HWI HAMBURG

HOCHSCHULÜBERGREIFENDER STUDIENGANG
WIRTSCHAFTSINGENIEURWESEN HAMBURG

Masterarbeit

Simulation eines IT-Netzwerks zur False-Data-Injection in Smart Grids

vorgelegt von

Moritz Volkmann

Matrikelnummer 6789467

Bereich:

1. Gutachter:	**Prof. Dr.-Ing. Volker Skwarek**
2. Gutachter:	**Bastian Pfarrherr**
vorgelegt am:	**01. Oktober 2024**

AM LEHRANGEBOT BETEILIGTE HOCHSCHULEN:
Universität Hamburg
Hochschule für Angewandte Wissenschaften Hamburg
Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg

Versicherung über die Selbstständigkeit

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit ohne fremde Hilfe selbstständig verfasst habe. Ich habe keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellverzeichnis nicht benannten Internet-Quellen – benutzt. Ich habe die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht. Die schriftliche Fassung entspricht der auf dem elektronischen Speichermedium.

Hamburg, den 01. Oktober 2024	
Ort, Datum	Unterschrift

Contents

List of Figures

List of Tables

Nomenclature

Symbols

Latin Symbols

Symbol	Unit	Meaning
$\mathbf{a}$	Vector	Attack Vector
E	J	Energy Consumption
$\mathbf{e}$	Vector	Vector of Measurement Errors
f	Hz	Frequency
$\mathbf{H}$	Matrix	Jacobian Matrix
I	A	Current
P	W	Active Power
Q	var	Reactive Power
S	W	Apparent Power
$\mathbf{t}$	Vector	Error Vector
U	V	Voltage
$\mathbf{x}$	Vector	State Vector
Y	S	Line Admittance
$\mathbf{z}$	Vector	Measurement Vector
Z	Ω	Line Impedance

Greek Symbols

Symbol	Unit	Meaning
ϵ, τ	–	State Estimation Residual
φ	–	Voltage Angle
χ^2	–	Chi-Squared Distribution / Test

Indices

Index	Meaning
i, j, m, n	Bus

Acronyms

Acronym	Meaning
5G	Fifth Generation (Technology Standard for Cellular Networks)
AC	Alternating Current
AMI	Advanced Metering Infrastructure
ANN	Artificial Neural Network
CR	Communication Requirement
CSV	Comma-Separated Values File Format
DC	Direct Current
DER	Distributed Energy Resource
FDIA	False Data Injection Attack
GA	Genetic Algorithm
GDPR	General Data Protection Regulation
GO	Grid Operator
GUI	Graphical User Interface
HEMS	Home Energy Management System
ICT	Information and Communication Technology
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LTE	Long Term Evolution (Technology Standard for Cellular Networks)
LV	Low-Voltage
MLGA	Machine-Learning and Genetic Algorithm
MLP	Multi-Layer Perceptron
MND	Mean Nodal Deviations
MQTT	Message Queueing Telemetry Transport
NS-3	Network Simulator 3
OSI	Open Systems Interconnection
PMU	Phasor Measurement Unit
PV	Photovoltaic
R	Requirement
ReLU	Rectified Linear Unit
RQ	Research Question
SCADA	Supervisory Control and Data Acquisition
TAC	Tariff Application Case
TCP	Transmission Control Protocol
TLS	Transport Layer Security
SMGW	Smart Meter Gateway
UDP	User Datagram Protocol

1 Introduction

The energy grid has become increasingly decentralized in recent years with the progressive expansion of renewable energy sources and the associated transition of households from energy consumers to self-producers of energy (Leal-Arcas et al., 2019). Traditionally, energy generation occurred through large power plants at the transmission and high-voltage network level, but it is now increasingly occurring at the distribution network level. As a result, energy network consideration has shifted towards distribution networks, which are taking on an increasingly central (Steffel, 2010).

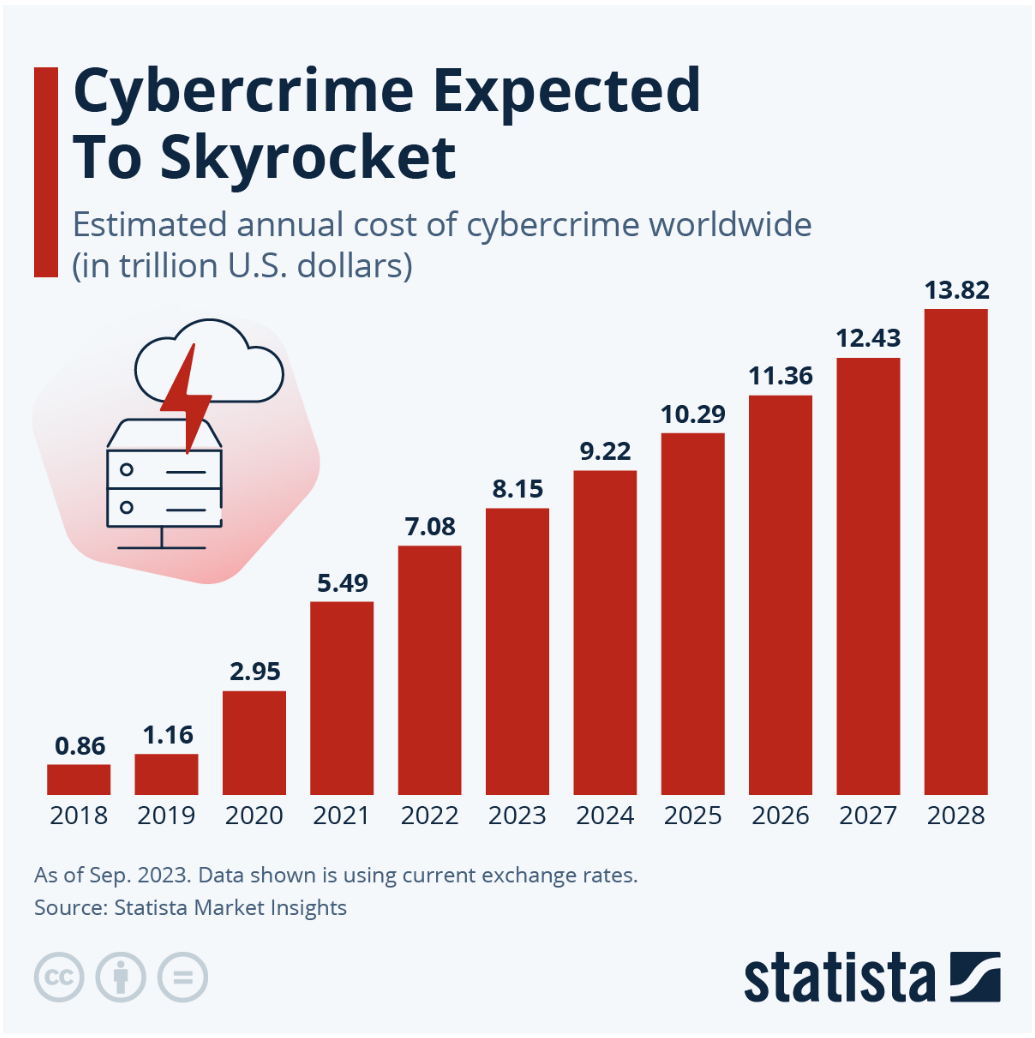

Figure 1.1: Trend in cybercrime
Source:
`https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027/`

Energy supply is part of critical infrastructure and is an essential component of a modern, digital society. Any impairment or failure of this supply could have serious economic and social consequences. The novel "Black Out" by Marc Elsberg vividly describes a total blackout of all European power grids caused by a cyberattack, employing realistic processes and scenarios. Even though it is a fictional scenario, the threat posed by cyberattacks should not be underestimated. Especially since the COVID-19 pandemic and Russia's war on Ukraine, the number of cyberattacks has dramatically increased in recent years and continues to trend upward, as shown in Figure 1.1. Nations such as Russia and North Korea have intensified their cyber warfare not only to achieve political goals but also for financial motives [1]. In recent years, there has been a notable increase in targeted attacks on the energy sector of the USA and EU member states [2]. The increased use of advanced metering infrastructure (AMI) and smart meter gateways (SMGWs) in the power grid and the accompanying intensified networking also open potential attack surfaces for hackers. To ensure the security of the energy supply, the power grid must be safeguarded not only from an electrical and physical perspective but also in terms of all communication pathways against cyberattacks.

The present thesis aims to simulate the communication infrastructure for controlling a microgrid up to the distribution network level. Existing simulation tools will be utilized to the greatest extent possible, and an interaction/co-simulation between the communication network and the energy network will be achieved. Custom developments and adjustments will be made in the simulation of the communication network as required, while the network simulation will be taken over unchanged as a black box.

To approximate a realistic scenario, smart meter gateways and communication infrastructure for measurement data and control communication will be simulated using an IT system that is connected to the network simulation. This co-simulation will be used to assess the risk and impact of cyberattacks, particularly false data injections, using simulated attacks, thereby uncovering attack surfaces and vectors. The focus will be on testing whether and how it is possible to influence the regular operation of a low-voltage (LV) network through false data injections, the specific consequences of such attacks on the network and communication infrastructure, and the measures that can be taken to mitigate these consequences and complicate the attacks.

The findings from this work will be used to secure the energy grid and develop recommendations for identifying and safeguarding against false data injections and cyberattacks. The simulation software developed during this work will also be usable and extendable for future attack simulations and security analyses in the energy grid.

1.1 Research Questions

This thesis aims to answer the following three research questions:

RQ 1: How can a power grid and its communication infrastructure be simulated so that false data injection attacks and their impact on the grid can be tested and evaluated?

To answer this question, we will first look into the components and communication infrastructure utilized in the power grid, as well as introduce the power grid calculation techniques that are targeted

[1]https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-040a
[2]https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-083a

by FDIA, particularly power flow calculation and state estimation. From this, we can outline which components are necessary for the simulation of the power grid and communication infrastructure. At the end of chapter 2, a literature review on simulation tools for simulation tools in the domain of power grid, communication, and co-simulation is conducted to assess whether the tools from the literature can be used to build the simulator. In chapter 3 we will start the process of designing the simulation by first deriving requirements for the simulated system, then creating a system design, and finally creating test cases to evaluate whether the system fulfills the requirements.

RQ 2: Which metrics can be used to evaluate the impact of false data injection attacks on the power grid?

Research question 2 builds directly on the first research question and will be explored both in chapter 2, as well as chapter 4. The literature review from section 2.4 will be used to outline which variables of the system are affected by FDIA and act as a basis for the creation of the metrics. In 4, we will then build on this basis by creating effect metrics for FDIA and building visualization and evaluation tools to convey their effect to the user.

RQ 3: What kinds of FDIA can be conducted on the power grid and how large is their impact on the power grid based on the metrics from RQ2?

In the literature review from section 2.4, several FDIA techniques are collected from the literature and a taxonomy for FDIA is created. The provided information on how these techniques achieve their goals is then utilized for the implementation of a selection of these techniques for the evaluation with the simulator. The implementation and the results of the simulation with these FDIA techniques will be explored in chapter 4.

1.2 Structure

To systematically address these research questions, the thesis is organized as follows:

Chapter 2: State of the Art This chapter provides an overview of the existing literature and technologies related to power grids, power flow calculation, and communication within smart grids. It also covers the current methods for co-simulation of power grids and communication networks, as well as the implications of false data injection attacks.

Chapter 3: Co-Simulation The third chapter details the simulation and co-simulation scope and requirements, the tools utilized for the simulation, and the overall architecture of the simulation. Furthermore, it includes the implementation of the co-simulation and its evaluation according to ISO/IEC/IEEE 29119 standards.

Chapter 4: False Data Injection The False Data Injection chapter provides metrics to measure the impact of FDIA and describes the implementation of both the metrics and a select number of FDIA techniques into the co-simulation. Finally, it will delve into the evaluation of the FDIA techniques based on the created metrics.

Chapter 5: Conclusion and Outlook In the final chapter, we conclude the thesis by answering the research questions based on the findings of the previous chapters. It also provides an outlook on future research directions and potential improvements to the current simulation framework.

2 State of the Art

The power grid has experienced significant transformations through the introduction of new technologies in recent years, evolving from a conventional grid into the advanced *smart grid*. The decentralization of supply, enabled by the availability of photovoltaic panels for private households, along with advanced metering technology that provides grid operators real-time access to grid data, have rendered the traditional top-down power grid a relic of the past (Salman, 2017). However, the implementation of these new technologies necessitates a robust communication network to disseminate metering and measurement data, as well as grid control commands. This chapter offers a comprehensive overview of these critical areas, starting with an in-depth examination of the power grid, including power flow calculation and state estimation, as well as smart metering techniques, particularly the smart meter gateway (SMGW). Understanding the structure and operational principles of the power grid establishes the foundation for subsequent discussions on smart grid innovations.

Within this overview, we explore the pivotal role of communication within the smart grid. Effective communication systems are essential for the real-time monitoring, control, and management of the grid, enabling advanced functionalities and improving overall grid performance (Baimel et al., 2016). Various communication protocols and standards, which ensure interoperability and efficiency, are discussed, along with the key components and challenges associated with smart grid communication infrastructure. The integration and co-simulation of power grids and communication networks are then examined to illustrate how these systems can be utilized to create a power grid and communication co-simulation for measuring and displaying the impact of FDIA.

Additionally, the chapter delves into the concept of FDIAs, a significant cyber threat to smart grids. By understanding the mechanisms and potential impacts of FDIAs as well as being able to simulate those impacts for any LV distribution grid, researchers and practitioners can better develop strategies to mitigate such risks.

Finally, we delve into simulation approaches for power grid and communication networks, as well as their co-simulation. Overall, this chapter aims to provide a detailed and insightful overview of the current state of the art in smart grid technologies, setting the stage for the simulation design and evaluation discussed in subsequent chapters.

2.1 Power Grid Overview and Components

The power grid is an intricate network designed to deliver electricity from producers to consumers with high reliability and efficiency. It comprises multiple components and varying topologies to facilitate efficient electricity transmission and distribution across different geographical and demand landscapes.

2.1.1 Components of the Power Grid

The power grid consists of several key components that maintain the flow of electricity.

Buses are nodes where electrical lines, generation units, or loads converge, serving as centralized points for voltage regulation and power distribution. They are characterized by voltage U, voltage angle φ, active power P, and reactive power Q, defining the system's electrical state and balancing power demand with generation.

Transmission lines transport electricity between buses, linking power generators to consumers, and are defined by parameters such as resistance R, inductance L, and capacitance C, which are essential for predicting voltage drops, power losses, and reactive power flows.

Transformers change voltage levels for optimized power transmission and distribution, either stepping up voltage for long-distance transport or stepping down voltage for safe consumer use. Key parameters include turn ratios a, impedance Z, and tap settings, which influence voltage transformation and reactive power control.

Finally, switchgear and circuit breakers are protective devices that ensure safe and reliable grid operation by controlling, isolating, and protecting electrical circuits. They are critical for disconnecting faulty grid sections, maintaining system integrity, and are crucial for grid stability assessments Schäfer (2023).

2.1.2 Voltage Levels and Transmission / Distribution Grids

The power grid operates across various voltage levels, each serving specific purposes in the transmission and distribution of electricity.

High voltage transmission networks transport electricity over vast distances from power generation sources to primary substations, typically operating at voltage levels ranging from 110 kV to 765 kV or higher. These elevated voltage levels are crucial for reducing energy losses during long-distance transmission and improving efficiency, as they link power plants to regional distribution networks and interconnect national grids.

Medium voltage distribution networks, operating between 1 kV and 36 kV, serve as intermediaries, transporting electricity from substations to local transformers where voltage is stepped down for consumer use, thus playing a pivotal in regional electricity distribution and grid stability through localized load management.

LV distribution networks represent the final stage of the electricity delivery chain, operating at voltage levels below 1 kV, typically at 400/230 V. They are crucial for directly supplying power to residential, commercial, and some industrial consumers, facilitating the integration of distributed energy resources (DERs) like rooftop solar panels and implementing smart grid technologies (Schäfer, 2023).

The LV distribution grid is central to smart grid development, enhancing energy efficiency and reliability using technologies such as smart meters and automated control systems for real-time monitoring and dynamic load management. This focus on LV grids is essential for facilitating consumer engagement and seamlessly incorporating renewable energy sources into the grid, highlighting their significance in the broader context of smart grid transformation as the primary interface between consumers and the larger power infrastructure (Salman, 2017).

2.1.3 Grid Topology

The configuration, or topology, of a power grid significantly influences its operational characteristics and the approach taken in power flow calculations and state estimations.

Radial grids, resembling tree-like structures, are primarily used in LV distribution networks due to their simplicity, cost-effectiveness, and ease of fault isolation. In these systems, each customer has a unique path connecting them back to a substation, making radial grids efficient for connecting residential and small commercial users while maintaining manageable complexity and low operational costs. However, evolving demands such as bidirectional power flow from distributed generation challenge this traditional topology, prompting innovations in grid management and control technologies (Salman, 2017).

In contrast, mesh networks, utilized mainly in high-voltage transmission systems, feature multiple paths between nodes, providing redundancy and enhancing reliability, albeit at a higher material cost.

