<tr>
<td>9</td>
<td>3.85 MW</td>
</tr>
<tr>
<td>P7</td>
<td>-26.25 MW</td>
</tr>
<tr>
<td>P15</td>
<td>26.25 MW</td>
</tr>
<tr>
<td>Cost (no attack)</td>
<td>$15,041.43</td>
</tr>
<tr>
<td>Cost (with attack)</td>
<td>$15,379.25</td>
</tr>
</tbody>
</table>

3.3 Conclusion

In this chapter, the bi-level mixed-integer linear programming model proposed in [3] was used to model a cyber-attack on the IEEE 14 bus benchmark. Case studies validate this method for overflowing single lines in this system. The attacker would also not be limited to one specific set of buses to be targeted. Several cases were considered where the attacker would have two different sets of buses.

In the next chapter, two state estimation algorithms capable of approximating the voltage magnitudes and angles at each bus during normal operation will be investigated. These results will serve as a baseline to later compare to the FDI attack results, providing a foundation for the novel algorithm that will be developed in Chapter 5. This novel algorithm will be able to detect FDI attacks and will be tuned to do so by establishing a comparison between normal operation and FDI attack operation. The FDI attack operation was seen in this chapter and the normal operation will be seen in the next chapter.
Chapter 4  
State estimation

4.1 Weighted Least Squares (WLS)

In power systems, in order to estimate the voltage magnitude and angle at each bus, as well as their active and reactive powers, state estimation is used. The power grid uses data from different buses spread through the network to estimate the state of the power grid. Maximum likelihood estimation is used to estimate the unknown parameters of each state by the use of probability density functions at each measurement. The solution to this problem is referred to as the Weighted Least Squares (WLS) estimator for $x$ [28].

The state variables that we are trying to estimate are the voltages and angles at each bus. Therefore, the matrix $x^T$ is composed of the angles of all the buses (except the slack bus since it is fixed at $0^\circ$) and all voltages (assuming the slack bus voltage to be 1).

$$x^T = [\theta_2, \theta_3, ..., \theta_N, V_1, V_2, ..., V_N]$$  \hspace{1cm} (4.1)

The equations for calculating the active and reactive power injection measurements are given below in Equations 4.2 and 4.3, respectively.

$$P_i = V_i \sum_{j=1}^{N} V_j (G_{ij} \cos(\theta_{ij}) + B_{ij} \sin(\theta_{ij}))$$  \hspace{1cm} (4.2)

$$Q_i = V_i \sum_{j=1}^{N} V_j (G_{ij} \sin(\theta_{ij}) - B_{ij} \cos(\theta_{ij}))$$  \hspace{1cm} (4.3)

The Weighted Least Squares algorithm is given below step by step.
Weighted Least-Square Estimation Algorithm

1. Start the iterations, set iteration index \( k = 0 \),

2. Initialize the state vector \( x^k \), e.g., \( v_i = 1 \angle 0 \),

3. Calculate the gain matrix, \( G(x^k) \),

4. Calculate the right-hand side of Equation C.4, e.g.,

\[
[J(x^k)]^T R^{-1} (z - h(x^k))
\]

5. Solve for \( \Delta x^{k+1} \) using normal equation,

6. Test for convergence, e.g., \( \text{Max} |\Delta x^k| \leq \epsilon \)?

7. If no, update \( x^{k+1} = x^k + \Delta x^{k+1}, k = k + 1 \) and go to step 3.

The line measurements for a given system can also be calculated using Equations 4.4 and 4.5, respectively.

\[
P_{ij} = V_i^2 (g_{si} + g_{ij}) - V_i V_j (g_{ij} \cos(\theta_{ij}) + b_{ij} \sin(\theta_{ij}))) \tag{4.4}
\]

\[
Q_{ij} = -V_i^2 (b_{si} + b_{ij}) - V_i V_j (g_{ij} \cos(\theta_{ij}) - b_{ij} \sin(\theta_{ij}))) \tag{4.5}
\]

It is noted that \( G_{ij} \) and \( B_{ij} \) are \( ij \)th elements of bus admittance matrix, and \( g_{ij} \) and \( b_{ij} \) are \( ij \)th elements of line impedance.

Lastly, the Jacobian matrix can be calculated by taking the partial derivative of each
measurement with respect to each voltage angle and magnitude in Equation 4.6

\[
J = \begin{bmatrix}
  \frac{\partial P_{\text{inj}}}{\partial \theta} & \frac{\partial P_{\text{inj}}}{\partial V} \\
  \frac{\partial P_{\text{flow}}}{\partial \theta} & \frac{\partial P_{\text{flow}}}{\partial V} \\
  \frac{\partial Q_{\text{inj}}}{\partial \theta} & \frac{\partial Q_{\text{inj}}}{\partial V} \\
  \frac{\partial Q_{\text{flow}}}{\partial \theta} & \frac{\partial Q_{\text{flow}}}{\partial V} \\
  \frac{\partial I_{\text{flow}}}{\partial \theta} & \frac{\partial I_{\text{flow}}}{\partial V} \\
  \frac{\partial V}{\partial \theta} & \frac{\partial V}{\partial V}
\end{bmatrix}
\]  \tag{4.6}

### 4.1.1 Weighted Least Squares Estimation Results

Following the steps described in 4.1 for the Weighted Least Squares algorithm, the voltage magnitudes and angles can be estimated for the IEEE 14-bus system. This iterative process yields accurate results in a small number of steps. For this system, the results settle at 5 iterations, where the values stabilize within 0.01%. The results for the Weighted Least Squares Estimation method are found below in Table 4.1. The results were obtained using Matlab. The code can be found in Listing A.2. in Appendix A.
Table 4.1. WLS Estimation Results

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0449</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>1.0315</td>
<td>-0.0898</td>
</tr>
<tr>
<td>3</td>
<td>0.9998</td>
<td>-0.2283</td>
</tr>
<tr>
<td>4</td>
<td>1.0177</td>
<td>-0.1890</td>
</tr>
<tr>
<td>5</td>
<td>1.0177</td>
<td>-0.1610</td>
</tr>
<tr>
<td>6</td>
<td>1.0171</td>
<td>-0.2678</td>
</tr>
<tr>
<td>7</td>
<td>1.0338</td>
<td>-0.2433</td>
</tr>
<tr>
<td>8</td>
<td>1.0684</td>
<td>-0.2430</td>
</tr>
<tr>
<td>9</td>
<td>1.0178</td>
<td>-0.2722</td>
</tr>
<tr>
<td>10</td>
<td>1.0138</td>
<td>-0.2780</td>
</tr>
<tr>
<td>11</td>
<td>1.0148</td>
<td>-0.2768</td>
</tr>
<tr>
<td>12</td>
<td>0.9985</td>
<td>-0.2954</td>
</tr>
<tr>
<td>13</td>
<td>1.0022</td>
<td>-0.2855</td>
</tr>
<tr>
<td>14</td>
<td>1.0118</td>
<td>-0.2903</td>
</tr>
</tbody>
</table>

4.2 Discrete Kalman Filter (DKF)

4.2.1 Introduction

Kalman filtering is a state estimation technique invented in 1960 by Rudolf E. Kálmán [29]. This technique allows its user to extract useful information from noisy data while remaining computationally inexpensive. It is used in many application areas including signal processing, and power systems [30–33]. In [34], state estimation is applied to power systems in estimating the voltages and currents at each bus using static and dynamic Kalman filtering techniques. There are several Kalman Filtering techniques, for this thesis, we will only be exploring the Discrete Kalman Filter (DKF). In [35], it is proposed a data-based anomaly detection algorithm to detect FDI attacks towards SCADA measurements by applying density-based analysis techniques after dimensionality reduction for the IEEE 14-bus system. The state estimation techniques used are similar to the ones used in this thesis by using static state estimation forecasting.

The Discrete Kalman Filter estimates the value of a process by using a form of
feedback control. There are two sets of equations in a Discrete Kalman Filter, time update equations and measurement update equations. The time update equations estimate the future values of the current states and the error covariance estimates to obtain the a priori estimates for the next time step. A priori indicates the step prior to the estimation process while a posteriori indicates the step after the estimation. The measurement update equations account for the feedback, which allows the incorporation of a new measurement into the a priori estimate, to obtain a more accurate a posteriori estimate [36].

The time update equations act as predictor equations, while the measurement update equations perform corrections to those predictions. This predictor-corrector relationship between the two sets of equations addresses the problem of trying to estimate the state $x \in \mathbb{R}$ of a discrete time-controlled process that is governed by the linear stochastic difference equation [36]. In a power system network, the voltage and its angle are considered as static states [37]. In [38], an IEEE-14 bus test system is used for finding the different static states using the Kalman filtering algorithm techniques. These algorithms are compared in [39] by evaluating the stability of the power network in a case study.

### 4.2.2 Implementing the algorithm

Kalman filters are used to estimate certain variables in a system that are not being measured directly, based on linear equations in a system. The process model evaluates the change of the state from time $k-1$ to time $k$ as:

$$x_k = Fx_{k-1} + w_{k-1} \quad (4.7)$$

where $F$ is the state transition matrix used on the previous value of the state vector, $x_{k-1}$, $B$ is the control-input matrix applied to the control vector $u_{k-1}$, and $w_{k-1}$ is the process noise vector which is assumed to be zero-mean Gaussian with the covariance $Q$, i.e., $w_{k1} \sim \mathcal{N}(0, Q)$.

The measurement vector, $z_k$, can be calculated based on the measurement matrix, $H$, and the measurement noise vector, $v_k$, which is assumed to be zero-mean Gaussian with the covariance $R$, i.e., $v_k \sim \mathcal{N}(0, R)$.

$$z_k = Hx_k + v_k \quad (4.8)$$
The Discrete Kalman Filter was chosen over other types of Kalman filters due to the amount of research available towards the power systems applications [40] [39]. When comparing the different methods, the DFK method was mathematically easier to incorporate to my detection algorithm over the other types of Kalman filters, while remaining computationally affordable and estimating with minimal error.

The Discrete Kalman Filter algorithm is given below step by step based on the algorithm presented by Rudolf E. Kálmán, adapted for power flow [40].

Power Flow Modified Discrete Kalman Filter Algorithm for IEEE 14-bus system

1. The dynamic system is given by the following equations:

\[
x_k = Fx_{k-1} + w_{k-1} \quad (27 \times 1 \text{ matrix}) \tag{4.9}
\]

\[
y_k = H_kx_k + v_k \quad (32 \times 1 \text{ matrix}) \tag{4.10}
\]

The matrix \( F \) is the Identity matrix as a 27x27 matrix, found below.

\[
F = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1
\end{bmatrix}
\]

The matrix \( x_{k-1} \) is the measurement matrix at step k-1, and \( w_{k-1} \) is the system process noise in a 27x1 matrix. Both matrices are represented below, and the variables \( a, b, c, \) and \( d \), in matrix \( w_{k-1} \), represent small random numbers.

\[
x_{k-1} = \begin{bmatrix}
1 \\
1 \\
\vdots \\
1 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix}, \quad w_{k-1} = \begin{bmatrix}
a \\
b \\
c \\
\vdots \\
d
\end{bmatrix}
\]
The matrix $H_k$ is the Jacobian matrix, with a detailed explanation in Chapter 2, containing the derivatives of the measurements with respect to the voltage and angles of the system in a $32 \times 27$ matrix and $v_k$ is assumed 0. Both matrices are represented below.

$$ H_k = \begin{bmatrix} \frac{\partial P_2}{\partial \theta_2} & \cdots & \frac{\partial P_2}{\partial \theta_N} & \frac{\partial P_2}{\partial V_2} & \cdots & \frac{\partial P_2}{\partial V_N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_N}{\partial \theta_2} & \cdots & \frac{\partial P_N}{\partial \theta_N} & \frac{\partial P_N}{\partial V_2} & \cdots & \frac{\partial P_N}{\partial V_N} \\ \frac{\partial Q_2}{\partial \theta_2} & \cdots & \frac{\partial Q_2}{\partial \theta_N} & \frac{\partial Q_2}{\partial V_2} & \cdots & \frac{\partial Q_2}{\partial V_N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Q_N}{\partial \theta_2} & \cdots & \frac{\partial Q_N}{\partial \theta_N} & \frac{\partial Q_N}{\partial V_2} & \cdots & \frac{\partial Q_N}{\partial V_N} \end{bmatrix}, \quad v_k = [0] $$

2. The Kalman filter is initialized as follows:

$$ \dot{x}_0^+ = E(x_0) \quad (27 \times 1 \text{ matrix}) \quad (4.11) $$

$$ P_0^+ = E[(x_0 - \dot{x}_0^+)(x_0 - \dot{x}_0^+)^T] \quad (27 \times 27 \text{ matrix}) \quad (4.12) $$

The matrix $x_0$ is initiated as a $27 \times 1$ matrix (1 for voltage magnitude and 0 for voltage angle) and $\dot{x}_0^+$ is initiated with small random numbers as a $27 \times 1$ matrix. Both matrices are represented below, and the variables $e, f, g, \text{ and } h$, in matrix $\dot{x}_0+$, represent small random numbers.

$$ x_0 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \dot{x}_0^+ = \begin{bmatrix} e \\ f \\ g \\ \vdots \\ h \end{bmatrix} $$
3. The Kalman filter is given by the following equations, which are computed for each time step $k = 1, 2, \ldots, n$

$$P_k^- = P_{k-1}^+ + Q_{k-1} \quad \text{(27x27 matrix)} \quad (4.13)$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad \text{(27x32 matrix)} \quad (4.14)$$

The matrix $Q$ is the process covariance 27x27 matrix which is randomized (commonly done in power systems unless the process is known), and $R_k$ is the diagonal matrix associated with uncertainties in the system which is also a small randomized 32x32 matrix. All the entries contained in matrix $Q$ and $R_k$ below, from $i$ to $z$, represent small random numbers.

$$Q = \begin{bmatrix} i & j & \cdots & k \\ l & m & \cdots & n \\ \vdots & \vdots & \ddots & \vdots \\ o & p & \cdots & q \end{bmatrix}, \quad R_k = \begin{bmatrix} r & s & \cdots & t \\ u & v & \cdots & w \\ \vdots & \vdots & \ddots & \vdots \\ x & y & \cdots & z \end{bmatrix}$$

$$\hat{x}_k^- = \hat{x}_{k-1}^+ + w_{k-1} \quad \text{(27x1 matrix)} \quad (4.15)$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k (y_k - H_k \hat{x}_k^-) \quad \text{(27x1 matrix)} \quad (4.16)$$

$$P_k^+ = (I - K_k H_k) P_k^- \quad \text{(27x27 matrix)} \quad (4.17)$$

In Equations 4.9 - 4.17, the $\hat{\,}$ operator , indicates the estimate of the state variable. The superscript $-$, indicates the predicted step or the a priori prediction. And the superscript $+$ indicates the updated or a posteriori estimate [36].

The algorithm above is used to estimate states of a system that have uncertainty. In power systems, the system’s measurements at each bus have uncertainty regarding the accuracy of the measurement, and may also not be able to give a measurement for every single time step. For this reason, we use state estimators, such as the Kalman Filter, to create an estimate that considers the inevitable uncertainty in the system.

For our purposes, the vector of state variables, $x_k$, will contain the voltage magnitudes and angles that we are trying to estimate. By initiating the values of the voltage magnitudes and angles in the system, $\hat{x}_o^+$, to 1 for magnitude and 0 for angle, and by estimating the amount of error the measurements contain, a first estimation can be acquired that can later be tuned. The covariance error ($P_k$) for those measurements is
then initiated following Equation 4.11 - 4.12.

The Jacobian matrix, $H$, is then calculated based on the power flow equations in Chapter 2, followed by the state variables that were initiated to be 0 and 1. The Kalman Gain is calculated based on Equation 4.14 and the rest of the equations are calculated based on the variables that we have already calculated. Thus yielding an accurate estimate of the state variables that we are trying to estimate.

To initialize the estimation process, an initial guess of state estimate is needed, $\hat{x}_0^+$, and the initial guess of the error covariance matrix, $P_0^+$. Together with $Q$ and $R$, $\hat{x}_0^+$ and $P_0^+$ are vital to obtaining the desired performance. Lastly, the implementation of both the prediction and update stages can be performed in an iterative process at each time step, after the estimate has been initialized.

The system used for the Kalman filter is assumed to be linear and is expressed based on the system variables $F$, $B$, and $H$, and the process and measurement noise are assumed to be Gaussian. It will provide an optimal estimate for a system only if the assumptions are satisfied [41].

### 4.2.3 DKF Results

Following the steps described in 4.2 for the Discrete Kalman Filter algorithm, the voltage magnitudes and angles can be estimated for the IEEE 14-bus system. This iterative process yields accurate results in a small number of steps. For this system, the results settle at 5 iterations, where the values no longer change between iterations. The number of steps is comparable to the Weighted Least Squares method. The results for the DKF Estimation method are found below in Table 4.2. The results were obtained using Matlab. The code can be found in Listing A.1. in Appendix A.
Table 4.2. Discrete Kalman Filter Estimation Results

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0971</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>1.0788</td>
<td>-0.0967</td>
</tr>
<tr>
<td>3</td>
<td>1.0368</td>
<td>-0.2402</td>
</tr>
<tr>
<td>4</td>
<td>1.0359</td>
<td>-0.1991</td>
</tr>
<tr>
<td>5</td>
<td>1.0480</td>
<td>-0.1711</td>
</tr>
<tr>
<td>6</td>
<td>1.0443</td>
<td>-0.2844</td>
</tr>
<tr>
<td>7</td>
<td>1.0518</td>
<td>-0.2573</td>
</tr>
<tr>
<td>8</td>
<td>1.0876</td>
<td>-0.2571</td>
</tr>
<tr>
<td>9</td>
<td>1.0346</td>
<td>-0.2881</td>
</tr>
<tr>
<td>10</td>
<td>1.0313</td>
<td>-0.2941</td>
</tr>
<tr>
<td>11</td>
<td>1.0362</td>
<td>-0.2931</td>
</tr>
<tr>
<td>12</td>
<td>1.0264</td>
<td>-0.3128</td>
</tr>
<tr>
<td>13</td>
<td>1.0271</td>
<td>-0.3022</td>
</tr>
<tr>
<td>14</td>
<td>1.0226</td>
<td>-0.3069</td>
</tr>
</tbody>
</table>

Now that the results for the IEEE 14-bus system during normal operation have been determined, a relationship between an FDI attack state and normal operation can be drawn. In the next chapter, I will combine the theoretical framework and quantitative results to establish a threshold to detect FDI attacks.
Chapter 5  
Detection algorithm and Results

5.1 Introduction

In this chapter, a novel detection algorithm will be developed to detect FDI attacks. The algorithm will establish if the system is under attack if a calculated threshold is exceeded. Based on the results from previous chapters, two sets of results will be used, during normal operation and during an FDI attack, to calculate a reasonable threshold. The first set with unaltered data will be referred to as 'Good Data' and does not contain any false data injections. The second set with false data injections will be referred to as 'Bad Data'.

To successfully detect the false data injections in the Bad Data set, it is necessary to determine a baseline of the expected results with the Good Data set. This set will be used to test that the original measurements can be reconstructed from the estimated results. The estimated results from the Kalman Filter, $x_{est}$, are used with the Jacobian matrix, $H$, to obtain the reconstructed measurements $z_{new}$ in Equation 5.1.

$$z_{new} = H x_{est}$$ (5.1)

Once the reconstructed measurements from the Good Data set have been calculated, we can calculate the Euclidean length between the reconstructed measurements and the original measurements. This calculation determines how close two vectors of the same length are and helps to establish a threshold to identify if the measurements contain false data or not. The Euclidean length equation used can be found below in Equation 5.2.

$$Euclidean \ length = \sqrt{|z_1 - z_{new,1}|^2 + |z_2 - z_{new,2}|^2 + ... + |z_n - z_{new,n}|^2}$$ (5.2)
Novel FDI attack detection algorithm

The novel algorithm begins by estimating the voltage magnitudes and angles that are unknown to the system, based on the known measurements and the system characteristics, such as the impedances of the lines. The estimation process is performed by using a Kalman Filter, and its results are tuned by comparing the results from the WLS algorithm for the system. Once the system has been tuned, the Jacobian matrix, calculated in the Kalman Filter part of the algorithm, is used to reconstruct all the measurements in the system, including the known and unknown measurements following Equation 5.1. The reconstructed measurements are compared to the original measurements of the system, and the Euclidean length between the two vectors is calculated. If the system is working under normal conditions, the difference between the reconstructed measurements and the original ones should be very small, since there is no outside noise in the system. When the system is under an FDI attack, the difference would grow much bigger due to the injections of power in the system that causes an increase or decrease in the voltage magnitudes and angles. The novel algorithm is shown below in Figure 5.1.

Figure 5.1. Novel algorithm for detecting FDI attacks
The original measurements for line flow measurements, bus injection measurements, as well as voltage measurements can be found below in Tables 5.1, 5.2, and 5.3, respectively. These measurements were obtained from [42].

In this chapter, we will be working with 32 measurements to estimate 27 voltage magnitudes and angles. In order to calculate all 27 voltage magnitudes and angles without using state estimation, a total of 68 measurements would be needed. The higher the number of measurements in the system, the more accurate the estimated results will be.

**Table 5.1.** Line Flow Measurements

<table>
<thead>
<tr>
<th>Bus No. (From)</th>
<th>Bus No. (To)</th>
<th>Active Power (P)</th>
<th>Reactive Power (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.56771</td>
<td>-0.20378</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.73161</td>
<td>0.03568</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.27870</td>
<td>-0.09478</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>0.08801</td>
<td>0.03591</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>0.13670</td>
<td>0.07187</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>0.12464</td>
<td>0.02372</td>
</tr>
</tbody>
</table>

**Table 5.2.** Bus Injection Measurements

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-0.94200</td>
<td>0.04393</td>
<td>5</td>
<td>-0.07600</td>
<td>0.01600</td>
</tr>
<tr>
<td>6</td>
<td>-0.11200</td>
<td>0.04718</td>
<td>7</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>8</td>
<td>0.00000</td>
<td>0.17357</td>
<td>9</td>
<td>-0.29500</td>
<td>-0.16600</td>
</tr>
<tr>
<td>10</td>
<td>-0.09000</td>
<td>-0.05800</td>
<td>11</td>
<td>-0.03500</td>
<td>-0.01800</td>
</tr>
<tr>
<td>13</td>
<td>-0.13500</td>
<td>-0.05800</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.3.** Voltage Measurements

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Bus No.</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.01870</td>
<td>14</td>
<td>1.03700</td>
</tr>
</tbody>
</table>
Based on these original measurements, a Kalman Filter can be established to estimate the voltage magnitudes and angles at each bus of this IEEE 14 bus system.