Ring topologies, often seen in medium voltage networks, create a closed-loop configuration that balances redundancy with lower costs, offering alternative paths for power delivery but with simpler implementation than mesh networks.

Finally, hybrid systems combine elements of radial, mesh, and ring topologies to optimize performance and reliability in response to varying load demands and generation profiles typical of modern distribution systems with distributed energy (Schäfer, 2023).

2.2 Communication in Smart Grid

As a result of the advent of DERs, especially private photovoltaic systems, and the resulting shift from one-way electric power grids to a more decentralized power distribution grid, as well as the increasing usage of the Internet of Things (IoT) and other sensors in the grid infrastructure, the amount of available information in and about the grid has led to new technologies for the grid control, monitoring, and planning. The result is the *smart grid*, which encompasses not only the physical power grid but also includes the information and communication technology (ICT) systems to form a cyber-physical system (Salman, 2017). In this section, the main components of these ICT systems will be explored. In doing so, we will particularly focus on the components that are of relevance for FDIA.

Table 2.1: Comparison between Existing Grid and Smart Grid from Fang et al. (2012)

Existing Grid	**Smart Grid**
Electromechanical	Digital
One-way communication	Two-way communication
Centralized generation	Distributed generation
Few sensors	Sensors throughout
Manual monitoring	Self-monitoring
Manual restoration	Self-healing
Failures and blackouts	Adaptive and islanding
Limited control	Pervasive control
Few customer choices	Many customer choices

The table 2.1 taken from Fang et al. (2012) provides a comparison between the attributes of the existing grid and the smart grid. The existing grid relies on electromechanical technology, while the smart grid employs digital technology. Communication in the existing grid is one-way, whereas it is two-way in the smart grid. Power generation is centralized in the existing grid but has become more distributed in the smart grid, due to the advent of privately owned PV plants and other generative facilities. Sensor deployment is minimal in the existing grid with few sensors, in contrast to the widespread sensor deployment throughout the smart grid. Monitoring in the existing grid is carried out manually, while the smart grid features self-monitoring capabilities such as advanced metering infrastructure, e.g. SMGWs. The restoration processes in the existing grid in case of grid faults have to be carried out manually, whereas the smart grid is designed to be self-healing through the usage of congestion management and state estimation. The existing grid is prone to failures and blackouts, while the smart grid adapts to circumstances and can operate in an islanding mode, meaning that a single low-voltage smart grid strain could run separately from higher voltage levels as a so-called microgrid. Control is limited in the existing grid, but pervasive in the smart grid. Finally, the existing grid offers few customer choices, whereas the smart grid provides many customer choices such as dynamic pricing and peer-to-peer energy trading.

As can be seen from this comparison, the integration of advanced communication technologies within smart grids plays a crucial role in enhancing the efficiency, reliability, and sustainability of electricity distribution systems.

2.2.1 Core Components of Smart Grid Communication

The communication infrastructure in smart grids is composed of several key components, each contributing to the system's functionality which we will allude to in the following:

Advanced Metering Infrastructure (AMI): AMI is the backbone of modern energy management systems, particularly in providing the grid operator (GO) with real-time information about the state of the grid, which can be utilized in power flow calculations or state estimations, which will be discussed in section 2.3. Smart meters are not only used as a more modern replacement of meters to capture customer consumption data, but also provide measurement data and live communication with the GO. The two-way communication enabled by AMI forms a vital link between utilities and consumers, fostering greater transparency and engagement on the client side, but also enabling more accurate predictions about the state of the grid for the GO. (Rashed Mohassel et al., 2014)

Supervisory Control and Data Acquisition (SCADA): SCADA systems form the central nervous system of power grids, tasked with the monitoring and control of diverse network components in real time. In traditional power grids, SCADA systems were mostly employed in the medium and high voltage transmission grids, but with the increase in AMI and sensors in the distribution grid, also found their way into the distribution grid. By aggregating data from widespread sensors and field devices and most importantly AMI, SCADA systems provide GOs with a holistic view of network performance, enabling timely interventions and strategic planning. (Cheng et al., 2024)

Phasor Measurement Units (PMUs): PMUs are precision instruments that offer time-synchronized measurements of electrical waveforms, capturing voltage and current phasors. The high temporal resolution provided by PMUs allows for detailed grid state analysis, supporting enhanced situational awareness and fault localization. By incorporating PMU data into state estimation processes, operators can achieve a granular understanding of the grid's dynamic behavior. (Zivanovic and Cairns, 1996)

Distributed Energy Resources (DERs): The proliferation of DERs, such as solar photovoltaic systems, wind turbines, and battery energy storage, demands effective communication and control solutions for seamless integration into the grid. Communication with DERs enables coordinated operations, supports distributed generation management, and facilitates grid balancing, contributing to a more resilient and sustainable energy system.

Communication Networks: To facilitate the flow of information across the grid, a diverse array of communication technologies is employed. Wireless solutions, including cellular networks like LTE and more recently 5G, Wi-Fi, and other technologies like LoRa and ZigBee, complement traditional wired options like fiber optics and power line communication. Hybrid network architectures are often adopted to optimize coverage, reliability, and data transmission efficiency, ensuring robust connectivity across varied grid environments.

In the scope of this work, we will focus mainly on the components of AMI and SCADA and their interaction. While there are extensive studies on the increased accuracy of state estimations with the addition of PMUs as shown in the review of Cheng et al. (2024), the goal of this work is to lay a foundation for simulating and assessing the effects of FDIA in a distribution grid. Due to time constraints, the integration of PMUs or the focus on specific communication channels like 5G or PowerLine communication, while a promising field of research, will therefore not be conducted in the scope of this work and will be left for future projects to address. The simulation, however, will be developed in such a way that it can easily be extended in such directions.

2.2.2 Advanced Metering Infrastructure

AMI is a system that plays a pivotal role in modernizing and enhancing the capabilities of electrical grids. At its core, AMI comprises smart meters, robust communication networks, and sophisticated data management systems. These components collaborate to facilitate two-way communication between utility providers and consumers, which is crucial for real-time grid monitoring, management, and optimization.

Smart meters, integral to AMI, are deployed at consumer premises to record energy consumption at high temporal resolutions, often in intervals of 15 minutes or less. This granularity of data is instrumental for accurate billing, demand forecasting, and energy analysis. Furthermore, the real-time feedback provided by smart meters allows consumers to adapt their energy usage patterns, contributing to overall energy efficiency and demand-side management.

A significant application of AMI data is in the domain of state estimation, which is central to effective grid management and optimization. State estimation involves determining the electrical state of the grid—specifically voltages, currents, and phase angles—across its various nodes and will be explained in detail in section 2.3.2. The high granularity and frequency of data provided by AMI enhance the accuracy of state estimation models, allowing more precise monitoring of grid conditions. This in turn enables operators to identify and mitigate issues such as load imbalances, voltage deviations, and potential fault conditions with greater confidence and speed. However, since the AMI is not under total control by the GO, it provides an attack surface for FDIA, which will be explored in this work, especially in section 2.4 and chapter 4.

AMI also supports the dynamic allocation and dispatch of resources by providing continuous data on consumption trends and grid performance. This facilitates the integration of DERs, such as solar and wind power, by offering up-to-date insights into generation capacities and load requirements. The integration of such data allows for the adaptive real-time management of DERs, improving grid stability and efficiency.

Moreover, AMI systems enable the deployment of advanced grid control strategies, such as demand response programs, where consumers are incentivized to alter their consumption during peak periods or when grid stability is at risk. The success of these programs relies heavily on the timely and accurate data generated by smart meters. (Salman, 2017)

SMGWs in Germany

In Germany, where we lay the focus of this study, the implementation of AMI is regulated by a framework that prioritizes security, privacy, and interoperability. Central to this framework is the SMGW, a pivotal component in ensuring secure and standardized data communication within the grid.

SMGWs serve as communication hubs that manage the flow of information between smart meters and external entities, such as utility providers and authorized third parties. They act as a secure intermediary, enabling the aggregation and transmission of detailed consumption data, which is crucial for grid management and customer engagement.

A key application of SMGWs is in the realm of dynamic tariff structures. By facilitating real-time data exchange, SMGWs enable utilities to implement and manage complex tariff models such as time-of-use pricing, critical peak pricing, and real-time pricing. These tariff structures are designed to incentivize energy consumption behaviors that align with grid stability and efficiency objectives. Consumers, informed by real-time usage data accessible through SMGWs, can adapt their consumption patterns to benefit from lower tariffs during off-peak periods, thus reducing overall energy costs and contributing to peak load management.

The deployment and operation of SMGWs in Germany are governed by precise requirements outlined in technical guidelines, particularly BSI TR-03109 (BSI, 2021). These guidelines ensure robust security measures, such as end-to-end encryption and robust authentication protocols, to protect sensitive consumer and grid data from unauthorized access and cyber threats. Compliance with these requirements is mandatory, ensuring that data integrity and confidentiality are maintained across all communication channels.

BSI (2021) define several tariff application cases (TAC), which can be understood as communication modes between the SMGW and the GO, not only for contractual purposes but also for supplying power grid measurements in high frequencies, which are depicted in table 2.2. Three TACs are used to provide periodic measurements to the GO, namely TAC 1, 7, and 14. Since the time resolution should be as high as possible for accurate state estimations, we will concentrate the simulation on TAC 7 and 14, and model the simulation measurement data, and process accordingly.

Table 2.2: Tariff Application Cases (TAC) defined in BSI (2021)

TAC	**Use Case**	**Periodicity**
TAC1	Data-Economic Tariffs	1 month
TAC2	Time-Variable Tariffs	variable
TAC6	On-Demand Meter Reading	on demand
TAC7	**Periodic Meter Reading**	**15 minutes**
TAC9	Retrieval of Current Feed-In	on demand
TAC10	Retrieval of Grid State Data	on demand
TAC14	**High-Frequency Periodic Measurement**	**1 minute**

In addition to security, SMGWs are designed to ensure interoperability across various manufacturers and utility systems, which is critical for seamless integration into the broader energy infrastructure. Adherence to specifications provided by the Forum Netztechnik/Netzbetrieb facilitates compatibility and interoperability between devices from different suppliers.

The German regulatory framework mandates the installation of SMGWs for large consumers and prosumers, defined as those with annual energy consumption exceeding 6,000 kWh and operators of generation plants above 7 kW. This phased rollout approach ensures that critical infrastructure is in place to support the transition towards a more digital and interconnected energy system.

2.2.3 SCADA Systems and Integration with AMI/SMGW

In the context of smart grids, SCADA systems are intricately integrated with AMI and SMGWs to enhance grid management capabilities. The integration with SMGWs is particularly significant, as it facilitates seamless communication and data exchange between end-user metering devices and central utility systems. This integration allows SCADA to leverage the granular, real-time data collected by SMGWs to improve the accuracy and efficiency of grid operations.

Through SMGWs, SCADA systems gain access to detailed consumption data and additional grid parameters such as voltage levels, phase angles, and power. This dataset is crucial for state estimation processes and facilitates predicting the current operating state of the grid based on the provided grid data.

State estimation using SCADA data is further enhanced by the real-time capabilities of PMUs, which provide synchronized phasor measurements across the grid. By incorporating this precise data, SCADA systems can refine grid models and improve situational awareness, ensuring that operators have a comprehensive view of the grid's dynamic behavior.

The integration of SCADA systems with SMGWs also supports advanced grid control strategies, such as demand response programs and dynamic tariff applications. By accessing real-time data on consumption patterns and load conditions, SCADA can orchestrate demand-side management initiatives, optimizing energy use during peak periods and enhancing grid stability. This capability is vital for accommodating the integration of DERs, where the balance between supply and demand must be carefully managed to maintain grid reliability.

Moreover, the secure communication protocols utilized in SMGWs ensure that data exchanged with SCADA systems is protected against unauthorized access and cyber threats. This security framework is critical for safeguarding the integrity and confidentiality of sensitive data, facilitating trust in the system's operation.

2.2.4 Challenges in Smart Grid Communication

Integrating cutting-edge communication technologies into smart grid operations presents several challenges (Salman, 2017), which will be considered in the requirements for the co-simulation:

Data Volume and Latency: Smart grids generate large volumes of data that necessitate efficient transmission and processing in near real-time. Reducing data latency and maintaining high reliability is crucial for preserving grid stability and performance.

Interoperability: Ensuring that disparate technologies and vendors work cohesively is a significant challenge. Standardized communication protocols and interfaces are vital for achieving seamless interoperability across the smart grid ecosystem.

Scalability: As smart grid infrastructures expand to accommodate increasing device connectivity and data traffic, the communication networks must be scalable to support this growth without compromising performance.

Security: With the rise of digital threats, safeguarding the smart grid against cyber-attacks is paramount. Communication networks must be fortified with robust security measures to prevent unauthorized access, data breaches, and operational disruptions.

2.2.5 Emerging Trends in Smart Grid Communication

Recent technological advancements are reshaping smart grid communication, highlighting key trends such as 5G, the Internet of Things (IoT), and decentralization:

The emergence of 5G networks provides exceptional capabilities for smart grids, offering high-speed, low-latency data exchanges that are critical for real-time grid operations. The expansive connectivity facilitated by 5G supports the efficient management of distributed energy resources and enhances advanced metering infrastructure applications. Additionally, 5G networks significantly bolster the capabilities of SCADA systems, extending their operational reach and reliability.

The integration of IoT devices into smart grids enables the seamless interconnection of diverse grid components, enhancing system visibility, situational awareness, and predictive maintenance. IoT sensors facilitate continuous data collection, empowering GOs to monitor and control grid parameters more effectively. Through IoT-enabled data analytics, operators can optimize grid operations and enhance energy efficiency.

The trend towards grid decentralization is driven by increased adoption of DERs, such as community solar projects and distributed wind installations. Decentralized grid models promote localized control and resilience, allowing communities to manage their energy resources more effectively. Advanced communication technologies like 5G and IoT are crucial in supporting decentralization, facilitating two-way power flows, and dynamic grid management. (Salman, 2017)

These technological trends are pivotal in advancing energy distribution frameworks, enhancing smart grid functionalities, and promoting sustainable power systems that are resilient, efficient, and adaptable to future energy needs. And the simulation will be designed to facilitate their future integration into the system.

2.3 Power Grid Calculation

The analysis of the power grid and its current state relies on a set of fundamental variables that describe the state and behavior of the network. These are essential for conducting power flow calculations and performing state estimation in LV distribution grids. In the scope of this work, we will rely mostly on the variables related to buses or nodes in the network outlined in section 2.1, since the inclusion of e.g. transformer calculations would increase the complexity immensely.

While the fundamental laws of electricity hold for all electric systems, the electric power grid mostly operates with three-phase current, which is why we need to operate with the complex number versions of the variables at hand. Ohm's law using complex numbers (equation 2.1) equates complex voltage to complex current multiplied by impedance $\underline{Z}$, the complex equivalent of resistance.

$$\underline{U} = \underline{Z} \cdot \underline{I} \tag{2.1}$$

Another important variable for the grid calculations is the admittance $\underline{Y}$, which is the reciprocal of the impedance.

$$\underline{Y} = \frac{1}{\underline{Z}} \tag{2.2}$$

In contrast to DC current, the complex electric power, denoted as $\underline{S}$, in three-phase current systems consists of two components: active power P and reactive power Q, where P is the real component and Q is the imaginary component of $\underline{S}$.

$$\underline{S} = P + j \cdot Q \tag{2.3}$$

In the calculations carried out for electric power grid analysis, the nodes or buses are abstracted and the power inputs and outputs balanced, such that each bus can be described utilizing only its voltage magnitude U, the voltage angle φ, the balanced active power P and the balanced reactive power Q. (Schäfer, 2023)

2.3.1 Power Flow Calculation

Power flow calculation is a fundamental tool used by electrical engineers to determine the operating conditions of a power grid. It involves the calculation of voltage, current, active power, and reactive power in each part of the network under a given load condition. We will utilize the Newton-Raphson method as described in (Tinney and Hart, 1967), which is an iterative method that can solve the power flow problem efficiently in a small number of iterations. As a major part of the calculation process, we utilize the nodal admittance matrix $\boldsymbol{Y}$, containing the topology and admittances of the grid. The matrix contains the admittances of the available connections between the nodes and is calculated in the following manner:

$$\boldsymbol{Y} = \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1j} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2j} & \cdots & y_{2N} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ y_{i1} & y_{i2} & \cdots & y_{ij} & \cdots & y_{iN} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nj} & \cdots & y_{NN} \end{pmatrix} \tag{2.4}$$

The admittances on the main diagonal are calculated in the following way:

$$\underline{y}_{ii} = \underline{Y}_{ii} = \sum_{j=1}^{N} (\underline{Y}_{0,ij} + \underline{Y}_{ij}) \tag{2.5}$$

Where the self-admittances $\underline{Y}_{0,ij}$ are the sums of all connected admittances plus the node's admittance. The values off the main diagonal $\underline{Y}_{ij}$ are the values of the connecting admittances between the two nodes i and j multiplied by (-1). The nodal admittance matrix is sparse in most cases, especially in radial grids, since most nodes are only connected to one or two other nodes. Together with Ohm's law, we get the following system of equations:

$$\begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1j} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2j} & \cdots & y_{2N} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ y_{i1} & y_{i2} & \cdots & y_{ij} & \cdots & y_{iN} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nj} & \cdots & y_{NN} \end{pmatrix} \cdot \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_i \\ \vdots \\ U_N \end{pmatrix} = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_i \\ \vdots \\ I_N \end{pmatrix} \tag{2.6}$$

Which can be depicted in a more compact form as

$$\boldsymbol{Y} \cdot \boldsymbol{u} = \boldsymbol{i} \tag{2.7}$$

This is the linear system of equations. However, we need to additionally consider non-linear equations for calculating the power flow. For the creation of the non-linear system equation, we assume constant node power for each node. This yields the following system of equations:

$$\boldsymbol{s} = 3 \cdot diag(\boldsymbol{u}) \cdot \boldsymbol{i} \tag{2.8}$$

Combined with equation 2.7 and using phase to phase voltage ($\underline{U}_{pp} = \sqrt{3} \cdot \underline{U}_i$) to get rid of the factor 3, we get the system equation

$$\boldsymbol{s} = diag(\boldsymbol{u}_{pp}) \cdot \boldsymbol{Y} \cdot \boldsymbol{u}_{pp} \tag{2.9}$$

In the power flow calculation, each bus is assumed to have a known value for $\underline{S}$. A slack node is selected, ideally a node in a central position in the grid, such as the transformer low voltage node, which is assigned a fixed voltage and voltage angle. The power flow calculation is then defined as follows:

$$f(x) - s_{iv} = 0 \tag{2.10}$$

f is a function vector describing the relation between the node power and the node voltages, x is a vector of the state variables $\boldsymbol{u}$ and $\boldsymbol{\varphi}$, and s_{iv} is the vector of the input variables $\boldsymbol{p}$ and $\boldsymbol{q}$.