The novel algorithm will now be tested on the two sets of data to establish a reasonable threshold to detect FDI attacks.

5.2 Reconstructed Good Data

To successfully reconstruct the original measurements from the Good Data set, it is necessary to first obtain the estimated voltage magnitudes and angles from the Kalman filter. The estimated values, $x_{est}$, are found below in Chapter 4 in Table 4.2.

Once the estimated values have been found using the techniques explained in Chapter 4.2 with the original measurements, the reconstructed measurements for the Good Data set can be calculated by using Equation 5.1. These reconstructed measurements can be found in Appendix D in Tables D.1, D.2, and D.3.

Using Equation 5.2, the Euclidean length between the reconstructed measurements vector, $z_{new}$, and the original measurements, $z$, is 0.0866. To allow for some variability in the accuracy of the measurements, a 5% increase of the calculated Euclidean length, 0.091, will be used to establish a relationship between Good Data and Bad Data. If the Euclidean length of the system is below 0.091, the system detects that there are no false data injections. If it is higher than 0.091, false data injections are detected in the system.

In Figure 5.2, a graph of the original measurements is shown compared to the reconstructed ones using the Kalman Filter and the Jacobian matrix from Equation 5.1. The Kalman Filter is able to accurately estimate all the measurements with only 1.67% error compared to the actual measurements. This estimate will serve as a baseline when the system is working under normal conditions. We will now reconstruct the measurements from Bad Data and see if the system is able to estimate the measurements accurately.
5.3 Reconstructed Bad Data

The reconstructed Bad Data follows the same process as the reconstructed Good Data, with the only difference being that the original measurements vector, \( z \), has been altered. To come up with the necessary false data injections to alter the \( z \) vector, the Attack Model in Chapter 3 is used. Using this model, false data injections, \( \Delta D \), are needed in the specified buses to overflow line 7.

Two sets of buses can be used to inject the false data injections in the system to overflow line 7 without being detected by the DC state estimation algorithm in the Attack Model. The buses and precise false data injections needed are found in Table 5.4 below. All false data injection are Active Power injections in the system (e.g. \( \Delta p_6 = -2.24 \) MW for Case 1)
Table 5.4. False Data Injections to overflow only Line 7

<table>
<thead>
<tr>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_2 = 0.2, \alpha_L = 1.05 )</td>
<td>( \tau_2 = 0.2, \alpha_L = 1.05 )</td>
</tr>
<tr>
<td>Targeted Buses</td>
<td>( \Delta D )</td>
</tr>
<tr>
<td>6</td>
<td>-2.24 MW</td>
</tr>
<tr>
<td>9</td>
<td>4.31 MW</td>
</tr>
<tr>
<td>13</td>
<td>-2.07 MW</td>
</tr>
<tr>
<td>P7</td>
<td>-26.25 MW</td>
</tr>
</tbody>
</table>

To obtain the original measurements for the Bad Data set, \( z_{BadData} \), the original measurements need to be added to the false data injections as shown in Equation 5.3.

\[
z_{BadData} = z + \Delta D
\]  

\[(5.3)\]

5.3.1 Reconstructed Bad Data: Case 1

Once the original measurements for the Bad Data set have been calculated considering the false data injections in Case 1 in Table 5.4, the values for the voltage magnitudes and angles can be estimated using the Kalman Filter and are shown below in Table D.4. The reconstructed measurements for the Bad Data set can be calculated by using Equation 5.1 with the estimated voltage magnitudes and angles, \( x_{est, BadData} \), instead of \( x_{est} \) as well as the new Jacobian matrix, \( H_{BadData} \), since its value depends on the partial derivatives of the original measurements used and the measurements have been altered by the false data injections \( \Delta D \). The reconstructed measurements for the Bad Data set can be found below in Tables D.5, D.6, and D.7.

Using Equation 5.2, the Euclidean length between the reconstructed measurements vector, \( z_{new, BadData} \), and the original measurements, \( z_{BadData} \), is 1.1095.

In Figure 5.3, a graph of the original measurements is shown compared to the reconstructed ones using the Bad Data from the FDI attack in Case 1. The percent error between the reconstructed measurements for the Bad Data set and the original false data measurements is 12.82 % for this system. The reconstructed measurements in this system clearly show that when the FDI attack from Case 1 is used, the system is not able to estimate the measurements accurately. This is evident in the Power measurements (P) in Figure 5.3, as the blue bars (original measurement) and the orange ones (reconstructed
Bad Data Case 1) do not align at all. We will now explore if this is the case only for this set of attacked buses, or if it is true for other types of attacks.

![Figure 5.3. Original (blue) vs Reconstructed Bad Data 1 measurements (orange)](image)

### 5.3.2 Reconstructed Bad Data: Case 2

Once the original measurements for the Bad Data set have been calculated considering the false data injections in Case 2 in Table D.8, the values for the voltage magnitudes and angles can be estimated using the Kalman Filter and are shown below in Table D.4. The reconstructed measurements for the Bad Data set can be calculated by using Equation 5.1 with the estimated voltage magnitudes and angles, \(x_{est,BadData2}\), instead of \(x_{est}\) as well as the new Jacobian matrix, \(H_{BadData2}\), as it was calculated in 5.3.1 for Case 1. The reconstructed measurements for the Bad Data set can be found below in Tables D.9, D.10, and D.11.
Using Equation 5.2, the Euclidean length between the reconstructed measurements vector, $z_{new, BadData2}$, and the original measurements, $z_{BadData2}$, is 0.2183.

In Figure 5.4, a comparison between the original and reconstructed measurements using the Bad Data from the FDI attack in Case 2 is shown. The percent error between the reconstructed measurements for the Bad Data set and the original false data measurements is 5.42 % for this system. The reconstructed measurements in this system, as it was in the Bad Data set Case 1, clearly show that when the FDI attack from Case 2 is used, the system is not able to estimate the measurements accurately. This is evident as well in the Power measurements (P) in Figure 5.3, as the blue bars (original measurement) and the orange ones (reconstructed Bad Data Case 2) do not align at all. Since we are injecting active power into the system via the false data injections, the power measurements are more affected compared to other reconstructed measurements. This explains the big differences between power measurements, and why the rest of the measurements are much closer in value.
5.4 Conclusion

Developing an algorithm to detect FDI attacks in a power network requires first to come up with an attack model to detect. By implementing the power flow methods introduced in Chapter 2 into the attack model designed by [3], the minimum power injections for an FDI attack to the IEEE 14-bus system was calculated. Kalman Filters were used to estimate the voltage and angle magnitudes throughout the system based on a specific set of randomly selected measurements. Combining and comparing the results during an attack-state and normal operation allowed me to establish a threshold to determine when an FDI attack is currently happening in a system.

Two different sets of buses were used in the modeled FDI attack, which both targeted transmission line 7, between buses 4 and 5 in the IEEE 14-bus system. Transmission line 7 is more vulnerable than the rest of the lines since it carries the maximum allowable amount of power during normal operation, so it would be easier to make that line overflow compared to other ones that carry less power. Two distinct targeted power injections in the system are drawn from the analysis of the attack model based on [3] in Chapter 3.

When comparing the two sets of attacks to the system during normal operations, a percent error ranging from 5.48% to 12.82% was seen between each estimated measurement with false data injections and the original measurements. These individual differences alone are not enough to determine if there is an FDI attack in the system or not, but when the Euclidean length or error between all the measurements is calculated, the systems with an FDI attack show a Euclidean length 2.52 to 12.81 times bigger than that of the system during normal operation. Therefore, by tracking the Euclidean length in the measured system, an FDI attack would be detected.

The system is able to detect when there are false data injections in the system by calculating the Euclidean length between the measurements and the reconstructed measurements. The Euclidean length for the Good Data set in section 5.3 in this thesis was 0.0866 which is very low compared to 1.1095 and 0.2183, seen in the Bad Data sets in sections 5.3 and 5.4, respectively. This implies that the Kalman Filter used in the Good Data set is very accurate, while the Kalman Filters when there are false data injections in the Bad Data sets is not able to estimate the measurements as well.

This algorithm will detect false data injections in a system that are sufficiently great to make a line or multiple lines overflow. False data injections that are small enough to only increase the Euclidean length of the system by less than 5% would not be detected. This is acceptable since injections that are less than 5% would not make a line overflow.
in this IEEE 14-bus system.

In this thesis, we have seen increases of the Euclidean length from 2.52 to 12.81 times compared to the system during normal operation. A 5% increase with respect to the system during normal operation can be perceived as a very small margin when compared to FDI attacks results. But it is worth noting, that the results obtained only apply to the IEEE 14-bus system, while the algorithm can be applied to any system. There could be a scenario where the attacker is able to come up with a very optimized attack that is capable of overflowing a line with a much smaller Euclidean length. For this type of attack, having a small margin, such as 5%, could mean detecting an FDI attack or not. A bigger study would need to be done with many different systems to determine if 5% is an appropriate margin to detect FDI attacks or if the percentage should be increased or decreased. This opens the door for future research to expand on reasonable margins for cyber-attack detectability. A 5% Euclidean length in this specific IEEE 14-bus system ensures that an FDI attack would be detected, but could differ in a different system since it is a value that was obtained empirically.

We can conclude that the designed system can detect false data injections attacks in this system, and it was proven by testing several attacks that targeted different sets of buses on the IEEE 14 bus system, which were detected by a wide margin, compared to the attack-free results.

It must be mentioned that there could be a scenario in a different system where an FDI attack would cause less than a 5% increase in the error between the reconstructed measurements and the original ones. In such a case, the FDI attack could bypass the DC-state estimation algorithms as well as my novel AC state estimation algorithm. This scenario is very unlikely due to the limits set to trigger the system to identify an FDI attack but remains theoretically possible.