For the usage of the Newton-Raphson method, we need to segment the system of equations given in equation 2.9 into two systems of equations. The equations will be displayed in cartesian coordinates instead of complex forms, which will facilitate the derivation of the equations in later steps.

$$\underline{U}_i = U_i \cdot e^{j \cdot \varphi_i} = E_i + j \cdot F_i \tag{2.11}$$

$$\underline{y}_{ik} = g_{ik} + j \cdot b_{ik} \tag{2.12}$$

$$\underline{S}_i = P_i + j \cdot Q_i \tag{2.13}$$

In reformulating the system of equations, we recognize the complex power $\underline{S}_i$ for each node is given by:

$$\underline{S}_i = U_i^2 \cdot \underline{y}_{ii}^* + \underline{U}_i \cdot \sum_{k \in N_i} \underline{y}_{ik}^* \cdot \underline{U}_k^* \tag{2.14}$$

The Newton-Raphson method starts with building a Taylor series from equation 2.10, from which only the linear part is taken into account. This leaves us with

$$\boldsymbol{F}(\boldsymbol{x}^{(\nu)}) \cdot \Delta\boldsymbol{x}^{(\nu)} = \Delta\boldsymbol{s}^{(\nu)} \tag{2.15}$$

Where $\boldsymbol{F}$ represents the Jacobian matrix, the partial derivatives of all node active and reactive powers by all unknown variables, $\Delta\boldsymbol{x}$ represents the change of the state variables, and $\Delta\boldsymbol{s}$ represents the difference between the calculated and the start values of the node's active and reactive powers. ν indicates the iteration step. This can then be described in the following system of equations:

$$\boldsymbol{F} \cdot \begin{pmatrix} \Delta\boldsymbol{\varphi} \\ \frac{\Delta \boldsymbol{u}}{diag(\boldsymbol{u})} \end{pmatrix} = \begin{pmatrix} \boldsymbol{H} & \boldsymbol{N} \\ \boldsymbol{J} & \boldsymbol{L} \end{pmatrix} \cdot \begin{pmatrix} \Delta\boldsymbol{\varphi} \\ \frac{\Delta \boldsymbol{u}}{diag(\boldsymbol{u})} \end{pmatrix} = \begin{pmatrix} \Delta\boldsymbol{p} \\ \Delta\boldsymbol{q} \end{pmatrix} \tag{2.16}$$

With $\boldsymbol{H}$ and $\boldsymbol{J}$ being the partial derivatives of the nodes active and reactive powers by their voltage angles and $\boldsymbol{N}$ and $\boldsymbol{L}$ being the partial derivatives of the nodes active and reactive powers by the absolute values of their voltages, respectively. $\Delta\boldsymbol{\varphi}$ denotes the vector of the change in voltage angles of each node, whereas $\frac{\Delta\boldsymbol{u}}{diag(\boldsymbol{u})}$ is the change in voltage of each node divided by its voltage value.

The Jacobian matrix and the vector have to be recalculated and the system of equations will be solved with Gaussian elimination in every iteration. The solution is reached when the following criteria are fulfilled:

$$\left|P_{i,iv} - P_i^{(\nu)}\right| \leq \varepsilon_P \tag{2.17}$$

$$\left|Q_{i,iv} - Q_i^{(\nu)}\right| \leq \varepsilon_Q \tag{2.18}$$

with ε denoting the accuracy limits of active and reactive powers, respectively. (Schäfer, 2023)

2.3.2 State Estimation

State estimation is a critical component in the monitoring and control of power systems, providing a real-time snapshot of the system's operational state. This process involves estimating the voltage magnitudes and phase angles across the network buses, formulated as a weighted least squares problem (Schweppe and Wildes, 1970). By integrating measurements and network configurations, state estimation ensures the observability and adequacy of system models requisite for decision-making processes in grid operations.

The fundamental equation characterizing state estimation is:

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{e} \tag{2.19}$$

Where $\boldsymbol{z}$ represents the vector of measurements, $\boldsymbol{h(x)}$ is the vector function mapping state variables $\boldsymbol{x}$ (voltages and voltage angles) to measurements, and $\boldsymbol{e}$ denotes the measurement error, typically modeled as a Gaussian random variable.

The objective of state estimation is to minimize the weighted sum of squared errors, formulated as:

$$J(\boldsymbol{x}) = (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}))^T \boldsymbol{W} (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x})) \tag{2.20}$$

Here, $\boldsymbol{W}$ is the diagonal matrix of weights, reflecting the inverse of the variance of the measurement errors.

By applying Gauss-Newton's method (Wang, 2012) to iteratively solve this optimization problem, we update the state estimates in each iteration k:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{\Delta x}^{(k)} \tag{2.21}$$

where $\Delta x^{(k)}$ is determined by:

$$\boldsymbol{G} \cdot \boldsymbol{\Delta x}^{(k)} = \boldsymbol{H}^T \boldsymbol{W}(\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}^{(k)})) \tag{2.22}$$

$\boldsymbol{G}$ is the gain matrix, computed as:

$$\boldsymbol{G} = \boldsymbol{H}^T \boldsymbol{W} \boldsymbol{H} \tag{2.23}$$

and $\boldsymbol{H}$ is the Jacobian matrix of $\boldsymbol{h}$ with respect to the state vector $\boldsymbol{x}$, which lets us formulate the function for $\boldsymbol{\Delta x}^{(k)}$ as follows:

$$\boldsymbol{\Delta x}^{(k)} = (\mathbf{H}^T \mathbf{W} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{W} \mathbf{z} \tag{2.24}$$

or

$$\boldsymbol{\Delta x}^{(k)} = (\mathbf{G})^{-1} \mathbf{H}^T \mathbf{W} \mathbf{z} \tag{2.25}$$

The iterative process continues until the changes in the state estimate $\boldsymbol{\Delta x}^{(k)}$ fall below a stipulated convergence threshold. The convergence is assessed using:

$$\left\| \boldsymbol{\Delta x}^{(k)} \right\| \leq \epsilon \tag{2.26}$$

Where ϵ is a predefined small number, indicating the error tolerance for convergence.

Upon convergence, $\Delta \boldsymbol{x}^{(k)}$ is denoted as $\hat{\boldsymbol{x}}$, the converged state vector. The resulting state estimates provide a comprehensive understanding of the grid's current condition, contributing to the efficiency and reliability of power system operation.

State estimation facilitates critical applications such as detecting faulty measurements, grid monitoring, and predictive analytics. Enhancing situational awareness, underscores the grid's resilience against operational anomalies or disturbances, thereby supporting real-time decision-making in energy management systems (Abur and Expósito, 2004).

While power flow calculation can provide information about the grid when all active power and reactive power values for busses are known, this is often not the case in LV distribution grids, which rely on measurement data that is not always available for every bus or line. This is why state estimation is utilized, which can provide information about the grid state with a lower number of available measurements. (Schäfer, 2023)

2.3.3 Bad Data Detection

Bad measurements can occur due to various reasons such as meter failures or malicious attacks, such as FDIA. To protect the accuracy of state estimation, techniques have been developed to detect these bad measurements. Normally, good meter readings give estimates close to the actual values, while bad ones can cause significant deviations. This creates an inconsistency between good and bad measurements.

Researchers propose calculating the measurement residual $\boldsymbol{z} - \mathbf{H}\hat{\boldsymbol{x}}$, which is the difference between observed measurements and estimated measurements. The 2-Norm of this residual, $\|\boldsymbol{z} - \mathbf{H}\hat{\boldsymbol{x}}\|$, is used to detect bad measurements. If this value exceeds a certain threshold τ, it indicates the presence of bad measurements. Assuming all state variables are independent and meter errors follow a normal distribution, it can be shown that $\|\boldsymbol{z} - \mathbf{H}\hat{\boldsymbol{x}}\|^2$, denoted as $L(\boldsymbol{x})$, follows a chi-squared distribution with $v = m - n$ degrees of freedom.

A χ^2 test is then often used for bad data detection because it allows us to statistically determine if the squared 2-Norm of the residual exceeds a certain threshold. This threshold τ can be determined through a hypothesis test with a significance level α. This means that if $L(\boldsymbol{x}) \geq \tau^2$, it indicates bad measurements with a false alarm probability of α. The χ^2 test provides a rigorous and statistically sound method to identify inconsistencies in the measurements, ensuring the reliability of the state estimation process.

There are other methods to detect bad measurements, such as using the normalized infinity-norm of the residual (Abur and Expósito, 2004). However, we focus on the χ^2 method as it is most commonly used. (Liu et al., 2011)

2.4 False Data Injection Attacks on Power Grid Systems

Building upon the state estimation framework outlined in Section 2.3.2, Liu et al. (2011) first highlights a significant cyber threat known as false data injection attacks. These attacks exploit vulnerabilities in the state estimation process, which is fundamental for monitoring and controlling power systems.

An attacker aims to alter the state vector $\boldsymbol{x}$, which contains voltage magnitudes and phase angles, by injecting false data into the measurement vector $\boldsymbol{z}$. By doing so, the attacker compromises the integrity of the estimated state without raising alarms in standard detection mechanisms.

Non-Generalized FDIA. The compromised measurement vector $\hat{\boldsymbol{z}}$ after an FDI attack can be expressed as:

$$\hat{\boldsymbol{z}} = \boldsymbol{z} + \boldsymbol{a}, \tag{2.27}$$

where $\boldsymbol{a}$ is the attack vector. Since the measurement vector is subject to the χ^2 test for bad data detection, the attacker designs $\boldsymbol{a}$ such that the altered measurements appear consistent with the unaltered measurements and are not detected by the bad data detection test, leading to:

$$\hat{\boldsymbol{z}} = \boldsymbol{H}\hat{\boldsymbol{x}}, \tag{2.28}$$

To ensure that the attack vector $\boldsymbol{a}$ remains undetected,$\boldsymbol{a}$ is constructed to lie within the null space of the Jacobian matrix $\boldsymbol{H}$ to satisfy:

$$\boldsymbol{a} = \boldsymbol{H}\boldsymbol{c}, \tag{2.29}$$

for some nonzero vector $\boldsymbol{c}$, allowing the residual from state estimation to remain unchanged. Hence, the condition:

$$\|\hat{\boldsymbol{z}} - \boldsymbol{H}\hat{\boldsymbol{x}}\| = \|\boldsymbol{z} - \boldsymbol{H}\boldsymbol{x}\| \leq \tau \tag{2.30}$$

Remains true, enabling the adversary to manipulate the state estimates covertly (Liu et al., 2011). Liu et al. (2011) differentiate between two types of FDIA, designating the type of attack where the residual stays unchanged after the injection as FDIA. Since the residual stays unchanged, the bad data detection cannot distinguish between the unaltered and altered data in the non-generalized FDIA. The non-generalized FDIA, however, requires the attacker to have full information about the grid, since knowledge about the Jacobian matrix $\mathbf{H}$ is needed to calculate an attack vector.

Generalized FDIA. However, it is also possible to bypass the bad data detection by exploiting the nature of the bad data detection's underlying χ^2-test, which tolerates small measurement errors. As long as the following equation is satisfied:

$$\|\hat{\boldsymbol{z}} - \boldsymbol{H}\hat{\boldsymbol{x}}\| \leq \tau \tag{2.31}$$

but

$$\|\hat{\boldsymbol{z}} - \boldsymbol{H}\hat{\boldsymbol{x}}\| \neq \|\boldsymbol{z} - \boldsymbol{H}\boldsymbol{x}\| \tag{2.32}$$

The attack vector will still pass the bad data detection, because of the measurement error tolerance in τ.

This is designated by Liu et al. (2011) as generalized FDIA. Since in the scope of this paper we will discuss and simulate both the generalized and non-generalized versions, but put our focus mostly on the generalized FDIA, we will use different terminology and designate the non-generalized FDIA as such. The term FDIA will not specifically be used for the non-generalized version of the attack but as a term for both kinds of attacks.

The are some different goals that the attacker may try to achieve with this attack. An attacker with financial motivation might want to manipulate measurement data to lower their power bill and engage in energy theft. They could also try to manipulate the underlying pricing algorithm by creating the illusion of high renewable energy production and profit from lower energy prices. The most dangerous kind of attacks on the availability and integrity of the power grid, however, are attacks that aim to destabilize the power grid by overloading the transformer or transmission lines, which is why we will focus on this motivation in this work. There is a plethora of manners in which the attacker can try to achieve this goal of destabilization, which will be categorized in the upcoming section.

2.4.1 FDIA Taxonomy

To describe the different categories of FDIAs, we adapt the taxonomy from Reda et al. (2022) with some changes. The adapted taxonomy is depicted in figure 2.1.

We categorize FDIA attack models in four major categories: the level of available information available to the attacker, the available resources - more concretely the number or fraction of compromised buses - of the attacker, the targeting of the attack or its lack thereof, and the underlying system model that is to be compromised.

Information Level

In the information category, we differentiate between three levels of available information:

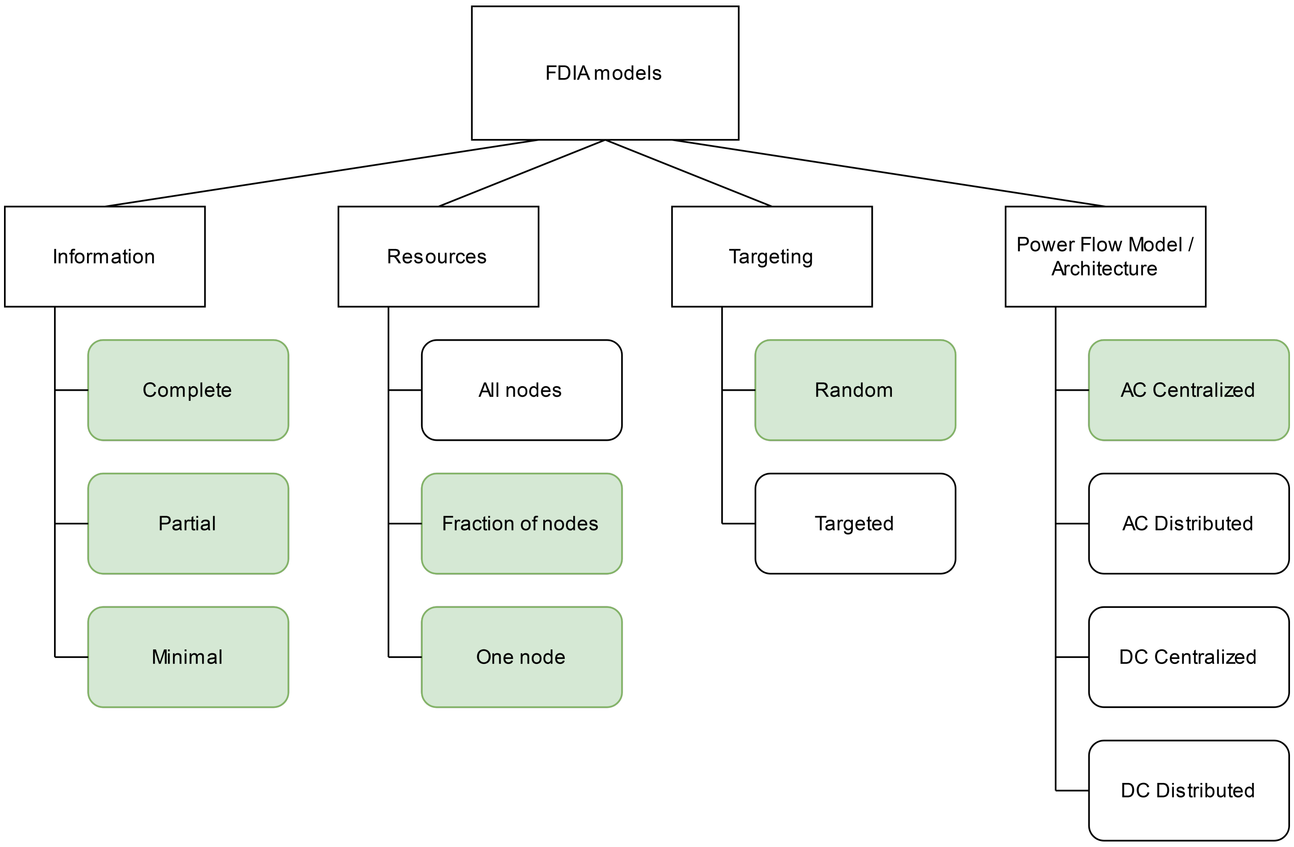

Figure 2.1: FDIA taxonomy

1. **Complete system information**: In this scenario, the attacker has full access to any information available to the grid operator. They know the exact grid topology, and have access to all measurement data, as well as the Jacobian matrix and nodal admittance matrix. With this information, the attacker can calculate random and targeted attack vectors to destabilize the power system through mathematical operations, as first described by Liu et al. (2011). Though this scenario is often assumed in FDIA literature, it is highly unlikely that any attacker would have access to all this information.