Modern-day power systems could benefit from using the methods presented in this thesis to detect FDI attacks on their power network. More research is needed in order to be able to not only detect FDI attacks, as described in this thesis but other potential types of cyber-attacks on the power grid. As cyber-attacks become more sophisticated, it is imperative to protect power grids and ensure that electricity stays available and reliable throughout the years.
Appendix A
Matlab Code

clear all
clc

mpc = case14;
Measurements = runpf(mpc)

% Branch information
Data_1 = mpc.branch;

n_b = length(mpc.bus);
n_l = length(mpc.branch);

Ybus = zeros(n_b, n_b);

for i = 1:n_l
    Ybus(Data_1(i,1), Data_1(i,2)) = -1/(Data_1(i,3) + j*Data_1(i,4));
    Ybus(Data_1(i,2), Data_1(i,1)) = Ybus(Data_1(i,1), Data_1(i,2));
end

for i = 1:n_b
    Ybus(i,i) = -sum(Ybus(i,:));
end

Ybus;

G = real(Ybus);
B = imag(Ybus);
g = zeros(n_b, n_b);
%%Non-diagonal G terms
for j = 1: n_b
    for i = 1: n_b
        if i == j
            %
        else
            g(i, j) = abs(G(i, j));
        end
    end
    end
    g = triu(g);

%%Non-diagonal B terms
for j = 1: n_b
    for i = 1: n_b
        if i == j
            %
        else
            b(i, j) = -abs((B(i, j)));
        end
    end
    end
    b = triu(b);

% Measurements
p12 = 1.56771;
p23 = 0.73161;
p47 = 0.2787;
p914 = 0.08801;
p613 = 0.1367;
p612 = 0.12464;
p3 = -0.942;
p5 = -0.076;
p6 = -0.112;
p7 = 0;
p8 = 0;
p9 = -0.295;
p10 = -0.09;
p11 = -0.035;
p13 = -0.135;
q12 = -0.20378;
q23 = 0.03568;
q47 = -0.09478;
q914 = 0.03591;
q613 = 0.07187;
q612 = 0.02372;
q3 = 0.04393;
q5 = 0.016;
q6 = 0.04718;
q7 = 0;
q8 = 0.17357;
q9 = -0.166;
q10 = -0.058;
q11 = -0.018;
q13 = -0.058;
V4 = 1.0187;
V14 = 1.037;

z = [1.56771; 0.73161; 0.2787; 0.08801; 0.1367; 0.12464; -0.942; -0.076; -0.112; 0;
     0; -0.295; -0.09; -0.035; -0.135; -0.20378; 0.03568; -0.09478; 0.03591; 0.07187;
     0.02372; 0.04393; 0.016; 0.04718; 0; 0.17357; -0.166; -0.058; -0.018; -0.058; 1.0187;
     1.037];

z_actual = z

x_k = [zeros(1,13) ones(1,14)]';

e_k_initial = x_k - [0.2*ones(1,13) 1.1*ones(1,14)]';

Q = xlsread('Q_matrix')
P_k = cov(e_k_initial*e_k_initial');
P_pre_pri = P_k + Q;
x_est_pri = x_k;

% Jacobian
n = 5; % iteration number

% x=[t2, t3,...t(n_b-1),V1, V2,... Vn_b]
for i = 1:(n_b-1)
\[ x_{0_t}(i,1) = 0; \]
\[ \text{end} \]
\[ \text{for } i = 1: n_b \]
\[ x_{0_V}(i,1) = 1; \]
\[ \text{end} \]
\[ X0 = [x_{0_t}; x_{0_V}]; \]
\[ X_es = \text{zeros}(2*(n_b)-1,n); \]
\[ Z_es = \text{zeros}(\text{length}(z),n); \]
\[ \text{for } RT = 1: n \]
\[ t1 = 0; \]
\[ \text{for } i = 1: (n_b - 1) \]
\[ t_n(i) = X0(i); \]
\[ \text{end} \]
\[ \text{for } i = 1: n_b \]
\[ V(i) = X0(n_b - 1 + i); \]
\[ \text{end} \]
\[ V = [V(1:n_b)]; \]
\[ t = [0, t_n]; \]
\[ \% \text{Calculating } dPflow/dt, dPflow/dV, dQflow/dt \text{ and } dQflow/dV \]
\[ \text{row_counter} = 1; \]
\[ \text{for } k = [1; 2], [2; 3], [4; 7], [9; 14], [6; 13], [6; 12] \]
\[ \text{select} = [k]; \]
\[ i = \text{select}(1); \]
\[ j = \text{select}(2); \]
\[ \text{for } m = 1: n_b \]
\[ \text{if } m == i \]
\[ P_{flow_t} = V(i) \times V(j) \times (g(i,j) \times \sin(t(i)-t(j)) - b(i,j) \times \cos(t(i)-t(j))) \];
\[ P_{flow_V} = -V(j) \times (g(i,j) \times \cos(t(i)-t(j))) + b(i,j) \times \sin(t(i)-t(j))) + 2 \times (g(i,j)) \times V(i); \]
\[ Q_{flow_t} = -V(i) \times V(j) \times (g(i,j) \times \cos(t(i)-t(j))) + b(i,j) \times \sin(t(i)-t(j)) \]
\[ Q_{\text{flow}_V} = -V(j) \left( g(i,j) \sin(t(i)-t(j)) - b(i,j) \cos(t(i)-t(j)) \right) - 2 \cdot (b(i,j)) \cdot V(i); \]

else

\[ P_{\text{flow}_t} = -V(i) \cdot V(j) \left( g(i,j) \sin(t(i)-t(j)) - b(i,j) \cos(t(i)-t(j)) \right); \]

\[ P_{\text{flow}_V} = -V(i) \left( g(i,j) \cos(t(i)-t(j)) + b(i,j) \sin(t(i)-t(j)) \right); \]

\[ Q_{\text{flow}_t} = V(i) \cdot V(j) \left( g(i,j) \cos(t(i)-t(j)) + b(i,j) \sin(t(i)-t(j)) \right); \]

\[ Q_{\text{flow}_V} = -V(i) \left( g(i,j) \sin(t(i)-t(j)) - b(i,j) \cos(t(i)-t(j)) \right); \]

else

\[ P_{\text{flow}_t} = 0; \]

\[ P_{\text{flow}_V} = 0; \]

\[ Q_{\text{flow}_t} = 0; \]

\[ Q_{\text{flow}_V} = 0; \]

end

end

\[ H1_{\text{Pflow}_t}(\text{row}_\text{counter},m) = P_{\text{flow}_t}; \]

\[ H1_{\text{Pflow}_V}(\text{row}_\text{counter},m) = P_{\text{flow}_V}; \]

\[ H1_{\text{Qflow}_t}(\text{row}_\text{counter},m) = Q_{\text{flow}_t}; \]

\[ H1_{\text{Qflow}_V}(\text{row}_\text{counter},m) = Q_{\text{flow}_V}; \]

end

\[ \text{row}_\text{counter} = \text{row}_\text{counter} + 1; \]

end

\% Calculating \( dP_{\text{inj}}/dt \), \( dP_{\text{inj}}/dV \), \( dQ_{\text{inj}}/dt \), and \( dQ_{\text{inj}}/dV \)

\[ m = 1; \]

for \( i = [3,5,6,7,8,9,10,11,13] \)

for \( j = 1: n_b \)

if \( i = j \)

\[ P_{\text{inj}_t}(m,j) = V(i) \cdot V(j) \left( g(i,j) \sin(t(i)-t(j)) - b(i,j) \cos(t(i)-t(j)) \right); \]

\[ P_{\text{inj}_V}(m,j) = V(i) \left( g(i,j) \cos(t(i)-t(j)) + b(i,j) \sin(t(i)-t(j)) \right); \]

end

end
Q_inj_t(m,j) = V(i)*V(j)*(-G(i,j)*cos(t(i)-t(j))-B(i,j)*sin(t(i)-t(j)))
Q_inj_V(m,j) = V(i)*(G(i,j)*sin(t(i)-t(j))-B(i,j)*cos(t(i)-t(j)))

else %i==j

for k = 1:n_b
    Value_Pinj_t(k,1) = V(i)*V(k)*(-G(i,k)*sin(t(i)-t(k))+B(i,k)*cos(t(i)-t(k)))
    Value_Pinj_V(k,1) = V(k)*(G(i,k)*cos(t(i)-t(k))+B(i,k)*sin(t(i)-t(k)))
end

P_inj_t(m,j) = sum((Value_Pinj_t))-V(i)^2*B(i,i)
P_inj_V(m,j) = sum((Value_Pinj_V))+V(i)*G(i,i)

Q_inj_t(m,j) = sum((Value_Qinj_t))-V(i)^2*G(i,i)
Q_inj_V(m,j) = sum((Value_Qinj_V))-V(i)*B(i,i)

end

end
m=m+1;
end

% Calculating dV/dt and dV/dV

H_V_4 = [zeros(1,16),1,zeros(1,10)];
H_V_14 = [zeros(1,26),1];
H = [H1_Pflow_t(:,2:n_b), H1_Pflow_V;
    P_inj_t(:,2:n_b), P_inj_V;
    H1_Qflow_t(:,2:n_b), H1_Qflow_V;
    Q_inj_t(:,2:n_b), Q_inj_V;
    H_V_4;
    H_V_14];

%% Measurement Covariance
R = eye(size(H,2),size(H,2));
for i = 1:6
    R(i,i) = 0.008^2;
end
for i = 7:15
    R(i,i) = 0.01^2;
end
for i = 16:21
    R(i,i) = 0.008^2;
end
for i = 22:30
    R(i,i) = 0.01^2;
end
R(31,31) = 0.004^2;
R(32,32) = 0.004^2;

%% Gain Matrix
Kalman_Gain = P_pre_pri*H'*inv(H*P_pre_pri*H'+R);
x_est = x_est_pri + Kalman_Gain*(z-H*x_est_pri)
P_k = (eye()-Kalman_Gain*H)*P_pre_pri;
P_pre_pri = P_k + Q;
x_pre_pri = x_est;
Estimation_results(:,RT)=x_est

disp('Iteration and Solution Data:');
disp('| Iteration | t1  | t2  | t3  | t4  | t5  | t6  | t7  | t8  |
|         | t9  | t10 | t11 | t12 | t13 | t14 |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|
for j = 1:RT

disp('
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
');
fprintf(' | %3g |', j);
fprintf(' %6.2 f |', 0);
fprintf(' %6.4 f |', Estimation_results(1,j));
fprintf(' %6.4 f |', Estimation_results(2,j));
fprintf(' %6.4 f |', Estimation_results(3,j));
fprintf(' %6.4 f |', Estimation_results(4,j));
fprintf(' %6.4 f |', Estimation_results(5,j));
fprintf(' %6.4 f |', Estimation_results(6,j));
fprintf(' %6.4 f |', Estimation_results(7,j));
fprintf(' %6.4 f |', Estimation_results(8,j));
fprintf(' %6.4 f |', Estimation_results(9,j));
fprintf(' %6.4 f |', Estimation_results(10,j));
fprintf(' %6.4 f |', Estimation_results(11,j));
fprintf(' %6.4 f |', Estimation_results(12,j));
fprintf(' %6.4 f |', Estimation_results(13,j));
fprintf('
');
end

fprintf('

');
disp('Iteration and Solution Data:');
disp('| Iteration | V1  | V2  | V3  |');
fprintf(' | %6.4f |', Estimation_results(14,j));
fprintf(' %6.4f |', Estimation_results(15,j));
fprintf(' %6.4f |', Estimation_results(16,j));
fprintf(' %6.4f |', Estimation_results(17,j));
fprintf(' %6.4f |', Estimation_results(18,j));
fprintf(' %6.4f |', Estimation_results(19,j));
fprintf('


');
for j = 1:n

disp('----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
');
fprintf(' | %3g |', j);
fprintf(' %6.4 f |', Estimation_results (14 ,j));
fprintf(' %6.4 f |', Estimation_results (15 ,j));
fprintf(' %6.4 f |', Estimation_results (16 ,j));
fprintf(' %6.4 f |', Estimation_results (17 ,j));
fprintf(' %6.4 f |', Estimation_results (18 ,j));
fprintf(' %6.4 f |', Estimation_results (19 ,j));
fprintf(' %6.4 f |', Estimation_results (20 ,j));
fprintf(' %6.4 f |', Estimation_results (21 ,j));
fprintf(' %6.4 f |', Estimation_results (22 ,j));
fprintf(' %6.4 f |', Estimation_results (23 ,j));
fprintf(' %6.4 f |', Estimation_results (24 ,j));
fprintf(' %6.4 f |', Estimation_results (25 ,j));
fprintf(' %6.4 f |', Estimation_results (26 ,j));
fprintf(' %6.4 f |', Estimation_results (27 ,j));
fprintf('
');
end

z_reconstructed_good_data = H* Estimation_results (: ,n);
saved_estimation_good = Estimation_results (: ,5)
norm2_good_data = norm (z- z_reconstructed_good_data )

%%
%
% BAD DATA
%
clearvars -except norm2_good_data Q e_k_initial n
    z_reconstructed_good_data z_actual saved_estimation_good

mpc=case14;
Measurements=runpf(mpc)

% Branch information
Data_1=mpc.branch;
n_b = length(mpc.bus);
n_l = length(mpc.branch);

Ybus = zeros(n_b, n_b);

for i = 1:n_l
    Ybus(Data_1(i,1), Data_1(i,2)) = -1/(Data_1(i,3) + j*Data_1(i,4));
    Ybus(Data_1(i,2), Data_1(i,1)) = Ybus(Data_1(i,1), Data_1(i,2));
end

for i = 1:n_b
    Ybus(i,i) = -sum(Ybus(i,:));
end

Ybus;

G = real(Ybus);
B = imag(Ybus);
g = zeros(n_b, n_b);

%Non-diagonal G terms
for j = 1:n_b
    for i = 1:n_b
        if i == j
            else
                g(i,j) = abs(G(i,j));
        end
    end
    end
    g = triu(g);

%Non-diagonal B terms
for j = 1:n_b
    for i = 1:n_b
        if i == j
            else
                b(i,j) = -abs((B(i,j)));
        end
    end
end
b = triu(b);

%% Measurements
p12 = 1.56771;
p23 = 0.73161;
p47 = 0.2787;
p914 = 0.08801;
p613 = 0.1367;
p612 = 0.12464;
p3 = -0.942;
p5 = -0.076;
p6 = -0.112;
p7 = 0;
p8 = 0;
p9 = -0.295;
p10 = -0.09;
p11 = -0.035;
p13 = -0.135;
q12 = -0.20378;
q23 = 0.03568;
q47 = -0.09478;
q914 = 0.03591;
q613 = 0.07187;
q612 = 0.02372;
q3 = 0.04393;
q5 = 0.016;
q6 = 0.04718;
q7 = 0;
q8 = 0.17357;
q9 = -0.166;
q10 = -0.058;
q11 = -0.018;
q13 = -0.058;
V4 = 1.0187;
V14 = 1.037;
delta_z = zeros(32,1);

% Select Targeted Buses
if target == 1
select1 = [6];
select2 = [9];
select3 = [13];
delta_z(9) = -2.24;
delta_z(12) = 4.3115;
delta_z(15) = -2.0715;
elseif target == 2
    select1 = [3];
    select2 = [5];
    select3 = [6];
delta_z(7) = 3.2356;
delta_z(8) = -1.52;
delta_z(9) = -1.7156;
end

z = [1.56771; 0.73161; 0.2787; 0.08801; 0.1367; 0.12464; -0.942; -0.076; -0.112; 0; -0.295; -0.09; -0.035; -0.135; -0.20378; 0.03568; -0.09478; 0.03591; 0.07187; 0.02372; 0.04393; 0.016; 0.04718; 0; 0.17357; -0.166; -0.058; -0.018; -0.058; 1.0187; 1.037; ];
z = z + delta_z;
x_k = [zeros(1, 13) ones(1, 14)]';
P_k = cov(e_k_initial*e_k_initial');
P_pre_pri = P_k + Q;
x_est_pri = x_k;

%% Jacobian

% x=[t2, t3,...t(n_b-1),V1, V2,... Vn_b]
for i = 1: (n_b - 1)
    x_0_t(i, 1) = 0;
end
for i = 1: n_b
    x_0_V(i, 1) = 1;
end
X0 = [x_0_t; x_0_V];

X_es = zeros(2*(n_b) - 1, n);
Z_es = zeros(length(z), n);

for RT = 1: n
    t1 = 0;
    for i = 1: (n_b - 1)
        t_n(i) = X0(i);
    end
    for i = 1: n_b
        V(i) = X0(n_b - 1 + i);
    end

V = [V(1: n_b)];
t = [0, t_n];

% Calculating dPflow/dt, dPflow/dV, dQflow/dt and dQflow/dV
row_counter = 1;
for k = [[1; 2], [2; 3], [4; 7], [9; 14], [6; 13], [6; 12]]
    select = [k];
    i = select(1);
    j = select(2);
    for m = 1: n_b
if m==i
    P_flow_t = V(i) * V(j) * (g(i,j) * sin(t(i)-t(j)) - b(i,j) * cos(t(i)-t(j)))
    P_flow_V = -V(j) * (g(i,j) * cos(t(i)-t(j)) + b(i,j) * sin(t(i)-t(j)))
    + 2 * (g(i,j)) * V(i);
    Q_flow_t = -V(i) * V(j) * (g(i,j) * cos(t(i)-t(j)) + b(i,j) * sin(t(i)-t(j)))
    Q_flow_V = -V(j) * (g(i,j) * sin(t(i)-t(j))) - b(i,j) * cos(t(i)-t(j)))
    - 2 * (b(i,j)) * V(i);
else
    if m==j
        P_flow_t = -V(i) * V(j) * (g(i,j) * sin(t(i)-t(j)) - b(i,j) * cos(t(i)-t(j)))
        P_flow_V = -V(i) * (g(i,j) * cos(t(i)-t(j)) + b(i,j) * sin(t(i)-t(j)))
        Q_flow_t = V(i) * V(j) * (g(i,j) * cos(t(i)-t(j)) + b(i,j) * sin(t(i)-t(j)))
        Q_flow_V = -V(i) * (g(i,j) * sin(t(i)-t(j))) - b(i,j) * cos(t(i)-t(j)))
    else
        P_flow_t = 0;
        P_flow_V = 0;
        Q_flow_t = 0;
        Q_flow_V = 0;
    end
end
H1_Pflow_t(row_counter,m) = P_flow_t;
H1_Pflow_V(row_counter,m) = P_flow_V;
H1_Qflow_t(row_counter,m) = Q_flow_t;
H1_Qflow_V(row_counter,m) = Q_flow_V;
end
row_counter = row_counter + 1;
end

% Calculating dPinj/dt, dPinj/dV, dQinj/dt, and dQinj/dV
m=1;
for i = [3,5,6,7,8,9,10,11,13]
    for j=1:n_b
        if i ~= j
            P_inj_t(m,j)=V(i)*V(j)*(G(i,j)*sin(t(i)-t(j))-B(i,j)*cos(t(i)-t(j)));
            P_inj_V(m,j)=V(i)*(G(i,j)*cos(t(i)-t(j))+B(i,j)*sin(t(i)-t(j)))
        Q_inj_t(m,j)=V(i)*V(j)*(-G(i,j)*cos(t(i)-t(j))-B(i,j)*sin(t(i)-t(j)));
        Q_inj_V(m,j)=V(i)*(G(i,j)*sin(t(i)-t(j))-B(i,j)*cos(t(i)-t(j)))
        else %i==j
            for k=1:n_b
                Value_Pinj_t(k,1) =V(i)*V(k)*(-G(i,k)*sin(t(i)-t(k))+B(i,k)*cos(t(i)-t(k)));
                Value_Pinj_V(k,1) =V(k)*(G(i,k)*cos(t(i)-t(k))+B(i,k)*sin(t(i)-t(k)))
                Value_Qinj_t(k,1) =V(i)*V(k)*(G(i,k)*cos(t(i)-t(k))+B(i,k)*sin(t(i)-t(k)));
                Value_Qinj_V(k,1) =V(k)*(G(i,k)*sin(t(i)-t(k))-B(i,k)*cos(t(i)-t(k)))
            end
            P_inj_t(m,j)=sum((Value_Pinj_t))-V(i)^2*B(i,i);
            P_inj_V(m,j)=sum((Value_Pinj_V))+V(i)*G(i,i);
            Q_inj_t(m,j)=sum((Value_Qinj_t))-V(i)^2*G(i,i);
            Q_inj_V(m,j)=sum((Value_Qinj_V))-V(i)*B(i,i);
        end
    end
end
m=m+1;
% Calculating dV/dt and dV/dV

H_V_4 = zeros(1,16), 1, zeros(1,10);  
H_V_14 = zeros(1,26), 1;

H = [H1_Pflow_t(:,2:n_b), H1_Pflow_V;  
P_inj_t(:,2:n_b), P_inj_V;  
H1_Qflow_t(:,2:n_b), H1_Qflow_V;  
Q_inj_t(:,2:n_b), Q_inj_V;  
H_V_4;  
H_V_14];

%% Measurement Covariance

R = eye(size(H,2),size(H,2));

for i = 1:6  
R(i,i) = 0.008^2;
end

for i = 7:15  
R(i,i) = 0.01^2;
end

for i = 16:21  
R(i,i) = 0.008^2;
end

for i = 22:30  
R(i,i) = 0.01^2;
end

R(31,31) = 0.004^2;  
R(32,32) = 0.004^2;

%% Gain Matrix
Kalman_Gain = P_pre_pri * H' * inv(H * P_pre_pri * H' + R);

x_est = x_est_pri + Kalman_Gain *(z - H * x_est_pri);

P_k = (eye() - Kalman_Gain * H) * P_pre_pri;

P_pre_pri = P_k + Q;

x_pre_pri = x_est;

Estimation_results(:, RT) = x_est;

end

disp('Iteration and Solution Data:')
disp('| Iteration | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 | t13 | t14 |')
for j = 1:RT

disp('_____________________________________________________________________________________');
fprintf('| %3g | ', j);
fprintf(' %6.2 f | ', 0);
fprintf(' %6.4 f | ', Estimation_results(1, j));
fprintf(' %6.4 f | ', Estimation_results(2, j));
fprintf(' %6.4 f | ', Estimation_results(3, j));
fprintf(' %6.4 f | ', Estimation_results(4, j));
fprintf(' %6.4 f | ', Estimation_results(5, j));
fprintf(' %6.4 f | ', Estimation_results(6, j));
fprintf(' %6.4 f | ', Estimation_results(7, j));
fprintf(' %6.4 f | ', Estimation_results(8, j));
fprintf(' %6.4 f | ', Estimation_results(9, j));
fprintf(' %6.4 f | ', Estimation_results(10, j));
fprintf(' %6.4 f | ', Estimation_results(11, j));
fprintf(' %6.4 f | ', Estimation_results(12, j));
fprintf(' %6.4 f | ', Estimation_results(13, j));