2. **Partial system information**: A well-informed attacker may have access to some of the system information, that they could have gathered using cyber attacks or through observation. It is e.g. thinkable, that an attacker could reconstruct the grid topology from the distribution of cable boxes and the placing of houses in a specific area, or that they could have intercepted all or a fraction of the incoming measurements of the other buses in the system. This would allow the attacker to make more informed decisions in creating an attack vector and thus improve its effectiveness (Li and Wang, 2019).

3. **Minimal system information**: An attacker, e.g. a person wanting to lower their power bill or a single attacker with little resources, could try to conduct an FDIA with just the information provided from the bus or buses they control. The information available to the attacker would then mainly be the measurement data from these buses. This could be used by the attacker

to optimize their attack vector based on the effects of previous attacks on their bus (Yu and Chin 2015, Zhang et al. 2016).

For an attacker, the more information about the system they try to compromise they have, the easier it is to create a targeted attack vector. Though effective attacks can be conducted with limited information access, the more data available, the better for the attack vector calculation.

Resource Availability

Similarly, we categorize the level of resources available to the attacker. This manifests in particular in the number of buses they can compromise to inject false measurement data. Analogous to the information category, we differentiate between three levels of resources available.

1. **All buses**: In an ideal attacking scenario[1], the attacker can manipulate the measurement data of all buses in the grid. This provides the most control and flexibility in calculating attack vectors because of the highest possible number of variables. Akin to the complete system information, however, this scenario, while being the ideal one for the attacker, is also the least probable scenario, which is why it will not be considered in the simulation process.
2. **Fraction of buses**: This scenario considers an attacker who can compromise multiple buses in the grid. This can range from controlling two buses, e.g. the attacker's bus and one of an accomplice, to controlling all buses in the system except for the buses controlled by the grid operator, e.g. the cable boxes and power station buses. The more buses the attacker can control, the higher the probability of creating an attack vector that can destabilize the system (Liu et al., 2011). However, not only the number of buses available to the attacker but also the placement of those buses in the grid is important to the effectiveness of the FDIA, which will be explored in later sections. We will conduct experiments mostly in variations of this resource category since it appears to be the most realistic in a real-world attack.
3. **Single bus**: On the lowest end of available resources, we consider the single bus attack. Since the attacker can only compromise the measurement of its bus, the variables to construct the attack vector are highly limited. The single-bus scenario will also be studied in the simulation process.

Targeting

Another important aspect of the categorization of FDIA is the targeting of the attack. In this case, targeting is not used as the term to describe the overall goal of the FDIA, since we assume this is to destabilize the grid but to categorize how specific the attacker is in trying to achieve this goal. We differentiate between two categories:

[1]from the attacker's viewpoint

1. **Random**: In a random FDIA, the attacker tries to attain their goal by any means possible. They do not focus on specific buses or specific state variables but pursue an attack vector that accomplishes grid destabilization without any restrictions. This category is not to be confused with the random data FDIA, which injects random data into the system.
2. **Targeted**: In the targeted FDIA scenario, the attacker sets certain subgoals and limitations in the attack vector calculation process. This could be a specific targeting of one or several buses, e.g. the power station or cable box bus for maximum impact, or a specific value that they want to influence, e.g. the voltage angle or the reactive power. While this may lead to more effective or targeted results, the addition of conditions makes the calculation of an attack vector more difficult.

In the simulation, we will concentrate on the random FDIA to assess the quality of the simulation and the developed metrics. Since the integration of targeted attacks would require more specific metrics than the random FDIA, we will consider them to be out of scope for this particular study.

Power Flow Model / Architecture

While information level, available resources, and targeting play a crucial role in the categorization of FDIA models, the underlying system that is attacked by the attacks also plays a role in the calculation of the attack vectors, since the algorithms for power flow calculation and state estimation may differ. We differentiate between two subcategories, namely power flow model and architecture.

Power Flow Model:

1. **DC**: While DC power flow models are easier to solve and utilize, they use a higher level of abstraction and thus yield inferior results to AC power flow (Overbye et al., 2004). In the FDIA literature, a majority of approaches are based on the DC power flow approach (Reda et al., 2022). However, since AC power flow is used in most real-world applications and the available computing power makes solving AC power flows more accessible, we will not be using DC power flow in the simulation of FDIA.
2. **AC**: AC power flow models take into account more variables than the idealized DC power flow and thus are more difficult to solve. Since the level of abstraction is lower, they offer more accurate results, which is why they are used in most real-world applications nowadays (Schäfer, 2023). Especially with smaller numbers of buses in the grid, AC power flow calculation can be utilized without large computing performance reductions relative to DC power flow.

Architecture

Another point of consideration is the architecture of the system, and more specifically the underlying architecture on which the grid operator's calculations are being made. We differentiate between centralized systems and the newly emerging decentralized systems, specifically regarding state estimation.

1. **Centralized System**: The centralized system describes an architecture where all measurement data is received by the grid operator, who then conducts the calculations centrally. This system is still predominant in most power grids and will be utilized in the simulation.

2. **Decentralized System**: A plethora of research is being conducted in the decentralization of the power grid, e.g. through the usage of multi-agent systems (Adjerid and Maouche, 2020) or other decentralized approaches for state estimation (Singh and Pal 2014, Liu and Yu 2018, Yin et al. 2019). While this is a promising research field, these approaches have not yet found footing in real-world applications and therefore will be relegated for future research endeavors.

When comparing this taxonomy with the one provided by Reda et al. (2022), we can see that the differentiation categories of the FDIA model have been slightly altered. The power flow model and network architecture have been combined into one category since the categorization in this regard is not as important in our specific use case, where we will only use centralized AC systems. The category of construction methods, however, has been split up into the three categories information and resource level, and targeting, since we deem these to be the most important distinguishing factors and the separation of categories makes it easier to categorize a given FDIA model for our purposes. We will explore the process of creating attack vectors for different information and resource levels with and without specific targeting in the upcoming sections.

The categories that are implemented in some form in the implementation are highlighted in green in figure 2.1 to provide an overview of the FDIA techniques based on the taxonomy.

2.4.2 Calculation of Attack Vectors

While the categories of FDIA we want to discuss in this work have been laid out in the previous section, the creation of the attack vectors in each of the scenarios will be described in this section. We will start with the FDIA scenarios with the lowest information level and the lowest amount of available resources. Here we will start with the lowest resource level and the random FDIA, and then examine the lowest resource level with the targeted attack. Afterward, the resource level will be increased and the process repeated until the highest resource level has been reached. This order of examination will then be repeated for the partial and complete information availability levels.

Minimal Available Information

In the scenario of minimal available information, we assume that the attacker has access only to the measurement information of the buses they control. In a single bus scenario, this entails only their own bus' information, while in an all buses scenario, they have all measurement data available. In the section 4.2 we will assume that they have access to the impact of their FDIA in the form of a percentage difference between the false and the unaltered state estimation for ease of implementation.

One bus

In a minimal information scenario with one bus available to inject false data, the attacker is comparable to a regular grid participant. Since the only information available to them is the measurement data from their bus and the impact of an FDIA on it, the possibilities for making a large impact with the attack are very limited. The attacker can only see the effect of their attack on their system, except in the case of a successful attack leading to a destabilization, which makes it difficult to construct an effective attack vector. To achieve an effective vector, the attacker could try to analyze this impact using e.g. machine learning algorithms. Since the amount of unknown variables is very large in this scenario, however, and the database for the training of the model is minimal, the performance of this machine learning model will probably not be ideal.

Fraction of buses

When the attacker can manipulate a fraction of the buses, they not only have access to more variables they can manipulate in the attack vector, but they also have a larger base of information about the system available to them, since they can access the measurement data of all available buses. This makes it easier to use e.g. machine learning models to infer the effect of the attack vector variables on the grid state. In the random attack, the attacker would try to create the greatest overall impact with their attack vector, without focusing on one or more specific buses.

All buses

In a minimal information case where the attacker has access to all of the buses in the grid, the attacker can access all bus measurements and impacts of FDIA on all buses, which provides them with a large database to train their attacking models on. Since they not only have access to all the measurement and impact data but can also manipulate measurements at any bus in the system and all possible combinations thereof, this gives them close to ideal conditions to destabilize the system. In a random FDIA, the objective typically involves maximizing disruption across state variables, potentially leading to overloads on lines or transformers. Overloading these components could lead to significant destabilization by potentially isolating all buses situated behind the cable box or transformer. Such an overload would effectively disconnect these buses from the grid. While the impact of an attack from an attacker with control over all buses in the grid could be devastating to the grid stability, the likeliness of an attacker gaining control over all buses, including or excluding the ones controlled by the grid operator, is very low.

Partial Available Information

As previously mentioned, we define partial available information about the grid to include all the measurement data of every bus and the results of the state estimation. While the attacker would not have access to information on the Jacobian matrix, we also assume that the attacker knows about the results the state estimation would have had, had the FDIA not happened. Emsalifalak et al. (2018) and Kim et al. (2015) suggest methods to infer the topological information and $\mathbf{H}$ from measurement data, which make this assumption plausible. This allows us to create more effective attack vectors in comparison to the minimal information scenario. The available information could be gathered by the attacker through a cyber attack on the grid operator's measurement accumulator or through other means.

One bus

With only one bus available to the attacker, in this scenario we have to differentiate between two sub-scenarios: Firstly the one where the attacker has one fixed bus available to them, e.g. their home bus, or secondly the one where the attacker can manage to gain access to one bus, and can decide which one. While the first sub-scenario is more likely, the second one would offer the possibility of finding the most effective bus for manipulation through an iterative process. This possibility will be explored in chapter 4 and can be utilized for finding the most effective bus for overall effect in the random FDIA. In addition to the previously mentioned machine learning approaches, the techniques proposed by Emsalifalak et al. (2018) and Kim et al. (2015) could lead to the possibility of using more advanced FDIA techniques by inferring topology information. In the simulation, we will specifically test this scenario with a non-targeted attack and the following two techniques for finding an attack vector:

1. **Random Data FDIA** With this FDIA technique, the attack injects randomly generated data as measurements of the available buses. While this may seem counter-intuitive, it will not only be used as a baseline to compare the effectiveness of other attacks but can be utilized to build a database that can be used for training machine learning models or using different inference techniques like singular value decomposition (Viberg and Ottersten, 1991) or sparse optimization (Liu et al., 2014). Since the attack vector values are randomly generated, this attack technique does not offer the possibility of targeting it to specific buses.

   Since the generator for the values of the attack vector has to be provided with upper and lower limits for the respective values as to pass the bad data detection, an iterative process was carried out to ascertain the ideal boundary values. The resulting random data space was then used to create the attack vectors for building the training dataset. The detailed process and the resulting limits will be described in detail in section 4.2.

2. **Machine Learning / Genetic Algorithm FDIA** Based on a dataset constructed via the repeated utilization of the random data FDIA technique, machine learning models are developed to create effective attack vectors for destabilizing the grid through FDIA. These models can be designed to target either the entire grid in a random FDIA scenario or specific buses in a targeted scenario. To ensure the attack vectors evade detection systems, the boundaries developed for the random data FDIA are incorporated into the model training process. We create a combination of a machine learning model with an artificial neural network (ANN) and a genetic algorithm (GA) to predict attack vectors based on previous measurement data. We call this technique machine learning and genetic algorithm FDIA, or in short MLGA-FDIA.

   Before training the model, the process of data preprocessing involves selecting the input and output dataset columns and scaling the features to enhance model performance and stability (Sharma, 2022), followed by splitting the data into training and test sets, with 20% reserved for testing. The ANN, built with TensorFlow Keras, features two hidden layers optimized for handling nonlinear data relationships and adapts its learning during training for effective results (Deru and Ndiaye, 2020).

   The training process runs over multiple epochs and uses a validation split to improve model generalization, ultimately testing the model's performance on separate data and returning the trained model and bounds for further applications.

Algorithm 1 Basic workflow of a genetic algorithm from Affenzeller et al. (2009b)

```
1: Produce an initial population of individuals
2: Evaluate the fitness of all individuals
3: while termination condition not met do
4:     Select fitter individuals for reproduction
5:     Produce new individuals (crossover and mutation)
6:     Evaluate fitness of new individuals
7:     Generate a new population by inserting some new "good" individuals and by erasing some
       old "bad" individuals
8: end while
```

Following model training, a GA (Affenzeller et al., 2009a) is employed to optimize FDIA settings, using the model to identify inputs that maximize attack impacts within specified bounds. The GA, which is described in algorithm 1, evolves potential solutions through techniques modeled after evolution, namely selection, crossover, and mutation operations while ensuring all solutions remain valid through the usage of a repair function based on the boundaries, which erases individuals that predict solutions outside the boundaries. The optimal solution identified by the genetic algorithm serves as the final attack vector, producing predictions on its potential impact. The implementation of this technique will be featured in section 4.2.

Fraction of buses

Since the information level is the most influential factor in the feasibility of FDIA techniques, the available techniques for the creation of an attack vector remain the same as in the scenario with one bus. Analogous to the single bus scenario described previously, both the random data and machine learning / genetic algorithm FDIA can be utilized. As described previously, there are several other attack techniques proposed in the literature, however, we will concentrate on the two previously described techniques.

The random data FDIA and the machine learning FDIA work in the same manner as in the single bus scenario in this scenario. However, more variables can be influenced in the attack vector, which makes it possible to create more effective attack vectors with both techniques. In a scenario where the attacker has access to a fixed number of buses of their selection, the attacker can use the dataset from the random data FDIA to find the most influential combination of buses for either creating the greatest overall effect of the FDIA. This process will be explored in section 4.3.

All buses

When the attacker has access to all measurement data and can manipulate the measurement data at all buses, they can again use this data to train models or use other inference techniques to calculate the remaining grid information such as topology and Jacobian matrix. Although connected to the computational effort, this would make this scenario almost equivalent to the complete information scenario, which gives the attacker total control over the system within the bounds of bad data detection. Since the most widespread bad data detection technique, however, is based on the χ^2-test, which can be tricked when all the measurement data is under the control of the attacker, it leads to the conclusion, that an attacker with full information and control over measurement data has full control over the grid. Though the testing of this scenario and the implementation of an attack which creates normally distributed attack vector data could offer valuable insights for consideration in bad data detection, it is out of scope for this work because of its unlikeliness and will be left for future works to explore.

Complete Information Available

With full information available to the attacker, they can not only use the previously mentioned techniques, which all fall under the category of generalized FDIA as designated by Liu et al. (2011), but also utilize the non-generalized mathematic approaches to creating an attack vector. In the appendix of Liu et al. (2011), the authors illustrate procedures for generating attack vectors using elementary operations, which are particularly relevant in Scenario I of random false data injection attacks.

Random Non-Generalized FDIA According to Liu et al. (2011)

To calculate an attack vector in a random non-generalized False Data Injection Attack (FDIA), the following equation must be satisfied:

$$\mathbf{a} = \mathbf{Hc} \iff \mathbf{Ba} = 0, \quad \text{where} \quad \mathbf{B} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T - \mathbf{I} \tag{2.33}$$

Here, $\mathbf{a}$ is the attack vector, $\mathbf{H}$ is the Jacobian matrix, and $\mathbf{c}$ is a vector of arbitrary values.

When the number of measurements m is greater than the rank n of the Jacobian matrix $\mathbf{H}$ by more than the number of compromised meter readings k (i.e., $k > m - n$), multiple solutions for $\mathbf{Ba} = 0$ exist. The attacker can use elementary column operations to construct the attack vector. The steps are as follows:

1. **Define the Set of Uncompromised Meters**:

$$\overline{I}_{\text{meter}} = \{j \mid 1 \leq j \leq m, j \notin I_{\text{meter}}\}$$

   where I_{meter} is the set of indices of compromised meters.