');
fprintf('
');
end
fprintf('
');
fprintf('
');
disp('Iteration and Solution Data:
');
disp('| Iteration | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 | V14 |');
for j = 1:n
    disp('----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
');
    fprintf('| %3g |', j);
    fprintf(' %6.4 f |', Estimation_results(14,j));
    fprintf(' %6.4 f |', Estimation_results(15,j));
    fprintf(' %6.4 f |', Estimation_results(16,j));
    fprintf(' %6.4 f |', Estimation_results(17,j));
    fprintf(' %6.4 f |', Estimation_results(18,j));
    fprintf(' %6.4 f |', Estimation_results(19,j));
    fprintf(' %6.4 f |', Estimation_results(20,j));
    fprintf(' %6.4 f |', Estimation_results(21,j));
    fprintf(' %6.4 f |', Estimation_results(22,j));
    fprintf(' %6.4 f |', Estimation_results(23,j));
    fprintf(' %6.4 f |', Estimation_results(24,j));
    fprintf(' %6.4 f |', Estimation_results(25,j));
    fprintf(' %6.4 f |', Estimation_results(26,j));
    fprintf(' %6.4 f |', Estimation_results(27,j));
    fprintf('
');
end
z_reconstructed_bad_data = H*Estimation_results(:,n);
norm2_bad_data = norm(z-z_reconstructed_bad_data)
norm2_good_data
Listing A.1. Matlab code Kalman Filter

\begin{verbatim}
error_bad = max(z - z_reconstructed_bad_data) / max(z) * 100
error_good = max(z_actual - z_reconstructed_good_data) / max(z_actual) * 100
bad_data_over_good_data = norm2_bad_data / norm2_good_data
\end{verbatim}
clear all
clc

%% Y-bus

mpc = case14;
Measurements = runpf(mpc)

% Branch information
Data_1 = mpc.branch;

n_b = length(mpc.bus);
n_l = length(mpc.branch);

Ybus = zeros(n_b, n_b);

for i = 1:n_l
    Ybus(Data_1(i,1), Data_1(i,2)) = -1/(Data_1(i,3) + j*Data_1(i,4));
    Ybus(Data_1(i,2), Data_1(i,1)) = Ybus(Data_1(i,1), Data_1(i,2));
end

for i = 1:n_b
    Ybus(i, i) = -sum(Ybus(i,:));
end

Ybus;

G = real(Ybus);
B = imag(Ybus);
g = zeros(n_b, n_b);

% Non-diagonal G terms
for j = 1:n_b
    for i = 1:n_b
        if i == j
            else
                g(i,j) = abs(G(i,j));
            end
        end
    end
    g = triu(g);
% Non-diagonal B terms
for j = 1:n_b
    for i = 1:n_b
        if i == j
            else
                b(i,j) = -abs((B(i,j)));
        end
    end
end
b = triu(b);

%% Measurements
p12 = 1.56771;
p23 = 0.73161;
p47 = 0.2787;
p914 = 0.08801;
p613 = 0.1367;
p612 = 0.12464;
p3 = -0.942;
p5 = -0.076;
p6 = -0.112;
p7 = 0;
p8 = 0;
p9 = -0.295;
p10 = -0.09;
p11 = -0.035;
p13 = -0.135;
p12 = -0.20378;
p23 = 0.03568;
p47 = -0.09478;
p914 = 0.03591;
p613 = 0.07187;
p612 = 0.02372;
p3 = 0.04393;
p5 = 0.016;
p6 = 0.04718;
p7 = 0;
p8 = 0.17357;
p9 = -0.166;
p10 = -0.058;
p11 = -0.018;
q13 = -0.058;
V4 = 1.0187;
V14 = 1.037;

z = [1.56771; 0.73161; 0.2787; 0.08801; 0.1367; 0.12464; -0.942; -0.076; -0.112; 0;
0; -0.295; -0.09; -0.035; -0.135; -0.20378; 0.03568; -0.09478; 0.03591; 0.07187;
0.02372; 0.04393; 0.016; 0.04718; 0; 0.03568; -0.09478; 0.03591; 0.07187;
1.037;];

%% Jacobian

n = 5; % iteration number

x = [t2, t3, ... t(n_b-1), V1, V2, ... Vn_b]

for i = 1: (n_b - 1)
    x_0_t(i, 1) = 0;
end

for i = 1: n_b
    x_0_V(i, 1) = 1;
end

X0 = [x_0_t; x_0_V];

X_es = zeros(2*(n_b) - 1, n);
Z_es = zeros(length(z), n);

for RT = 1: n
    t1 = 0;
    for i = 1: (n_b - 1)
        t_n(i) = X0(i);
    end

    for i = 1: n_b
        V(i) = X0(n_b - 1 + i);
    end

V = [V(1:n_b)];
t = [0, t_n];

% Calculating dPflow/dt, dPflow/dV, dQflow/dt and dQflow/dV
row_counter = 1;
for k = [1; 2], [2; 3], [4; 7], [9; 14], [6; 13], [6; 12]
select = [k];
i = select (1);
j = select (2);
for m = 1:n_b
    if m == i
        P_flow_t = V(i) * V(j) * (g(i, j) * sin(t(i) - t(j)) - b(i, j) * cos(t(i) - t(j)));
        P_flow_V = -V(j) * (g(i, j) * cos(t(i) - t(j)) + b(i, j) * sin(t(i) - t(j))) + 2*(g(i, j)) * V(i);
        Q_flow_t = -V(i) * V(j) * (g(i, j) * cos(t(i) - t(j)) + b(i, j) * sin(t(i) - t(j)));
        Q_flow_V = -V(j) * (g(i, j) * sin(t(i) - t(j)) - b(i, j) * cos(t(i) - t(j))) - 2*(b(i, j)) * V(i);
    else
        if m == j
            P_flow_t = -V(i) * V(j) * (g(i, j) * sin(t(i) - t(j)) - b(i, j) * cos(t(i) - t(j)));
            P_flow_V = -V(i) * (g(i, j) * cos(t(i) - t(j)) + b(i, j) * sin(t(i) - t(j)))
            Q_flow_t = V(i) * V(j) * (g(i, j) * cos(t(i) - t(j)) + b(i, j) * sin(t(i) - t(j)));
            Q_flow_V = -V(i) * (g(i, j) * sin(t(i) - t(j)) - b(i, j) * cos(t(i) - t(j)))
        else
            P_flow_t = 0;
            P_flow_V = 0;
            Q_flow_t = 0;
            Q_flow_V = 0;
        end
    end
end
H1_Pflow_t(row_counter,m) = P_flow_t;
H1_Pflow_V(row_counter,m) = P_flow_V;
H1_Qflow_t(row_counter,m) = Q_flow_t;
H1_Qflow_V(row_counter,m) = Q_flow_V;
% Calculating dPinj/dt, dPinj/dV, dQinj/dt, and dQinj/dV

m = 1;
for i = [3, 5, 6, 7, 8, 9, 10, 11, 13]
for j = 1: n_b
    if i == j
        \[
P_{\text{inj}_t}(m,j) = V(i) V(j) (G(i,j) \sin(t(i) - t(j)) - B(i,j) \cos(t(i) - t(j)))
        \]
        \[
P_{\text{inj}_V}(m,j) = V(i) (G(i,j) \cos(t(i) - t(j)) + B(i,j) \sin(t(i) - t(j)))
        \]
        \[
Q_{\text{inj}_t}(m,j) = V(i) V(j) (-G(i,j) \cos(t(i) - t(j)) - B(i,j) \sin(t(i) - t(j)))
        \]
        \[
Q_{\text{inj}_V}(m,j) = V(i) (G(i,j) \sin(t(i) - t(j)) - B(i,j) \cos(t(i) - t(j)))
        \]
    end

    else if i != j
        for k = 1: n_b
            Value_PINJ_t(k,1) = V(i) V(k) (-G(i,k) sin(t(i) - t(k)) + B(i,k) * cos(t(i) - t(k)));
            Value_PINJ_V(k,1) = V(k) (G(i,k) * cos(t(i) - t(k)) + B(i,k) * sin(t(i) - t(k)));
        end
        \[
P_{\text{inj}_t}(m,j) = \text{sum}((\text{Value_PINJ}_t)) - V(i)^2 B(i,i);
        \]
        \[
P_{\text{inj}_V}(m,j) = \text{sum}((\text{Value_PINJ}_V)) + V(i) G(i,i);
        \]
        \[
Q_{\text{inj}_t}(m,j) = \text{sum}((\text{Value_QINJ}_t)) - V(i)^2 G(i,i);
        \]
        \[
Q_{\text{inj}_V}(m,j) = \text{sum}((\text{Value_QINJ}_V)) + V(i) B(i,i);
        \]
    end
Q_{inj}_V(m,j) = \sum (\text{Value}_{Qinj}_V) - V(i) * B(i,i);

% Calculating dV/dt and dV/dV

H_{V_4} = [\text{zeros}(1,16), 1, \text{zeros}(1,10)];
H_{V_{14}} = [\text{zeros}(1,26), 1];

H = [H1_{Pflow_t}(::2:n_b), H1_{Pflow_V};
     P_{inj_t}(::2:n_b), P_{inj_V};
     H1_{Qflow_t}(::2:n_b), H1_{Qflow_V};
     Q_{inj_t}(::2:n_b), Q_{inj_V};
     H_{V_4};
     H_{V_{14}} ];

%% Measurement Covariance

R = \text{eye(size}(H,2), \text{size}(H,2));

for i = 1:6
    R(i,i) = 0.008^2;
end

for i = 7:15
    R(i,i) = 0.01^2;
end

for i = 16:21
    R(i,i) = 0.008^2;
end
for i = 22:30
    R(i,i) = 0.01^2;
end
R(31,31) = 0.004^2;
R(32,32) = 0.004^2;

%% Gain Matrix
GG = H' * inv(R) * H;

% check the eigenvalues to ensure it is positive definite (positive eigenvalues)
eig(GG);

% Cholesky Decomposition
L = chol(GG)';

% % initial conditions
t(1) = 0;
for i = 2:n_b
t(i) = X0(i-1);
end
for i = n_b
V(i) = X0(i+(n_b-1));
end

%% building the estimated measurements (h(x))

% pij and qij estimated measurements
m = 1;
for k = [1; 2], [2; 3], [4; 7], [9; 14], [6; 13], [6; 12]
    select = [k];
    i = select(1);
    j = select(2);
    p_ij(m,1) = V(i)^2 * g(i,j) - V(i) * V(j) * (g(i,j) * cos(t(i)-t(j)) + b(i,j))
\begin{verbatim}
\*sin(t(i)-t(j));
q_ij(m,1)=-V(i)^2*b(i,j)-V(i)*V(j)*(g(i,j)*\*sin(t(i)-t(j))-b(i,j)\*cos(t(i)-t(j)));
m=m+1;
end

\%pi and qi estimated measurements
m=1;
for i =\[3,5,6,7,8,9,10,11,13]\n  for j=1:n_b
    p_k(j,1)=V(i)*V(j)*(G(i,j)*\*cos(t(i)-t(j))+B(i,j)*\*sin(t(i)-t(j)));
    q_k(j,1)=V(i)*V(j)*(G(i,j)*\*sin(t(i)-t(j))-B(i,j)*\*cos(t(i)-t(j)));
  end
end
p(m,1)=sum((p_k));
q(m,1)=sum((q_k));
m=m+1;
end

%Assigning V (1 to 14) to v(1 to 14)
m=1;
for i =\[1,14]\n  v(m)=V(i);
m=m+1;
end

% vector of estimated measurements
h_x=[p_ij; p; q_ij; q; v'];

% right side of state-estimation
t_k=H'*inv(R)*(z-h_x);
\end{verbatim}
% forward substitution
u1 = inv(L) * t_k;

% backward substitution
dx = inv(L') * u1;

% calculation of new estimated values
x = X0 + dx;
X0 = x;

%%% Saving Results

t(1) = 0;
for i = 2:n_b
t(i) = X0(i - 1);
end
for i = 1:n_b
V(i) = X0(i + (n_b - 1));
end

%p i j and q i j estimated measurements
m = 1;
for k = [[1; 2], [2; 3], [4; 7], [9; 14], [6; 13], [6; 12]]
select = [k];
i = select(1);
j = select(2);

    p_ij(m, i) = V(i)^2 * g(i, j) - V(i) * V(j) * (g(i, j) * cos(t(i) - t(j)) + b(i, j) * sin(t(i) - t(j)));
    q_ij(m, i) = V(i)^2 * b(i, j) - V(i) * V(j) * (g(i, j) * sin(t(i) - t(j)) - b(i, j) * cos(t(i) - t(j)));
    m = m + 1;
end

%p i and q i estimated measurements
m = 1;
for i = [3, 5, 6, 7, 8, 9, 10, 11, 13]
for j = i:n_b
\[ p_{k}(j,1) = V(i) * V(j) * (G(i,j) * \cos(t(i)-t(j)) + B(i,j) * \sin(t(i)-t(j)) \); \\
q_{k}(j,1) = V(i) * V(j) * (G(i,j) * \sin(t(i)-t(j)) - B(i,j) * \cos(t(i)-t(j)) \); \\
end \\
p(m,1) = \text{sum}((p_{k})); \\
q(m,1) = \text{sum}((q_{k})); \\
m = m + 1; \\
end \\

% Assigning V (1 to 14) to v(1 to 14) \\
m = 1; \\
for i = [1, 14] \\
v(m) = V(i); \\
m = m + 1; \\
end \\
h_{es} = [p_{ij}; p; q_{ij}; q; v']; \\
X_{es}(:, RT) = X0; \\
Z_{es}(:, RT) = h_{es}; \\
end \\

%% Printing Results \\
disp('Iteration and Solution Data:') \\
disp('| Iteration | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 | t10 | t11 | t12 | t13 | t14 | ') \\
for j = 1:RT
```matlab
392 disp('-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
393 fprintf(' | %3g |', j);
394 fprintf(' | %3g |', j);
395 fprintf(' %6.2 f |', 0);
396 fprintf(' %6.4 f |', X_es(1,j));
397 fprintf(' %6.4 f |', X_es(2,j));
398 fprintf(' %6.4 f |', X_es(3,j));
399 fprintf(' %6.4 f |', X_es(4,j));
400 fprintf(' %6.4 f |', X_es(5,j));
401 fprintf(' %6.4 f |', X_es(6,j));
402 fprintf(' %6.4 f |', X_es(7,j));
403 fprintf(' %6.4 f |', X_es(8,j));
404 fprintf(' %6.4 f |', X_es(9,j));
405 fprintf(' %6.4 f |', X_es(10,j));
406 fprintf(' %6.4 f |', X_es(11,j));
407 fprintf(' %6.4 f |', X_es(12,j));
408 fprintf(' %6.4 f |', X_es(13,j));
409 fprintf(' %6.4 f |', X_es(14,j));
410 fprintf(' %6.4 f |', X_es(15,j));
411 fprintf(' %6.4 f |', X_es(16,j));
412 fprintf(' %6.4 f |', X_es(17,j));
413 fprintf(' %6.4 f |', X_es(18,j));
414 fprintf(' %6.4 f |', X_es(19,j));
415 end

416 fprintf('
');
417 fprintf('
');
418 disp('Iteration and Solution Data:');
419 disp('| Iteration | V1 | V2 | V3 |
420 | V4 | V5 | V6 | V7 |
421 | V8 | V9 | V10 | V11 |
422 | V12 | V13 |
423 | V14 | '); for j = 1:RT
424fprintf(' | %3g |', j);
425fprintf(' | %3g |', j);
426fprintf(' %6.2 f |', 0);
427fprintf(' %6.4 f |', X_es(14,j));
428fprintf(' %6.4 f |', X_es(15,j));
429fprintf(' %6.4 f |', X_es(16,j));
430fprintf(' %6.4 f |', X_es(17,j));
431fprintf(' %6.4 f |', X_es(18,j));
432fprintf(' %6.4 f |', X_es(19,j));
```

69
fprintf('%6.4f |', X_es(20,j));
fprintf('%6.4f |', X_es(21,j));
fprintf('%6.4f |', X_es(22,j));
fprintf('%6.4f |', X_es(23,j));
fprintf('%6.4f |', X_es(24,j));
fprintf('%6.4f |', X_es(25,j));
fprintf('%6.4f |', X_es(26,j));
fprintf('%6.4f |', X_es(27,j));
fprintf('
');
end

fprintf('
');
fprintf('
');
disp('Iteration and Solution Data:)
for j = 1:RT
    disp('------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------');
    fprintf('| %3g |', j);
    fprintf('%6.4f |', Z_es(1,j));
    fprintf('%6.4f |', Z_es(2,j));
    fprintf('%6.4f |', Z_es(3,j));
    fprintf('%6.4f |', Z_es(4,j));
    fprintf('%6.4f |', Z_es(5,j));
    fprintf('%6.4f |', Z_es(6,j));
    fprintf('%6.4f |', Z_es(7,j));
    fprintf('%6.4f |', Z_es(8,j));
    fprintf('%6.4f |', Z_es(9,j));
    fprintf('%6.4f |', Z_es(10,j));
    fprintf('%6.4f |', Z_es(11,j));
    fprintf('%6.4f |', Z_es(12,j));
    fprintf('%6.4f |', Z_es(13,j));
    fprintf('%6.4f |', Z_es(14,j));
    fprintf('%6.4f |', Z_es(15,j));
    fprintf('
');
    fprintf('%3g |', j);
    fprintf('%6.4f |', Z_es(1,j));
    fprintf('%6.4f |', Z_es(2,j));
    fprintf('%6.4f |', Z_es(3,j));
    fprintf('%6.4f |', Z_es(4,j));
    fprintf('%6.4f |', Z_es(5,j));
    fprintf('%6.4f |', Z_es(6,j));
    fprintf('%6.4f |', Z_es(7,j));
    fprintf('%6.4f |', Z_es(8,j));
    fprintf('%6.4f |', Z_es(9,j));
    fprintf('%6.4f |', Z_es(10,j));
    fprintf('%6.4f |', Z_es(11,j));
    fprintf('%6.4f |', Z_es(12,j));
    fprintf('%6.4f |', Z_es(13,j));
    fprintf('%6.4f |', Z_es(14,j));
    fprintf('%6.4f |', Z_es(15,j));
    fprintf('
');
```
end

fprintf ('\n');
fprintf ('\n');
disp ('Iteration and Solution Data:\n');
disp ('| Iteration | q12 | q23 | q47 | q914 | q613 | q612 | q3 | q5 | q6 | q7 | q8 | q9 | q10 | q11 | q13 |');
for j = 1:RT

disp ('------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------');
fprintf ('| %3g |', j);
fprintf (' %6.4 f |', Z_es (16 ,j));
fprintf (' %6.4 f |', Z_es (17 ,j));
if j ==1
fprintf (' %6.4 f |', Z_es (19 ,j));
else
fprintf (' %6.4 f |', Z_es (19 ,j));
end
fprintf (' %6.4 f |', Z_es (20 ,j));
fprintf (' %6.4 f |', Z_es (21 ,j));
fprintf (' %6.4 f |', Z_es (22 ,j));
fprintf (' %6.4 f |', Z_es (23 ,j));
fprintf (' %6.4 f |', Z_es (24 ,j));
fprintf (' %6.4 f |', Z_es (25 ,j));
fprintf (' %6.4 f |', Z_es (26 ,j));
fprintf (' %6.4 f |', Z_es (27 ,j));
fprintf (' %6.4 f |', Z_es (28 ,j));
fprintf (' %6.4 f |', Z_es (29 ,j));
fprintf (' %6.4 f |', Z_es (30 ,j));
fprintf ('\n');
end

Listing A.2. Matlab code Weighted Least Squares
```
Appendix B
Power Flow Algorithms

B.1 Gauss-Seidel Method in depth

The first step when using the Gauss-Seidel method for power flow is to find the $Y_{bus}$ matrix, which consists of the following terms:

$$Y_{bus} = \begin{cases} Y_{ii} & \text{Sum of all admittances connected to bus i} \\ Y_{ij} & \text{-[Sum of all admittances connected to bus i and j]} \end{cases}$$

It is important to then calculate the active and reactive power at each bus $k$, considering the generated power, $P_{Gk}$ and $Q_{Gk}$, and the load power, $P_{Lk}$ and $Q_{Lk}$ as shown in Equations B.1 and B.2.

$$P_k = P_{Gk} - P_{Lk} \quad \text{(B.1)}$$

$$Q_k = Q_{Gk} - Q_{Lk} \quad \text{(B.2)}$$

Once we have calculated all the basic variables of the system, we can start establishing the iterative Gauss-Seidel Method. To initialize this algorithm, we first have to have an initial guess by assuming that all voltages are 1’s and that all angles are 0’s. For PV buses, the active power is known so we initialize it for all PV buses. For PQ buses, the active and reactive power are known so we initialize those for all PQ buses.

Considering the mentioned variables, we have all the necessary information to solve for the power flow of a given system. The slack bus will always have a voltage magnitude of 1 and an angle of 0 throughout this algorithm. The PV and PQ buses will follow a similar process when calculating the voltage for the next iteration following Equation
\[ V_k(i + 1) = \frac{1}{Y_{kk}} \left[ \left( \frac{P_k - jQ_k}{V_k^*(i)} \right) - \sum_{n=1, n \neq k}^{N} Y_{kn}V_n(i) \right] \] (B.3)

For the PQ buses, the new voltage will be updated using Equation B.3, for both the magnitude and the angle.

For the PV bus, only the voltage angle will be extracted from Equation B.3 since the voltage is already known. And the new voltage will be updated to Equation B.4. The reactive power for the next step will also be calculated using Equation B.5 and the reactive power will be limited to be between the maximum and minimum specified by the system as shown in Equation B.6.

\[ V_{\text{new}}(i) = 1 \ast (\cos(\theta_i) + j\sin(\theta_i)) \] (B.4)

\[ Q_{\text{new}}(i) = \text{imag}(V(i) \ast (\sum_{n=1, n \neq k}^{N} Y_{kn}V_n(i))^*) \] (B.5)

\[ Q_k^{\text{min}} \leq Q_k(i + 1) \leq Q_k^{\text{max}} \] (B.6)

Once we have performed these steps, the first iteration for the Gauss-Seidel method has been completed. We will then go back to Equation B.3 and continue with the iterative process until the difference between the previous step and the current step is \( \leq \epsilon \), where \( \epsilon \) is a very small decimal number. The Gauss-Seidel method algorithm is summarized below in Figure 2.1.
Appendix C
State estimation algorithms

C.1 Weighted Least Squares in-depth

The measurement equation for a power system state estimation is given as

\[ z = h(x) + e \] (C.1)

Where \( x \) is the state vector, \( z \) is the measurement vector, and \( h(x) \) is the measurement function.