2. **Locate a Suitable Column Vector**: For a random index $j \in \overline{I}_{\text{meter}}$, find a column vector $\boldsymbol{h}$ in $\mathbf{H}$ such that the j-th element is non-zero. Swap this column with $\boldsymbol{h}_1$ in $\mathbf{H}$.

3. **Construct a Reduced Matrix**: Construct an $m \times (n-1)$ matrix $\boldsymbol{H}_1 = (\boldsymbol{h}_{11}, \ldots, \boldsymbol{h}_{1,n-1})$ by zeroing out the j-th element in all column vectors through elementary operations:

$$\boldsymbol{h}_{1i} = \begin{cases} \boldsymbol{h}_{1i} - \frac{h_{j,1}}{h_{j,i+1}}\boldsymbol{h}_{i+1}, & \text{if } h_{j,i+1} \neq 0 \\ \boldsymbol{h}_{i+1}, & \text{if } h_{j,i+1} = 0 \end{cases}, \quad 1 \leq i \leq n-1. \tag{2.34}$$

4. **Repeat the Reduction Process**: Repeat this reduction process on the resulting $\boldsymbol{H}_1$ and subsequent reduced matrices, using different elements in $\overline{I}_{\text{meter}}$, until all such indices are exhausted.
5. **Construct the Attack Vector**: The attacker arrives at a reduced matrix containing at least one column vector (since $m - k \leq n - 1$), where each column vector is a linear combination of the original columns of $\boldsymbol{H}$, with all j-th rows (indices in $\overline{I}_{\text{meter}}$) consisting of zeros. Any column vector from the final reduced matrix can be used as an attack vector.

Random Generalized Attacks According to Liu et al. (2011)

Generalized FDIA extends traditional attacks by exploiting small measurement errors tolerated by state estimation algorithms. The attack vector $\mathbf{a}$ must satisfy $\|\mathbf{a} - \mathbf{Hc}\| \leq \tau_a$ to bypass detection, where τ_a is the detection threshold.

Let $I_{\text{meter}} = \{i_1, \ldots, i_k\}$ be the set of indices of the k meters that can be compromised. The attacker constructs $\mathbf{a}$ such that $\mathbf{a} - \mathbf{Hc} = \mathbf{t}$ and $\|\mathbf{t}\| \leq \tau_a$. The attack vector $\mathbf{a}$ is determined by solving $\mathbf{B}'\mathbf{a}' = \mathbf{Bt}$, where $\mathbf{B} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T - \mathbf{I}$.

Steps for Random Generalized Attacks.

1. **Define the Error Vector** $\mathbf{t}$: Choose $\mathbf{t}$ such that $\|\mathbf{t}\| \leq \tau_a$.
2. **Transform the Equation**: Convert $\mathbf{a} - \mathbf{Hc} = \mathbf{t}$ into:

$$\mathbf{a} - \mathbf{t} = \mathbf{Hc} \iff \mathbf{B}(\mathbf{a} - \mathbf{t}) = 0 \iff \mathbf{Ba} = \mathbf{Bt}$$

3. **Solve for** $\mathbf{a}$: Represent $\mathbf{a}$ as $\mathbf{a} = (0, \ldots, 0, a_{i_1}, 0, \ldots, 0, a_{i_2}, 0, \ldots, 0, a_{i_k}, 0, \ldots, 0)^T$, where $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ are the unknown variables. Let $\mathbf{B}' = (\mathbf{b}_{i_1}, \ldots, \mathbf{b}_{i_k})$ and solve:

$$\mathbf{B}'\mathbf{a}' = \mathbf{Bt}$$

   If $\mathbf{B}'$ is rank-deficient, there are infinite solutions for $\mathbf{a}'$. Otherwise, if $\mathbf{B}'$ is full rank, solve for $\mathbf{a}'$ directly.

Both the generalized and the non-generalized techniques as outlined by Liu et al. (2011) will be implemented in the simulation and evaluated with the developed metrics.

Impact and Implications

The implications of false data injection attacks are significant, potentially leading to erroneous control actions based on manipulated state estimates. Such vulnerabilities necessitate enhanced detection and prevention strategies beyond conventional methods, through robust optimization techniques and real-time anomaly detection algorithms.

In this work, we concentrate on the effects of non-targeted FDIA techniques to assess the validity of the created power grid and communication network co-simulation as well as the developed metrics for FDIA effectiveness. Hence, the implementation for the simulation will feature the random data FDIA as a baseline, the MLGA-FDIA specifically developed for this study, as well as the random non-generalized and random generalized FDIA techniques proposed by Liu et al. (2011).

There is extensive literature (e.g. Reda et al. 2022), on the impacts of FDIA on power grids, which we will try to further in this work. We will discuss measures for the effectiveness of FDIA and the resulting impact on the grid in section 4.1 and discuss the implementation and effect of the selected FDIA techniques in sections 4.2 and 4.3.

2.5 Cosimulation of power grid and power grid communication

To integrate simulations of the power grid and communication networks, a co-simulation approach is essential. Co-simulation refers to the simultaneous execution of multiple simulation models that represent different but interacting systems, allowing for a comprehensive analysis of their interdependencies (Gomes et al., 2019). This methodology is particularly beneficial in fields like smart grids, where the interplay between electrical networks and communication infrastructures is critical for effective operation and management. Through co-simulation, we can better understand how changes in one domain - like an FDIA in the communication domain - affect overall system performance, leading to more robust and informed decision-making.

2.5.1 Literature Review

The area of co-simulation for the power grid and its communication represents an active research field, with numerous studies addressing the topic, which will be examined in this section. We will evaluate the objectives and methodologies of existing co-simulations to derive insights for our co-simulation of the power grid and its communication systems, as well as to identify simulation tools that may apply to our implementation.

Le et al. (2020) propose a co-simulation approach utilizing the MOSAIK framework in combination with Pypower for testing agent-based distributed algorithms related to electrical network calculations. While agent-based systems offer an interesting approach to simulating distributed networks and user interaction, the implementation work required for such an approach exceeds the scope of this work. Although their focus differs from ours, the technologies discussed could be applicable for use in our implementation. Especially MOSAIK (Schütte et al., 2011), which was specifically designed for smart grid co-simulations, will be considered a viable co-simulation tool for our implementation.

Fernandes et al. (2024) explore distributed control in a multi-agent environment, offering insights into co-simulation methodologies similar to those of Le et al., albeit with a slightly different focus on DC bus voltage control.

Hauser et al. (2005) offer a comprehensive description of their co-simulation process, highlighting an MQTT-like method for data transfer between simulations. While their study is relatively old for the fast-developing area of smart grid co-simulations, it presents critical insights into the communication challenges faced in co-simulation environments. The approach of using an MQTT-like data transfer method between simulations also appears to be an easy-to-implement solution for simulations that are harder to make compatible with one another, e.g. simulations written in different programming languages.

Kazmi et al. (2016) introduce a flexible co-simulation environment using OMNeT++ and PowerFactory, examining the impacts of communication delays and failures on the performance of smart grid systems. Their findings underscore the importance of considering communication dynamics in co-simulation scenarios. While their scope is different from ours, the simulation tools and the manner of co-simulation they used will be considered for our implementation.

Czekster (2021) conduct a survey analyzing various frameworks for smart grid co-simulation, providing a useful comparison of different technologies and identifying further literature that supports ongoing research in this domain.

Wermann et al. (2016) develop ASTORIA, a framework focusing on attack simulations within smart grids, emphasizing command control attacks. This approach, employing tools such as PyPower, NS-3, and MOSAIK, outlines simulation requirements that are particularly relevant to the objectives of this research, since it is also cyber security driven. The utilization of MOSAIK in this work underlines its applicability in simulating complex interactions within smart grid environments, reaffirming its applicability for our implementation effort. Moreover, the incorporation of NS-3 and PyPower in the ASTORIA framework highlights the potential benefits of using these tools in our implementation.

Mets et al. (2014) provides a detailed step-by-step guide for undertaking co-simulation based on diverse criteria, which will serve as a guideline for designing our co-simulation approach. In their paper, the authors systematically outline their co-simulation design guidelines by addressing essential components such as model integration, communication protocols, simulation tools, and performance evaluation metrics. This structured approach not only facilitates a clearer understanding of the key aspects involved in co-simulation but also allows us to adapt their recommendations to enhance the effectiveness of our approach. Their work complements Li et al. (2014), who discussed planning considerations and challenges associated with smart grid communication co-simulation.

Gougeon et al. (2022) examine the influence of communication technologies in power congestion management within smart grids, utilizing a combination of Pandapower, NS-3, and SimGrid. Their results prompt further examination of SimGrid as a potential alternative to MOSAIK. The simulators used will be considered for the implementation, with NS-3 being utilized again and Pandapower being a further developed version of Pypower (Thurner et al., 2018).

Vrtal et al. (2022) address the increasing interdependencies among urban critical infrastructures and the resulting demands for enhanced analysis and cybersecurity measures. The authors point out that current open-source solutions cannot simulate interconnected data networks and power grids, which are crucial for identifying vulnerabilities and mitigating emerging threats. They develop a framework based on NS-3 for the communication simulation component the MatPower tool for MatLab, for the power grid component. Moreover, their approach to communicating between the two simulators in the form of a designated JSON structure provides a straightforward solution for co-simulation interfacing, which will be explored in the simulation design section.

Further investigations, including those conducted by Pipattanasomporn et al. (2009) and Jain and Bhullar (2022), have concentrated on the foundational aspects of multi-agent communication systems and have assessed network performance utilizing smart meters and TCP/IP protocols, respectively. Given that this research focuses on communication with the SMGW, the insights derived from these studies may prove valuable in identifying effective methods for the integration of SMGWs into our implementation.

While the previously mentioned studies present valuable methodologies and implementations for co-simulating power grids and their communication networks, none fully align with our specific objectives and research questions. We intend to synthesize the findings, methodologies, and tools identified in these works to develop a co-simulation tailored to the scope of this study. The subsequent section will provide a comprehensive overview of the simulation and co-simulation tools discussed and will elucidate the decision-making process guiding the selection of tools for our implementation.

2.5.2 (Co-)Simulation Tools

To decide which simulation software best fits our needs, we take a look at the works described in the previous section. Table 2.3 shows the power grid simulation tool, the communications simulation tool, and, if existent, the co-simulation tool.

Table 2.3: Overview of Simulation Tools

Paper	Power Grid Simulator	Communications Simulator	Co-Simulation Tool
Le et al. (2020)	PyPower		MOSAIK
Fernandes et al. (2024)	PSIM	PSIM	PSIM
Hauser et al. (2005)	GridStat	GridStat	GridStat
Kazmi et al. (2016)	PowerFactory	OMNeT++	
Wermann et al. (2016)	PyPower	NS-3	MOSAIK
Gougeon et al. (2022)	PandaPower	NS-3	SimGrid
Vrtal et al. (2022)	MatPower	NS-3	

Based on the literature review, a multitude of tools has been identified for the simulation of the individual components, as well as for their co-simulation. To arrive at an informed decision regarding the selection of appropriate software, we will consider the following decision-making criteria:

1. **Features**
   This criterion encompasses the specific functionalities and capabilities offered by the software, including compatibility with existing systems and the ability to perform the simulation we want to create effectively.

2. **Availability**
   This aspect pertains to the accessibility of the software, including its licensing models, deployment options, and whether it is open-source or commercial. In this decision process, open-source, freely available software will be preferred over closed-source software.

3. **User Friendliness**
   This factor evaluates the ease with which users can navigate and utilize the software, including the intuitiveness of the interface, quality of documentation, and availability of support resources. Another Factor in this is the programming language used and the compatibility with its counterpart simulators.

In the following, we will list the simulators we obtained from the literature review and judge them based on the aforementioned criteria. We will start with the power grid simulators, continue with the communication simulators, and then finalize with judging the co-simulation tools.

Power Grid Simulators

Table 2.4 shows the grid simulation tools from the literature review and their judgment based on the aforementioned criteria. Because the tools from Fernandes et al. (2024) and Hauser et al. (2005) were specifically developed for their use cases and were not available for testing, they were excluded from the consideration. Along with the criteria we discussed earlier, the table shows qualitative judgments on the available features of the tool, the available documentation, whether it is open-source or not, and lists the programming language or interface to program the simulation.

In the features category, we see that MatPower (2024) and PyPower (2023) did not pass the qualitative check. Both of these tools offer only power flow and optimal power flow calculation. The state estimation we want to conduct in this paper thus would not be possible with the use of these tools. It is of note that PyPower is a Python implementation of MatPower, which is the reason for both of them having the same available features. PandaPower (2018) and PowerFactory (2024) both offer a plethora of features, including the power flow calculation and state estimation necessary in the scope of this work, making them valid choices in the features category.

In the documentation category, the software tools are evaluated based on the availability and comprehensiveness of their documentation. Notably, all tools, except PowerFactory, received favorable evaluations due to their high-quality and readily accessible documentation resources. In the case of PowerFactory, no assessment could be conducted as access to a license was not available, thereby precluding an evaluation of the available documentation.

Both PyPower and PandaPower are open-source projects and are available for free use. MatPower itself is a piece of open-source software, however, MatLab, the software needed to use it, is closed-source and commercial software. PowerFactory is fully closed-source and commercial and thus did not receive a positive evaluation in this category.

The final column describes the interfaces and programming languages required for simulation development. As previously noted, MatPower operates within the MATLAB programming environment. Similarly, both PyPower and PandaPower utilize Python as their programming language. In contrast, PowerFactory predominantly relies on a GUI for user interaction; the extent to which programming languages can be employed for its customization remains unclear due to the lack of access to a software license for evaluation.

Table 2.4: Comparison of Grid Simulation Tools

Simulator	**Features**	**Documentation**	**Open-Source**	**Language/Interface**
MatPower		✓	(✓)	MatLab
PandaPower	✓	✓	✓	Python
PowerFactory	✓	?		GUI
PyPower		✓	✓	Python

As the only software tool that was positively evaluated in all categories, we chose to use PandaPower as our simulation tool and make use of its various features. PandaPower comes with its test cases for grid simulation and can also integrate other test grids. This, however, will be discussed in a later section, where we will decide on what type of grid (e.g. urban, rural, etc.) we want to simulate.

Communication Simulators

We will employ the same criteria for evaluating the communication simulators identified in the literature review as we previously implemented for the evaluation of power grid simulators. Analogously, we will exclude the simulators utilized by (Fernandes et al., 2024) and (Hauser et al., 2005). Given our objective to model the communication pathway from the SMGW to the grid operator, it is essential to simulate the relevant technologies accurately. Consequently, the simulation tools must encompass the following features: IP routing, TCP communication, and TLS encryption. For future studies, the software should also be capable of simulating communication channels such as 5G and Powerline. The evaluation of the remaining categories will follow the same criteria as applied to the power grid simulation tools and will be shown in table 2.5.

Table 2.5: Comparison of Communication Simulation Tools

Simulator	Features	Documentation	Open-Source	Language/Interface
NS-3	✓	✓	✓	C++ & Python
OMNeT++	✓	✓	✓	C++

The two simulation tools, NS-3 and OMNeT++, both offer an extensive array of technology stacks capable of simulating communication networks, including the technology stacks required for our use case. Additionally, both tools support simulations for 5G and PowerLine communication channels, among others. Furthermore, both NS-3 and OMNeT++ are well-documented, with their documentation being regularly updated. Both tools are open source and primarily developed in the programming language C++. However, NS-3 provides the added flexibility of being programmable in Python, which aligns with the programming language used by our power grid simulator, PandaPower. This feature grants NS-3 a slight advantage, leading to its selection as our communications simulator.

Co-Simulation Tools

Finally, we must select a co-simulation tool for integrating the power grid simulator and the communication simulator. While this integration does not mandate a specific tool—solutions such as interfaces or Python scripts could suffice—it may be advantageous to employ a tool specifically designed for such purposes. Applying the same criteria used in our previous evaluations, the desired features for this co-simulation tool include compatibility with both PandaPower and NS-3, support for various simulation modes, and extensibility to accommodate future research needs.

Table 2.6: Comparison of Co-Simulation Tools

Simulator	Features	Documentation	Open-Source	Language/Interface
MOSAIK	✓	✓	✓	C#, Java, MatLab, Python
SimGrid	✓	✓	✓	C, C++, Java, Python

As depicted in table 2.6, SimGrid (Casanova et al., 2014) excels in simulating distributed computing systems with a strong emphasis on network and resource management, MOSAIK (Schütte et al.,

2011) offers a versatile co-simulation framework tailored for integrating diverse simulation models, particularly in energy systems. Both are open source and well-documented, with SimGrid offering support for C, C++, Java, and Python, and MOSAIK leveraging C#, Java, Matlab, and Python. Both SimGrid and MOSAIK have been effectively utilized alongside Python-based power grid simulators and NS-3, as can be seen from Gougeon et al. (2022) and Wermann et al. (2016), but MOSAIK offers built-in support for PandaPower, which is the primary argument for selecting it over SimGrid.