The solution can be formulated as a minimization of the following objective function.

\[ J(x) = \sum_{i=1}^{m} \frac{(z_i - h_i(x))^2}{R_{ii}} \] (C.2)

The WLS objective function can be written as follows.

\[ J(x) = (z - h(x))^T R^{-1} [z - h(x)] \] (C.3)

Where \( R \) is the measurement error covariance matrix. When using the WLS method, it is assumed that the measurement errors, \( e \), are Gaussian and independent [43].

Equations C.4 and C.5 are the two main equations utilized in this iterative method.

\[ \begin{bmatrix} G(x^k) \end{bmatrix} \Delta x^{k+1} = \begin{bmatrix} J(x^k) \end{bmatrix}^T R^{-1} (z - h(x^k)) \] (C.4)

\[ \Delta x^{k+1} = x^{k+1} - x^k \] (C.5)

The Normal Equation, Equation C.5, is used to solve for the change until \( \max |\Delta x_i^{k+1}| \leq \epsilon \) in the state vector, where \( \epsilon \) is the accuracy of convergence [44].
\[ g(x^k) = -\left[ J(x^k) \right]^T R^{-1}(z - h(x^k)) \] (C.6)

The Gain matrix, \( G(x^k) \), is given by Equation C.7, and is calculated from the partial of matrix \( g(x^k) \) with respect with \( x \), or by using the Jacobian matrix \( J(x) \) and the measurement error covariance \( R \).

\[
G(x^k) = \frac{\partial g(x^k)}{\partial x} = \left[ J(x^k) \right]^T R^{-1} \left[ J(x^k) \right] \tag{C.7}
\]

The Jacobian matrix, \( J(x) \), is calculated in Equation C.8 and is calculated by taking the partial derivatives with respect with \( x \) of the measurement function \( h(x) \).

\[
J(x) = \left[ \frac{\partial h(x)}{\partial x} \right] \tag{C.8}
\]

The gain matrix \( G \) is normally not invertible. It can be written as a product of a lower triangular sparse matrix and its transpose by using the Cholesky decomposition in Equations C.9 - C.12

\[
G = LL^T \tag{C.9}
\]

\[
\left[ G(x^k) \right] \Delta x^{k+1} = \left[ J(x^k) \right]^T R^{-1}(z - h(x^k)) \tag{C.10}
\]

\[
LL^T \Delta x^{k+1} = \left[ J(x^k) \right]^T R^{-1}(z - h(x^k)) \tag{C.11}
\]

\[
LL^T \Delta x^{k+1} = t^k \tag{C.12}
\]

By using forward/backward substitution, we can solve for the above problem by letting \( L^T \Delta x^{k+1} = u \). Using this substitution, simplifies Equation C.12 to \( Lu = t^k \). We can now apply the forward substitution by solving for \( u = inv(L) t^k \), and now that we have solved for \( u \), we can solve for \( \Delta x^{k+1} \) using backward substitution in Equation C.13.

\[
\Delta x^{k+1} = inv(L^T)u \tag{C.13}
\]

Due to the sparse nature of the triangular matrix \( L \), the state-estimation process is significantly efficient. Next, we need to find the estimated parameters using Equation C.14. The Weighted Least-Squares Estimation Algorithm for power systems can be found below step by step.

\[
x^{k+1} = x^k + \Delta x^{k+1} \tag{C.14}
\]
Appendix D
Reconstructed measurements raw data

D.1 Results Reconstructed Good Data

Table D.1. Reconstructed Line Flow Measurements

<table>
<thead>
<tr>
<th>Bus No. (From)</th>
<th>Bus No. (To)</th>
<th>Active Power (P)</th>
<th>Reactive Power (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.5683</td>
<td>-0.2039</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.7337</td>
<td>0.0382</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.2781</td>
<td>-0.0759</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>0.0739</td>
<td>0.0097</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>0.1614</td>
<td>0.0496</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>0.1175</td>
<td>0.0136</td>
</tr>
</tbody>
</table>

Table D.2. Reconstructed Bus Injection measurements

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-0.9405</td>
<td>0.0476</td>
<td>5</td>
<td>-0.0762</td>
<td>0.0183</td>
</tr>
<tr>
<td>6</td>
<td>-0.1197</td>
<td>0.0645</td>
<td>7</td>
<td>0.0013</td>
<td>0.0286</td>
</tr>
<tr>
<td>8</td>
<td>0.0010</td>
<td>0.2035</td>
<td>9</td>
<td>-0.2918</td>
<td>-0.1376</td>
</tr>
<tr>
<td>10</td>
<td>-0.0883</td>
<td>-0.0310</td>
<td>11</td>
<td>-0.0375</td>
<td>0.0034</td>
</tr>
<tr>
<td>13</td>
<td>-0.1194</td>
<td>-0.0692</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table D.3. Reconstructed Voltages

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Bus No.</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.0359</td>
<td>14</td>
<td>1.0226</td>
</tr>
</tbody>
</table>

### D.2 Results Reconstructed Bad Data 1

### Table D.4. Results

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0780</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>1.0597</td>
<td>-0.0966</td>
</tr>
<tr>
<td>3</td>
<td>1.0222</td>
<td>-0.2310</td>
</tr>
<tr>
<td>4</td>
<td>1.0299</td>
<td>-0.1767</td>
</tr>
<tr>
<td>5</td>
<td>1.0101</td>
<td>-0.1957</td>
</tr>
<tr>
<td>6</td>
<td>0.8873</td>
<td>-0.6865</td>
</tr>
<tr>
<td>7</td>
<td>1.0594</td>
<td>-0.1323</td>
</tr>
<tr>
<td>8</td>
<td>1.0885</td>
<td>-0.2220</td>
</tr>
<tr>
<td>9</td>
<td>1.0580</td>
<td>0.0030</td>
</tr>
<tr>
<td>10</td>
<td>1.0141</td>
<td>-0.1559</td>
</tr>
<tr>
<td>11</td>
<td>0.9408</td>
<td>-0.4413</td>
</tr>
<tr>
<td>12</td>
<td>0.8629</td>
<td>-0.7328</td>
</tr>
<tr>
<td>13</td>
<td>0.8377</td>
<td>-0.7197</td>
</tr>
<tr>
<td>14</td>
<td>1.0280</td>
<td>-0.0731</td>
</tr>
</tbody>
</table>
### Table D.5. Reconstructed Line Flow Measurements

<table>
<thead>
<tr>
<th>Bus No. (From)</th>
<th>Bus No. (To)</th>
<th>Active Power (P)</th>
<th>Reactive Power (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.5649</td>
<td>-0.2045</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.6857</td>
<td>0.0270</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>-0.2121</td>
<td>-0.1412</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>0.2733</td>
<td>-0.0175</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>0.3562</td>
<td>0.2001</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>0.1842</td>
<td>0.0071</td>
</tr>
</tbody>
</table>

### Table D.6. Reconstructed Bus Injection measurements

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-0.9767</td>
<td>0.0417</td>
<td>5</td>
<td>-0.0952</td>
<td>0.0139</td>
</tr>
<tr>
<td>6</td>
<td>-2.5157</td>
<td>-0.0195</td>
<td>7</td>
<td>-0.5086</td>
<td>-0.0107</td>
</tr>
<tr>
<td>8</td>
<td>0.0010</td>
<td>0.1648</td>
<td>9</td>
<td>3.6450</td>
<td>-0.1447</td>
</tr>
<tr>
<td>10</td>
<td>-0.4245</td>
<td>-0.0494</td>
<td>11</td>
<td>-0.2859</td>
<td>-0.0462</td>
</tr>
<tr>
<td>13</td>
<td>-2.1026</td>
<td>0.0053</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table D.7. Reconstructed Voltages

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Bus No.</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.0299</td>
<td>14</td>
<td>1.0280</td>
</tr>
</tbody>
</table>
### D.3 Results Reconstructed Bad Data 2

**Table D.8. Results**

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.3137</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>1.2954</td>
<td>-0.0969</td>
</tr>
<tr>
<td>3</td>
<td>1.2546</td>
<td>-0.2454</td>
</tr>
<tr>
<td>4</td>
<td>1.0371</td>
<td>-0.7655</td>
</tr>
<tr>
<td>5</td>
<td>1.0882</td>
<td>-0.6453</td>
</tr>
<tr>
<td>6</td>
<td>1.0445</td>
<td>-1.0569</td>
</tr>
<tr>
<td>7</td>
<td>1.0569</td>
<td>-0.8402</td>
</tr>
<tr>
<td>8</td>
<td>1.0895</td>
<td>-0.8297</td>
</tr>
<tr>
<td>9</td>
<td>1.0461</td>
<td>-0.8297</td>
</tr>
<tr>
<td>10</td>
<td>1.0428</td>
<td>-0.9233</td>
</tr>
<tr>
<td>11</td>
<td>1.0449</td>
<td>-0.9869</td>
</tr>
<tr>
<td>12</td>
<td>1.0261</td>
<td>-1.0911</td>
</tr>
<tr>
<td>13</td>
<td>1.0315</td>
<td>-1.0521</td>
</tr>
<tr>
<td>14</td>
<td>1.0288</td>
<td>-0.9214</td>
</tr>
</tbody>
</table>

**Table D.9. Reconstructed Line Flow Measurements**

<table>
<thead>
<tr>
<th>Bus No. (From)</th>
<th>Bus No. (To)</th>
<th>Active Power (P)</th>
<th>Reactive Power (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.5709</td>
<td>-0.2056</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.7565</td>
<td>0.0269</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0.3573</td>
<td>-0.0945</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>0.1114</td>
<td>0.0117</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>0.0111</td>
<td>0.0943</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>0.1369</td>
<td>0.0063</td>
</tr>
</tbody>
</table>
**Table D.10.** Reconstructed Bus Injection measurements

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
<th>Bus No.</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2.3113</td>
<td>0.0425</td>
<td>5</td>
<td>-0.15895</td>
<td>0.0168</td>
</tr>
<tr>
<td>6</td>
<td>-1.7722</td>
<td>0.0621</td>
<td>7</td>
<td>0.0605</td>
<td>0.0071</td>
</tr>
<tr>
<td>8</td>
<td>0.0596</td>
<td>0.1854</td>
<td>9</td>
<td>-0.2647</td>
<td>-0.1550</td>
</tr>
<tr>
<td>10</td>
<td>-0.0542</td>
<td>-0.0441</td>
<td>11</td>
<td>0.0113</td>
<td>-0.0059</td>
</tr>
<tr>
<td>13</td>
<td>-0.2092</td>
<td>-0.0244</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table D.11.** Reconstructed Voltages

<table>
<thead>
<tr>
<th>Bus No.</th>
<th>Voltage</th>
<th>Bus No.</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.0371</td>
<td>14</td>
<td>1.0288</td>
</tr>
</tbody>
</table>
Bibliography


URL https://www.i-scoop.eu/industry-4-0/smart-grids-electrical-grid/