3 Co-Simulation

To assess the influence of FDIA and other cyberattacks on the power grid, various scenarios have to be accounted for. Since the power grid belongs to the critical infrastructure and components could be damaged or destroyed during a test of those scenarios, an evaluation of potential attacks on real hardware would be costly. For this reason, simulation tools for the power grid have been developed, which can be used to create an FDIA testbed as described in the first chapter of this work.

This chapter will describe the requirements for the simulation and the overall simulation design for the power grid and the communication network, as well as how to combine these two simulation components into a co-simulation framework. Finally, we will delve into the testing and evaluation process of the simulation according to ISO/IEC/IEEE 29119 testing standards and the developed requirements.

3.1 Simulation Scope and Requirements

To create a simulation software that can be used to simulate a power grid with its communication infrastructure to accurately assess the impact of FDIA, we first need to address what aspects of the power grid and the communication network are relevant for simulating the impact of FDIA and what the requirements to the simulation itself are. We will start by addressing the requirements for the power grid, then take on the requirements for the communication, and conclude with the requirements for the simulation software itself. The assessment of the metrics for the impact of FDIA will be described in section 4.1.

3.1.1 Requirements of the power grid

To decide which aspects of the power grid have to be simulated, we first have to look at which components an FDIA could influence in the specific context of our research. Since we concentrate on the LV grid and set our system boundary at the transformer to the medium voltage grid, we do not necessarily need to simulate the transformer, although it could well be influenced by the FDIA. We will consider the transformer station as a part of the grid, but not simulate the transition of an FDIA from the LV side of the transformer to the medium voltage level. Moreover, since the system boundary we set on the lower end is the SMGW of the households, we will not simulate the impact on HEMS or smart home applications. This consideration will be left to future studies. All of the aspects in between however have to be considered candidates for being considered in the simulation.

Because we want to focus on the single household prevalent in the LV grid, specifically in the German power distribution system, we will begin with the requirements of the metering hardware present in the households. For this, we will assume that a majority of households are equipped with a state-of-the-art SMGW as specified in (BSI, 2021), although the current rollout data does not yet support this assumption (FfE, 2023). The requirements can be extracted from this specification and its TAC. In the scope of this work, we will concentrate on 7 and 14 - "High-frequency measurement provision for value-added services" - which provide measurements in a frequency of 15-minute and 1-minute intervals respectively. The SimBench load data profiles only provide time steps of 15 minutes, but the 1-minute interval can be simulated by using the state estimation results from the previous time step. Both of these simulation scenarios will be explored in the FDIA evaluation in section 4.3. The TAC 7 or 14 measurement data can be used to conduct state estimation calculations, which is necessary for the operation of a smart grid. The requirements that apply to the TAC 14 and are relevant to the simulation are, according to the technical guideline (BSI, 2021):

R1: The SMGW MUST periodically or immediately upon receipt in the gateway transmit selected values to authorized recipients. Authorized recipients are generally all market participants (e.g., suppliers, direct marketers, and aggregators) who have the necessary consent from the connected user for the respective data collection.

R2: The SMGW MUST support the following triggering events for the provision of the specified measurement values:

- ▷ Periodic sending,
- ▷ Ad-hoc sending upon the receipt of a new measurement value,
- ▷ Sending when a measurement value exceeds a certain threshold,
- ▷ Sending when a measurement value falls below a certain threshold.

R4: If a sending period has been parameterized, the SMGW MUST regularly transmit the measurement values recorded since the last transmission, considering the parameterized sending period. The SMGW MUST support sending periods of 60 seconds. The SMGW MAY support sending periods shorter or longer than 60 seconds. The sending period determines how many measurement values are transmitted together.

R5: If thresholds have been parameterized, the SMGW MUST transmit the measurement values recorded since the last transmission when a threshold is exceeded or fallen below. It MUST be identifiable which of the transmitted measurement values are responsible for triggering the measurement value transmission.

R6: If no sending period and no thresholds have been parameterized, the SMGW MUST send each measurement value immediately upon its registration in the SMGW.

R7: For the recording of measurement values, the SMGW MUST support a sampling rate of 60 seconds for the meter. The SMGW MAY support smaller or larger sampling rates.

The specification in (BSI, 2019) contains further requirements for the SMGW in TAC14. However, since those are tailored more to aspects such as billing and consumer protection, they are not deemed relevant in the scope of this work. The following data is made available according to the specification:

Measurement Data:

- ▷ **Active power (P)**

- ▷ **Reactive power (Q)**
- ▷ Apparent power (S)
- ▷ Power factor (cos ϕ)
- ▷ **Voltage (U)**
- ▷ Current (I)
- ▷ Frequency (f)
- ▷ Energy consumption (E)
- ▷ Maximum Demand (MD)
- ▷ Meter status
- ▷ Event logs

User Information:

- ▷ Consumer ID
- ▷ Contract account number
- ▷ Meter Point Administration Number (MPAN)
- ▷ Aggregator ID
- ▷ Supplier ID
- ▷ Direct marketer ID

When comparing the available measurements with the ones necessary for power flow calculation and state estimation described in subsection 2.3.1, only the measurement data points written in bold font are relevant for the calculations. The remaining measurements and data points from the specification therefore can be either disregarded or replaced with placeholder values in the simulation process. In this comparison however, we notice that some of the necessary data points, especially the line measurements, are still missing. Since these values are not highly variable, they do not need to be simulated, but realistic values should be assumed for the calculations to obtain realistic results (Schäfer, 2023).

To monitor the health of the power grid at any time, it is necessary to monitor some important metrics that could indicate a destabilization of the grid. For this, we take into account the metrics of the maximal power of the transformer and the fuse current of the power lines. Once one of those is exceeded by a certain percentage, power lines, and the buses lying behind them will be cut off from the grid, and in the case of the transformer, the whole LV grid will be cut off from the medium voltage level. For this reason, we add requirements for the monitoring of grid health:

R8: The power level of the transformer node MUST be monitored at all times and the simulation MUST provide a notification in case the power level reaches the level at which the transformer will shut off.

R9: The line current of each power line MUST be monitored at all times and a visual indicator MUST be provided by the simulation to indicate whether the line current has passed 80% of the fuse current

R10: When the line current reaches the fuse current, a visual indicator MUST be provided to show the separation of the line and the sub-grid being cut off from the grid.

R11: When the line current reaches the fuse current, the sub-grid being cut off by the fuse failure SHOULD be excluded from the grid and its calculations.

The requirements for the grid have been summarized in the following table:

Table 3.1: Summary of Requirements to the power grid simulation

Requirement	Short Description	Priority
R1	**SMGW must transmit selected values to authorized recipients periodically or immediately.**	**Must**
R2	SMGW must support events triggering the provision of measurement values.	May
R4	**SMGW must transmit recorded values regularly if a sending period is parameterized (60s period supported).**	**Must**
R5	SMGW must transmit values recorded since the last transmission upon thresholds being exceeded or fallen below.	May
R6	If no periods/thresholds, SMGW must send each value immediately upon registration.	May
R7	**SMGW must support a 60-second sampling rate for measurement values.**	**Must**
R8	**Transformer node power level must be monitored continuously with warnings and notifications.**	**Must**
R9	**Line current must be monitored with visual indicators when reaching the fuse current.**	**Must**
R10	**Provide visual indicator for line separation when line current reaches fuse current.**	**Must**
R11	**Exclude sub-grid cut off by fuse failure from calculations.**	**Should**

The requirements that are considered to apply to our simulation scope have been marked in bold font, whereas the remaining requirements do not apply to the scope, since the registration period of SMGWs will not be modeled and the simulation will be conducted in a discrete-event simulation Sharma (2015), where measurement-triggering events will not be included. The non-applicable requirements have been marked with the priority "may", as they could be implemented in future versions of the simulation.

3.1.2 Requirements of the communication network

To achieve a realistic testbed for FDIA attacks on the power grid, not only do we need to simulate the power grid itself, but also the communication between the components and households of the grid. Since measurement data such as described in the previous section is sent via different communication channels such as 5G, LTE, or Powerline communication (Abrahamsen et al., 2021), we need to either account for all of those possibilities or increase the level of abstraction to the biggest common denominator. We can do this by simulating only the transport layer and upwards of the OSI-model (Day and Zimmermann, 1983). The adaptation and testing of FDIA on specific communication channels is out of the scope of this work.

The technical guideline for the SMGW (BSI, 2021) provides us with a list of requirements for the communication and encrption of the data:

CR1: Secure Data Transmission

- ▷ The SMGW MUST ensure the confidentiality, integrity, and authenticity of data during transmission.
- ▷ Encryption methods MUST adhere to the latest cryptographic standards.
- ▷ Communication channels MUST be protected against unauthorized access and tampering.

CR2: Authentication and Authorization

- ▷ The SMGW MUST support strong authentication mechanisms for both users and devices.
- ▷ Access control policies MUST ensure that only authorized entities can access specific data and services.

CR3: Communication Protocols

- ▷ The SMGW MUST support standardized communication protocols for data exchange (e.g., DLMS/COSEM, IEC 62056).
- ▷ Protocols implemented SHOULD be interoperable with existing smart meter infrastructure.

CR4: Data Exchange Formats

- ▷ The SMGW MUST use standardized data exchange formats (e.g., XML, JSON) for interoperability.
- ▷ Profile data MUST be clearly defined and consistently formatted.

CR5: Reliability and Availability

- ▷ The communication system of the SMGW MUST ensure high reliability and availability.
- ▷ Mechanisms MUST be in place for data redundancy and fault tolerance.

CR6: Time Synchronization

- ▷ The SMGW MUST support time synchronization protocols to ensure accurate time-stamping of data.

- ▷ NTP (Network Time Protocol) or similar standards SHOULD be used to maintain consistent time across all devices.

CR7: Scalability

- ▷ The SMGW communication infrastructure SHOULD be scalable to handle an increasing number of devices and data volumes.
- ▷ Performance SHOULD remain consistent regardless of the scale of deployment.

CR8: Legal and Regulatory Compliance

- ▷ The SMGW MUST comply with national and international legal and regulatory requirements.
- ▷ Data privacy regulations (e.g., GDPR) MUST be adhered to throughout data processing and transmission.

CR9: Firmware Updates and Patching

- ▷ The SMGW MUST support secure firmware updates and patching mechanisms.
- ▷ Updates MUST be verifiable to ensure integrity and authenticity.

CR10: Monitoring and Logging

- ▷ The SMGW MUST provide robust monitoring and logging capabilities.
- ▷ Logs MUST be detailed enough to trace security incidents and system performance issues.

As some of these communication requirements pertain to aims that are not inside the scope of this work, we will concentrate on the requirements CR3, CR4, CR5, and CR7. Requirements CR1, CR2, CR6, CR8, CR9 Yand CR10 may be added in later works, but, as discussed earlier, this work does not differentiate between the different communication channels and mediums that could be used for the measurement communication or the encryption of the data. Furthermore, the setup, maintenance, and authentication process of the SMGW is not taken into account. An overview of the requirements can be found in the following table, with the requirements in focus being displayed in bold font. The priority of the excluded requirements has been updated to "May" in the table.

To achieve compliance with requirement CR4, we use JSON-formatted measurement messages with the data points described in the previous section, and send these messages in the form of TCP packets for compliance with requirement CR3. An example JSON message with TAC 14 measurement data can be found in figure 3.1.

Table 3.2: Summary of Communication Requirements

Requirement	**Short Description**	**Priority**
CR1	Ensure confidentiality, integrity, and authenticity of data transmission using the latest encryption standards.	May
CR2	Support strong authentication and access control mechanisms for users and devices.	May
CR3	**Support standardized communication protocols and ensure interoperability with smart meter infrastructure.**	**Must**
CR4	**Use standardized data exchange formats and clearly defined profile data.**	**Must**
CR5	**Ensure high reliability, availability, data redundancy, and fault tolerance in communication.**	**Must**
CR6	Support time synchronization protocols like NTP for accurate time-stamping.	May
CR7	**Ensure communication infrastructure scalability and consistent performance.**	**Should**
CR8	Comply with legal regulations, including data privacy requirements like GDPR.	May
CR9	Support secure firmware updates and verifiable patching mechanisms.	May
CR10	Provide detailed monitoring and logging for tracing security incidents and performance issues.	May

```
{
  "MeasurementData": {
    "ActivePower": 123.45,
    "ReactivePower": 67.89,
    "ApparentPower": 130.00,
    "PowerFactor": 0.95,
    "Voltage": 230,
    "Current": 5.4,
    "Frequency": 50,
    "EnergyConsumption": 1500.67,
    "MaximumDemand": 120,
    "MeterStatus": "ok",
    "EventLogs": [
      "Event1: Power Failure at 03:00",
      "Event2: Power Restoration at 03:10"
    ]
  },
  "UserInformation": {
    "ConsumerID": "C123456",
    "ContractAccountNumber": "CA7891011",
    "MeterPointAdministrationNumber": "MPAN987654",
    "AggregatorID": "AG87654321",
    "SupplierID": "SP12345678",
    "DirectMarketerID": "DM12345678"
  }
}
```

Figure 3.1: Example JSON Message for TAC14

3.2 Simulation Design

In this section, we will detail the design of the power grid and communication simulation as well as how they are connected. We will go into detail about the data flow and handling as well as the software architecture and provide code snippets for better explanation.

3.2.1 Power Grid Simulation

As discussed in the preceding sections, our power grid simulation will utilize the PandaPower tool, which combines the functionalities of PYPOWER with the Pandas library, which provides DataFrames to store information, which is comparable to CSV files or spreadsheets and facilitate data handling. PandaPower stores all information about the power grid in the `net` variable, with sub--DataFrames for power flow and state estimation values, among others. Furthermore, PandaPower offers a comprehensive suite of functions tailored for our analysis, including power flow calculations, state estimation, the integration of measurement data, and bad data detection using a χ^2 test. Moreover, PandaPower provides robust visualization tools that will facilitate the effective presentation of our simulation results (Thurner et al., 2018).

For the grid and load profile data, we employ datasets from the SimBench project, specifically designed to represent the characteristics of the German electricity market (Meinecke et al., 2019). The SimBench grids and load profiles can be easily imported since their formatting is supported by PandaPower. In the simulation and evaluation of the FDIA attacks, we utilize an LV semiurban grid and its load profile data.

As illustrated in Figure 3.2, the power grid simulation is executed in an iterative loop that continues as long as load data remains available in the dataset or for a predefined number of iterations. This process is broadly divided into two phases: phase 1 involves the generation of measurement data, while phase 2 encompasses the state estimation process.

Several considerations are crucial when executing the simulation, particularly concerning the modification of measurement values and the number of available measurements in the TAC 14 scenario. In reality, not every household is equipped with an SMGW, implying that certain measurements may be unavailable and should be excluded from the dataset. Conversely, state estimation accuracy improves with increased data availability, and below a certain threshold of measurement capability, namely when $m < 2 \cdot n - 1$, where m is the number of measurements and n is the number of nodes in the grid, the grid state cannot be calculated (Schäfer, 2023). Therefore, instead of entirely omitting unavailable measurements, one can substitute some data points with historical measurements from load profiles which are transformed to pseudo measurements as described in Baran and Kelley (1994). However, since PandaPower provides no function to add weights to the measurements, as is done by Baran and Kelley to distinguish between the real and pseudo measurements, we will rely on the time series data provided by the SimBench dataset instead. Furthermore, it is improbable that the measurement infrastructure of households will yield values identical to those calculated by the power flow analysis, necessitating adjustment, such as noise or a randomly generated factor to truly reflect realistic measurement data. However, since we assume the effect of implementing such adjustments would have minimal effect on the performance and outcome of the simulation, the data points will be used as is.

In the following, the simulation process will be described in broad terms, before the components will be individually analyzed and their implementation illustrated along pseudo-code examples.

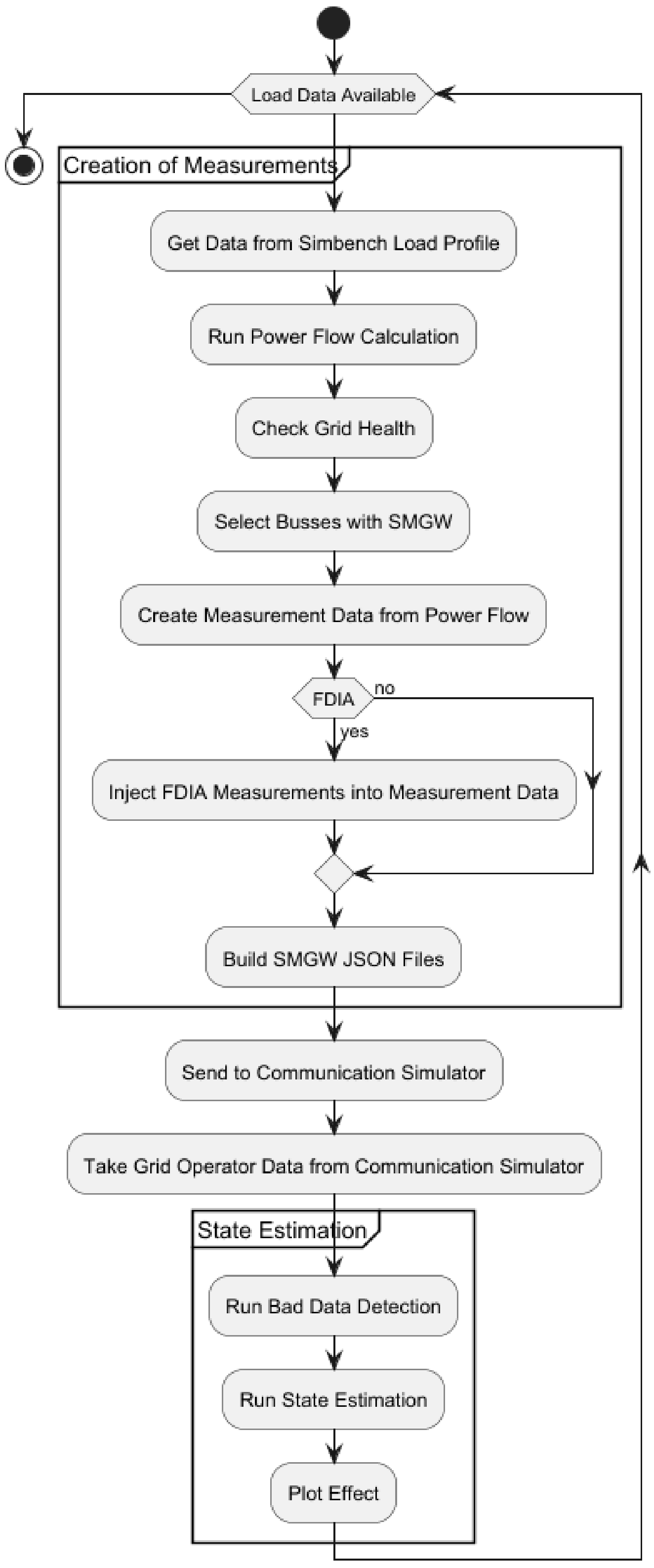
Load Data Available
Creation of Measurements
Get Data from Simbench Load Profile
Run Power Flow Calculation
Check Grid Health
Select Busses with SMGW
Create Measurement Data from Power Flow
FDIA
no
yes
Inject FDIA Measurements into Measurement Data
Build SMGW JSON Files
Send to Communication Simulator
Take Grid Operator Data from Communication Simulator
State Estimation
Run Bad Data Detection
Run State Estimation
Plot Effect

Figure 3.2: Power Grid Simulation Activity Diagram

In phase 1, each iteration begins by extracting the current time step values from the SimBench load profile. This load profile data is then employed to execute a power flow analysis of the grid, as detailed in section 2.3.1. The power flow calculation takes the load profile data of active and reactive powers and supplies the respective voltages and voltage angles, which can be used as measurement data for the buses. Subsequently, if an FDIA occurs during the simulation cycle, the corrupted data is integrated with the standard measurements, replacing the original data points. In the following step, both the genuine and false measurement data are formatted into the JSON TAC 14 schema, as illustrated in figure 3.1.

The formatted JSON data is subsequently input into the network simulator, which models the communication processes and transmits the data as received by the grid operator back to the power grid simulation.

In Phase 2, the measurements received by the grid operator undergo bad data detection using the χ^2 test before serving as input for the state estimation phase, which marks the end of the iteration. Here, we will assess the impact of FDIA by comparing the state estimation outcome with the state estimation outcome without manipulation. The state estimation results are then visualized using the plotting functions, which will be described in section 4.2.

Get Data from SimBench load profile

The SimBench load profiles provide load profiles in 15-minute intervals for the whole year of 2016, which was a leap year, amounting to 35136 total load profiles.

For each time step, the network model is updated with the data points necessary to conduct the power flow using the `appl_absolute_values` function, which transfers the load profile measurements to the `net`, which is the variable containing all of the grid information, as previously discussed. Although the script can be utilized to run with all 35136 time steps for a complete yearly simulation, the simulation is usually run with the first 96 time steps to simulate the power grid for one day.

Algorithm 2 Getting load profiles from SimBench dataset

```
Require: SimBench code for specific grid network, i.e. "1-LV-semiurb4–0-sw"
1: Set sb_code to the desired SimBench code.
2: Retrieve network data: net $\leftarrow$ get_simbench_net(sb_code)
3: Fetch load profiles: profiles $\leftarrow$ get_absolute_values(net)
4: for each time step i do
5:     Apply load profile to net
6: end for
```

Run Power Flow Calculation

The PandaPower library offers a simple function to run the power flow calculation of the grid. When the values from the load profile have been added to the `net` variable, the power flow can be run by simply using the following command:

Algorithm 3 Run Power Flow Calculation

```
Require: net
1: pandapower.runpp(net, calculate_voltage_angles=True, init="results")
```

This function runs a balanced AC power flow calculation based on the PYPOWER library. The Option calculate_voltage_angles decides whether the voltage angles should be computed during the power flow calculation, which is enabled for our simulation purposes. From the second iteration on, the init input is set to "results", which allows the power flow calculation to use the previous iteration's results as a starting point to increase performance (Schäfer, 2023).

Create Measurement Data

After the power flow calculation, the measurement data is created based on the power flow results. PandaPower offers a Pandas DataFrame containing the results, from which the resulting voltage, active power, and reactive power for each bus are taken to be used as measurements in the simulation. As discussed previously, the measurements could be subjected to noise or other adjustments at this point. The data points of each bus are then collected and a JSON object as described in figure 3.1 is created, where the values for bus, voltage, active power, and reactive power are replaced with the respective values from the power flow calculation. All of these JSON objects are then stored in the form of a list, which is then returned by the create_measurement function. The amount option can be used to create measurement data for only a certain percentage of randomly selected nodes in the system.

Algorithm 4 Measurement Creation for Grid Model

```
1: procedure CREATE_MEASUREMENT(net, amount)
2:     Initialize an empty list measurements
3:     for each bus in net.res_bus.index do          ▷ DataFrame with Power Flow Results
4:         if a random number is less than amount then
5:             Create a measurement dictionary measurement with:
     ▪ MeasurementData containing:
       ▷ Voltage set to the bus voltage measurement
       ▷ ActivePower set to the bus active power measurement
       ▷ ReactivePower set to the bus reactive power measurement
       ▷ Other attributes (e.g., ApparentPower, Current) marked as “nA” (not available)
     ▪ UserInformation specifying:
       ▷ ConsumerID, ContractAccountNumber, etc.
6:             Append measurement to measurements
7:         end if
8:     end for
9:     return measurements
10: end procedure
```

Conduct FDIA

The injection of the FDIA is integrated between the creation of measurement data and the sending of the measurement data to the communication simulator. We assume that the attacker has control over the available nodes' SMGWs and can thus inject the data before they are sent to the grid operator. Other attack surfaces like intercepting and manipulating the TCP packets with which the measurement data are sent are thinkable, but the implementation of such an attack is very complicated, which is why we will not attempt it in the scope of this work.

In the simulation, four of the previously discussed FDIA techniques will be implemented and can be selected when running the simulation. Furthermore, there is the option to simulate without any FDIA. The following table shows an overview of the available attack techniques and the necessary inputs to create an attack vector.

Table 3.3: Overview of available FDIA functions

Name	**Description**	**Additional Inputs**
random_data_fdia	Injects random data at buses	`bounds`
random_fdia_liu	Column operation attack vector	`net`, H
random_generalized_liu	Column operation attack vector	`net`, H, τ
machine_learning_genetic_alg	Machine Learning Model & Genetic Algorithm	`bounds`, `dataset`

All of the FDIA functions require a list of buses and a list of measurements as input variables. The return value of all FDIA functions is the list of measurements with the attack vector injected into the measurement data. The detailed design of the FDIA functions will be discussed in chapter 4.

Send to Communication Simulator

After an FDIA algorithm or no FDIA algorithm has been selected and run on the measurement data list, the measurements are sent to the communication simulator. First, a message with the amount of measurements is sent to port 8081, where the communication simulator is listening. Because of the aim to keep this process as close to reality as possible, a TCP port is then opened for every bus on the communication simulator's side and the individual JSON measurement object is sent to the specific port for the respective bus in the form of real TCP packets. Since the communication between the simulators is based on TCP communication, the two simulators can either be run on the same machine or two separate machines, as long as they can establish a TCP connection between each other. The separation of ports between the nodes furthermore helps with the race condition (Netzer and Miller, 1992) of parsing the received messages on the communication simulator's side. Furthermore, the JSON objects from the measurement list are saved as JSON files for later inspection with a file name corresponding to the time step and bus number.

Algorithm 5 Send Measurement Data to Network Simulator

```
1: procedure SEND_TO_NETWORK_SIM(SMGW_data, timestep)
2:     for (i, measurement) in enumerate(SMGW_data) do
3:         file ← open("./JSON/measurement_timestep_i.json", "w")
4:         json.dump(measurement, file)
5:         file.close()
                                   ▷ Send the measurement data file to the network simulator
6:         send_message("127.0.0.1", 8081 + i, json.dumps(measurement))
7:     end for
8: end procedure
```

Take Grid Operator Data from Communication Simulator

Since all the measurement data from the different buses has been congregated by the grid operator during the communication simulation, the communication simulations return the measurement data as a list of measurement JSON objects, similar - or rather identical - to the list of measurement data that was returned from the FDIA function. The simulation waits for a TCP message on port 8080 until the list of measurements is sent and received from the communication simulator. The function then simply returns the list of measurements.

Algorithm 6 Receive Measurement from Communication Simulator

```
1: procedure RECEIVE_FROM_NETWORK_SIM
2:     message ← wait_for_message(8080)
3:     return message
4: end procedure
```

Run Bad Data Detection

After the list of measurement data has been returned from the communication simulator, the JSON measurement objects are first parsed and then added as measurements to the `net`, using PandaPower's create_measurement function.

The create_measurement function takes more inputs, namely standard deviation and a spelled-out version of the value it represents, e.g. "voltage", but this has been shortened in the pseudo-code for better readability. After all the measurements from the list have been added to the net, the bad data detection is carried out.

In the bad data detection, both the chi2_analysis function and the remove_bad_data function of PandaPower are utilized. Since remove_bad_data removes bad data but does not return any indicator of whether bad data was removed or not, a try-except structure was implemented that uses the chi2_analysis function to indicate whether bad data was found in the measurements. The function does not return specific information about the removed data, which could be helpful in training machine learning models, but only a boolean value that is true when bad data is found.

Algorithm 7 Parse Measurement Data

```
1: procedure PARSE_MEASUREMENT(measurements, net)  ▷ Parse the measurement data from the network simulator
2:     for measurement in measurements do
3:         voltage ← measurement["MeasurementData"]["Voltage"]
4:         active_power ← measurement["MeasurementData"]["ActivePower"]
5:         reactive_power ← measurement["MeasurementData"]["ReactivePower"]
6:         bus ← measurement["UserInformation"]["ConsumerID"]
7:         if voltage is not None then
8:             pandapower.create_measurement(net, "v", element_type="bus", element=bus)
9:         end if
10:        if active_power is not None then
11:            pandapower.create_measurement(net, "p", element_type="bus", element=bus)
12:        end if
13:        if reactive_power is not None then
14:            pandapower.create_measurement(net, "q", element_type="bus", element=bus)
15:        end if
16:    end for
17: end procedure
```

Algorithm 8 Check for Bad Data in Measurement

```
1: procedure CHECK_BAD_DATA(net)
2:     try bad_data ← pandapower.estimation.chi2_analysis(net, init="slack")
3:     if bad_data then
4:         print("Bad Data Detected")
5:     end if
6:     except (AttributeError)
7:          print("No Bad Data Detected")
8:     pandapower.estimation.remove_bad_data(net, init="slack")
9: end procedure
```

After the indicator for the removal of bad data was printed, the remove_bad_data function is used to remove the bad data from the measurements.

Run State Estimation

After the bad data has been removed from the measurements of the `net` variable, the state estimation is run with PandaPower's estimate function. The function takes a large number of input parameters, which will be shortened in the pseudo-code for readability purposes. Analogous to the power flow calculation, the first iteration uses the slack method as a starting value, whereas from the second iteration, the results of the previous state estimation are utilized to increase performance. Additionally, the calculate_voltage_angles option, shortened in the pseudo-code to "cva", will be set to true to calculate the voltage angles in the system. The remaining options such as τ and the maximum number of iterations will be kept as the standard values.

Algorithm 9 Run State Estimation

```
1: procedure RUN_STATE_ESTIMATION(net)
2:     if timestep = 0 then
3:         try pandapower.estimation.estimate(net, init="slack", cva=True)
4:         except (ValueError)
5:             print("State Estimation failed.")
6:     else
7:         try pandapower.estimation.estimate(net, init="results", cva=True)
8:         except (ValueError)
9:             print("State Estimation failed.")
10:    end if
11: end procedure
```

Similar to the bad data detection, we utilize a try except for structure, since testing has shown that the program would stop in some cases when too many of the measurements were removed by the bad data detection. In this case, a textual indicator of the state estimation failing will be printed out in the console to inform the user.

3.2.2 Communication Simulation

To simulate the communication between SMGWs at individual households and the grid operator, we implement a streamlined communication simulation. This simulation focuses on the TCP/IP interactions between SMGWs and the grid operator, abstracting away the complexities of specific network topologies and communication technologies such as 5G or Powerline Communication and encryption techniques such as TLS.

As illustrated in Figure 3.3, the simulation is executed using the NS-3 simulation tool and can be divided into two phases: setup and simulation. The setup phase begins with the instantiation of multiple network nodes representing SMGWs, with the quantity equating to the total number of power grid buses[1] plus one additional node representing the grid operator. This grid operator node is tasked with collecting TAC 14 data transmitted from the SMGWs.

Each SMGW node is linked to the grid operator node via a point-to-point communication channel. These channels are configured with the InternetStack of the NS-3 simulator, which equips them with IP and TCP functionalities. Each channel between an SMGW node and the grid operator node is allocated a unique subnet and corresponding IP address to ensure distinct and efficient data routing.

[1]Which was sent as a TCP message to port 8081 from the power grid simulator

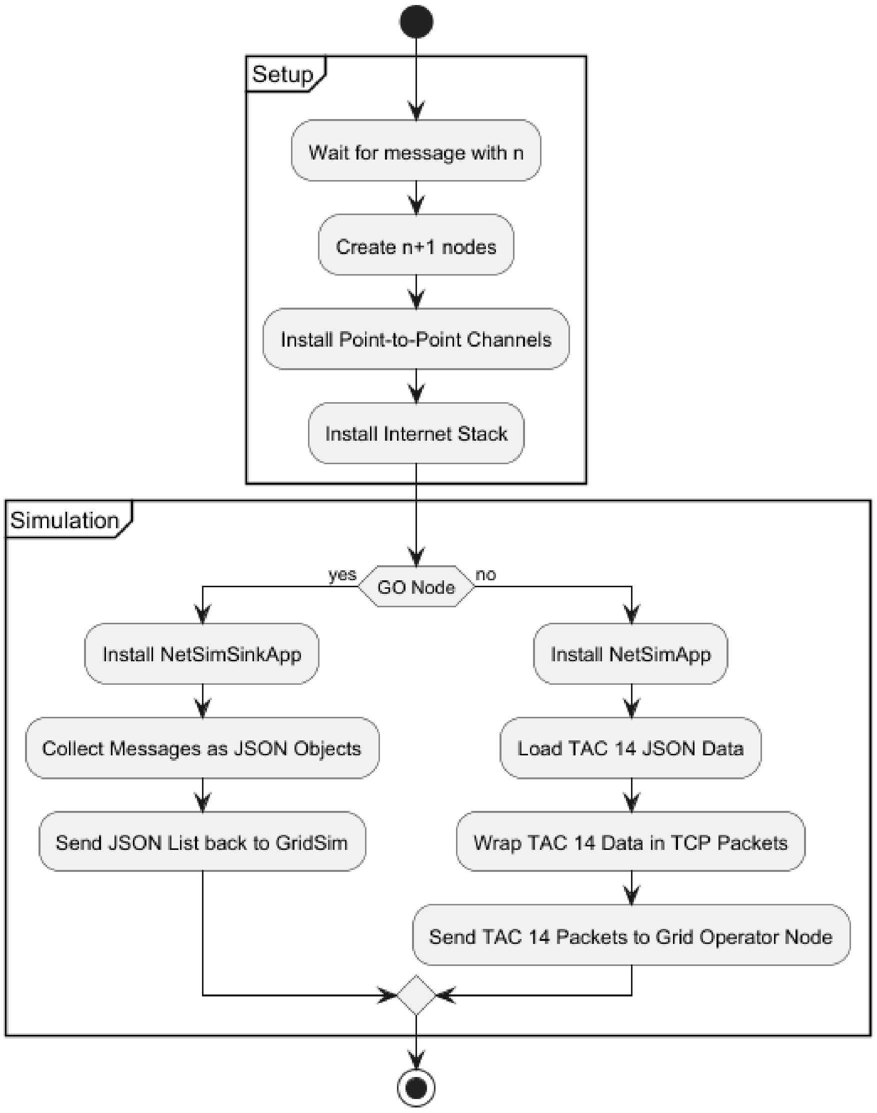
Setup
Wait for message with n
Create n+1 nodes
Install Point-to-Point Channels
Install Internet Stack
Simulation
GO Node
yes
no
Install NetSimSinkApp
Collect Messages as JSON Objects
Send JSON List back to GridSim
Install NetSimApp
Load TAC 14 JSON Data
Wrap TAC 14 Data in TCP Packets
Send TAC 14 Packets to Grid Operator Node

Figure 3.3: Communication Simulator Activity Diagram

To facilitate data reception, a modified PacketSinkApplication is installed on the grid operator node. This application serves as a terminal for incoming TCP packets, effectively functioning as a data sink for all transmitted information from the SMGW nodes.

Conversely, the SMGW nodes are equipped with a customized version of NS-3's OnOffApplication. This application, originally designed for sending contentless TCP or UDP packets, has been modified to transmit TCP packets containing specific contents, namely JSON objects containing TAC 14 data, rather than merely packets of a predetermined size.

In the second phase of this simulation, each SMGW node opens a real TCP socket on a port predetermined by its corresponding bus number. It then waits for data reception from the grid simulation, which will send the TAC 14 JSON object of the bus to this socket. The node then wraps the TAC 14 JSON data into TCP packets and dispatches them to the grid operator node. The grid operator node, acting as an aggregator, systematically collects all TAC 14 JSON data packets, discarding any packets with irrelevant content. This collected dataset is then consolidated into a singular list of JSON objects and is subsequently sent back to the power grid simulation for usage in the state estimation process, as described in the previous section.

In the following, we will discuss the various steps and components of the communication simulation and examine them along with pseudo-code descriptions.

The code begins by including the essential NS-3 modules along with a JSON library called `nlohmann/json` for data manipulation. The primary modules included in the code are as follows:

- **Core Module:** Basic functionalities for NS-3 simulation.
- **Network Module:** Essential for handling network devices and nodes.
- **Internet Module:** Provides necessary internet protocols including TCP.
- **Point-to-Point Module:** Implements point-to-point connections between nodes.
- **Applications Module:** Aids in application development for sending and receiving data packets.
- **PCAP Module:** For packet capture functionality.

The code initializes logging for the component `TcpExample`, allowing for debugging and performance monitoring during the simulation.

Two global functions, `SendPacketTrace` and `ReceivePacketTrace`, are defined to trace the sending and receiving of packets. `SendPacketTrace` logs the size of the sent packet, and `ReceivePacketTrace` Logs the source and destination addresses of received packets and attempts to parse the payload as JSON, saving it to a specified file.

The `simulation_loop` function contains the core logic for packet transmission and reception among a specified number of nodes. It includes the following steps:

1. Data Reception: Handles receiving JSON messages from the grid simulation.
2. Node Creation: Initializes SMGW and grid operator nodes and connects them with a point-to-point link.
3. TCP Socket Setup: Creates TCP sockets on both sender and receiver nodes.
4. Application Setup: Instantiates applications for transmitting packets and receiving them.
5. Event Scheduling: Schedules events to manage sending and receiving operations.
6. Simulation Execution: Runs the simulator and cleans up post-execution.

Algorithm 10 provides a pseudo-code depiction of how the simulation_loop is implemented.

The main function serves as the entry point of the application where it begins listening for an incoming message on port 8081 to get the number of simulation nodes and enters a loop running `simulation_loop` indefinitely.

Algorithm 10 Simulation Loop Function

```
1:  procedure SIMULATION_LOOP(n_nodes)
2:      measurements = receive_messages(n_nodes, 8081)
3:      create nodes (n_nodes + 1)
4:      setup point-to-point link with data rate and delay
5:      install the Internet stack
6:      for each node do
7:          create socket for sending
8:          bind and connect socket
9:      end for
10:     setup packet sink on the receiver side
11:     schedule packet sending
12:     -run simulation
13:     collect and send measurements back to Python
14: end procedure
```

Two key classes, `NetSimApp` and `NetSimSinkApp`, the aforementioned modified versions of OnOffApplication and PacketSinkApplication, manage the sending and receiving of packets. `NetSimApp` is responsible for managing packet generation, sending packets in sequence, and scheduling the next packet transmission, while `NetSimSinkApp` acts as a sink for the packets and congregates them to a list of JSON objects which are eventually sent back to the power grid simulation.

The pseudo-code for the core methods in `NetSimApp` is as follows:

Algorithm 11 NetSimApp Class Methods

```
1:  procedure StartApplication
2:      connect to the peer
3:      begins sending packets
4:  end procedure
5:  procedure StopApplication
6:      cleanup and close socket
7:  end procedure
8:  procedure SendPacket
9:      create and send a packet
10: end procedure
```

To facilitate sending and receiving messages between Python and C++, helper functions are implemented using Boost ASIO. The two helper functions are `wait_for_message`, which listens on a specified port until a message arrives, and `send_message`, which sends a message to a specified IP address and port. A handshake protocol was implemented in these functions so that the sender first sends the length of the incoming message and after a response from the listener the message. The same functions have also been implemented in Python for the power grid simulation as to ensure compatibility.

Additionally, helper functions for parsing JSON objects as well as creating a list of JSON objects from the received TCP packages have been implemented. The pseudocode for these helper functions can be found in algorithm 12:

Algorithm 12 Helper Functions

```
 1: procedure COLLECT_JSONS(jsonData)
 2:     collected_jsons ← new JSON array
 3:     for JSON String in jsonData do
 4:         JSON Object ← Parse JSON String
 5:         Append JSON Object to collected_jsons
 6:     end for
 7:     return collected_jsons
 8: end procedure
 9: procedure RECEIVE_MESSAGES(n_msgs, base_port)
10:     messages ← vector of size n_msgs
11:     tasks ← vector of futures
12:     for i ← 0 to n_msgs - 1 do
13:         task ← async:                                  ▷ Let tasks run in parallel
14:         message ← wait_for_message(base_port + i)      ▷ Listen on multiple ports at once
15:         messages[i] ← message
16:         Append task to tasks
17:     end for
18:     for task in tasks do
19:         Wait until all tasks are finished
20:     end for
21:     return messages
22: end procedure
```

In summary, the communication simulation program waits for meter measurements from the power grid simulation and then simulates the sending process from the SMGWs to the GO via TCP connections using the NS-3 framework, finally passing the congregated data back to the power grid simulation.

3.2.3 Co-simulation

To effectively interface between the two simulators, which are implemented in different programming languages and thus cannot be directly integrated as modules within a single program, a robust control and data transfer mechanism is essential. In Section 2.5.2, we explored the potential application of the co-simulation tool MOSAIK. Nevertheless, given that our data is formatted as JSON files, which can be seamlessly read and processed by the individual simulators, the data transfer capabilities provided by MOSAIK are not required for our specific needs. Despite this, achieving a coherent and synchronized co-simulation necessitates the development of control mechanisms to coordinate the simulation sequence. These mechanisms can be designed in a more streamlined fashion, eliminating the need for an additional simulation tool while still ensuring an efficient and well-structured co-simulation framework.

To effectively organize the co-simulation and ensure that each simulation tool executes the appropriate actions at the correct times, a TCP socket-based communication framework is employed. Each simulation tool establishes a TCP socket, functioning as a listener for incoming packets and facilitating data transfer to the corresponding simulator when required. The sequence diagram presented in Figure 3.4 illustrates the ordered sequence of actions executed by each simulator and elucidates the interface between them.

Initially, the power grid simulator commences its operation by loading the data for the current timestep from the Simbench load profile. This data serves as input for the power flow calculation, which generates the requisite measurement values. Depending on the simulation settings, an FDIA may be performed. The resulting measurement data are formatted as JSON files and transmitted from the power grid simulation's TCP socket to the communication simulation's TCP socket, which is primed to receive incoming messages.

Upon receipt of the measurement data, the simulation of communication from the SMGWs to the grid operator is conducted, as depicted in Figure 3.3. Subsequently, the power grid simulator's TCP socket enters a listening state, awaiting incoming messages. Once the measurement data have been aggregated by the grid operator node, they are forwarded to the communication simulator's TCP socket, which then relays them to the power grid's TCP socket. The power grid simulation resumes by undertaking bad data detection and state estimation.

This iterative process continues until the load profile data is fully processed, ensuring an orderly and comprehensive co-simulation. This approach underscores the systematic coordination and communication between simulators essential for achieving accurate co-simulation outcomes.

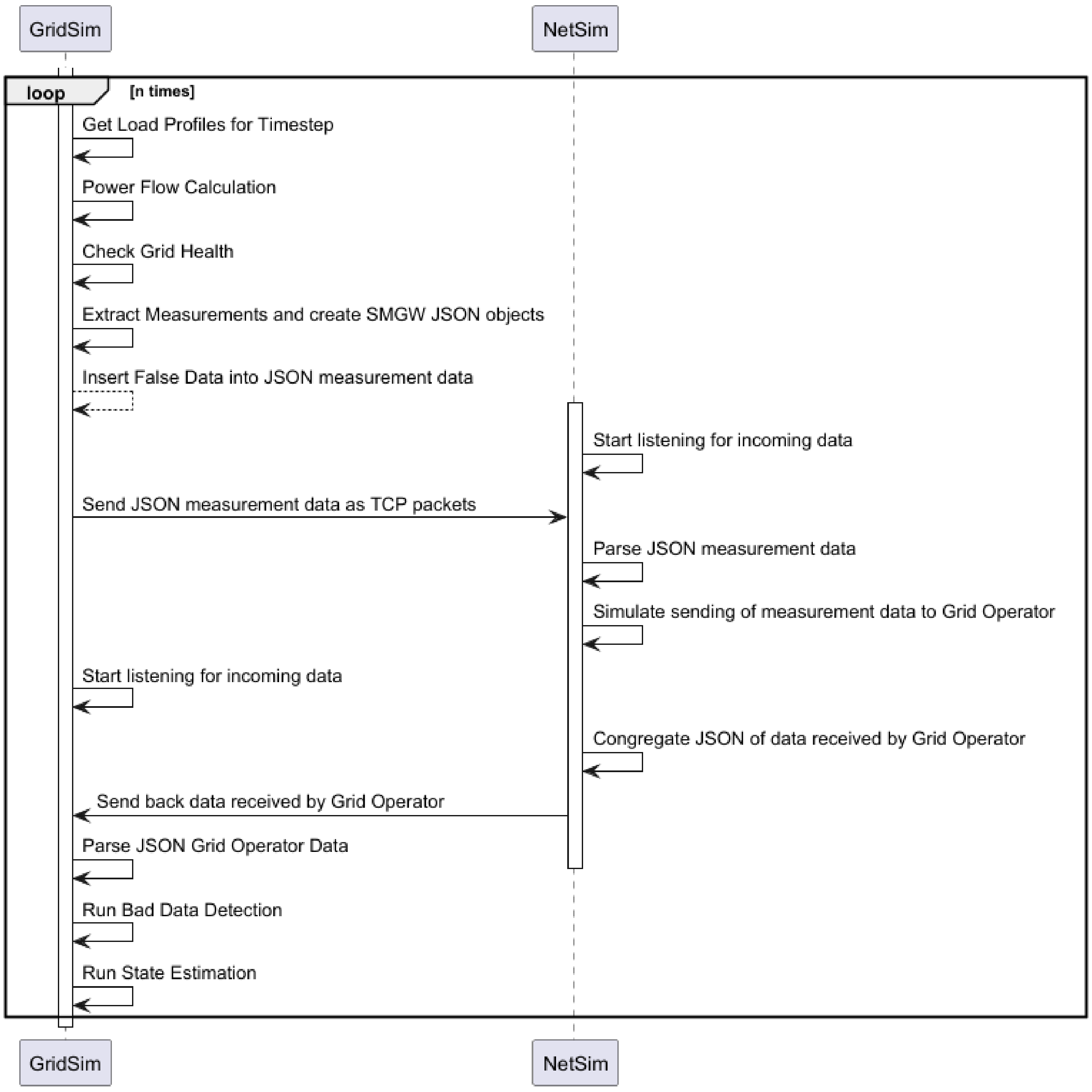
GridSim
NetSim
loop
[n times]
Get Load Profiles for Timestep
Power Flow Calculation
Check Grid Health
Extract Measurements and create SMGW JSON objects
Insert False Data into JSON measurement data
Start listening for incoming data
Send JSON measurement data as TCP packets
Parse JSON measurement data
Simulate sending of measurement data to Grid Operator
Start listening for incoming data
Congregate JSON of data received by Grid Operator
Send back data received by Grid Operator
Parse JSON Grid Operator Data
Run Bad Data Detection
Run State Estimation
GridSim
NetSim

Figure 3.4: Co-Simulation Sequence Diagram

3.3 Testing and Evaluation

In this section, we will outline the testing process for the simulation framework, as well as the evaluation of the implementation based on the previously developed requirements.

Before deployment for the evaluation of FDIA techniques, the designed simulation framework was tested in a process based on the ISO/IEC/IEEE 29119 software testing standard (ISO/IEC/IEEE, 2022). This standard ensures comprehensive testing procedures to validate and verify the functionality and performance of the software system. In this process, unit, integration, and system tests were conducted.

The implemented functions and components, as well as the external libraries, are tested in unit tests, in the case of the external libraries the provided unit and integration testing software was used to carry out the tests. The self-developed functions were tested with different inputs to assess their proper functioning as well as their behavior when provided with inputs that were not intended in their conception. E.g. the parsing functions were provided with a proper input but were additionally tested when provided with a random text string.

Following successful unit testing of the functions, integration tests were performed to evaluate the interaction between the functions in the two separate simulation tools. These tests we conducted to ensure that the functions worked together properly and in the way that was intended. This process was carried out iteratively so that bugs and errors that occurred during the testing were fixed afterward and then another test was conducted until all integrations functioned properly. The same procedure was carried out for integrating the two simulation tools in the co-simulation, where problems e.g. as previously described timing issues and race conditions, were handled and fixed to ensure a frictionless co-simulation process.

Finally, system tests were carried out, where the complete simulation process was evaluated, and like in the previous testing phases, changes were made to the simulation framework, so that it functions as intended.

3.3.1 Evaluation Based on the Requirements

The requirements for the simulation framework and its components are described in section 3.1. In this section, we will evaluate their fulfillment. Table 3.4 shows all requirements from the previous sections with their priority and status. As the table shows, all requirements that are prioritized as "Must" are fulfilled in the scope of the simulation design. Requirements R8-R10 are marked as fulfilled with an asterisk because their implementation will be described in the upcoming section. Requirement R11 is fulfilled by default, as PandaPower handles the calculation this way in case of isolation. The only requirement that could not yet be verified is CR7 which requires scalability and consistent performance since the simulation framework has not yet been evaluated outside of the scope of this work. This evaluation will be left for future development cycles of the simulation framework.

Table 3.4: Summary of Simulation Requirements

Requirement	Short Description	Priority	Status
R1	SMGW must transmit selected values to authorized recipients periodically or immediately.	Must	Fulfilled
R4	SMGW must transmit recorded values regularly if a sending period is parameterized (60s period supported).	Must	Fulfilled
R7	SMGW must support a 60-second sampling rate for measurement values.	Must	Fulfilled
R8	Transformer node power levels must be monitored continuously with warnings and notifications.	Must	Fulfilled∗
R9	Line current must be monitored with visual indicators when reaching fuse current.	Must	Fulfilled∗
R10	Provide visual indicator for line separation when line current reaches fuse current.	Must	Fulfilled∗
R11	Exclude sub-grid cut off by fuse failure from calculations.	Should	Fulfilled
CR3	Support standardized communication protocols and ensure interoperability with smart meter infrastructure.	Must	Fulfilled
CR4	Use standardized data exchange formats and clearly defined profile data.	Must	Fulfilled
CR5	Ensure high reliability, availability, data redundancy, and fault tolerance in communication.	Must	Fulfilled
CR7	Ensure communication infrastructure scalability and consistent performance.	Should	Not tested

Since all requirements except for CR7, which was assigned the priority "Should", have been fulfilled, the simulation framework performs as initially intended.

4 False Data Injection Attack

This chapter describes the integration of an FDIA into the co-simulation. We will first develop metrics for the impact of FDIA, then describe the implementation of the selected FDIA techniques and finally discuss the results of the simulation of these techniques on the co-simulation with an evaluation based on the metrics. We start with the random data FDIA, first with one available node, then with a fraction of available buses. We will then continue with the mathematical FDIA techniques first described by Liu et al. (2011) and finally utilize the machine learning and genetic algorithm-based FDIA described in section 2.4.

4.1 Metrics for the Effect of FDIA

In this section, we will analyze the goals of FDIA and use the targets of the FDIA techniques to develop metrics, which can then be used for evaluating the effect of a specific FDIA technique on the power grid in the co-simulation.

4.1.1 Grid Health Tool

The first metric we propose is based on the grid health tool based on requirements R8-R11 described in section 3.1.1. The grid health tool will be used to supervise the load of the transformer and the line current in each line in every time step. It is implemented via PandaPower's diagnostic tool, which provides functions for extracting the transformer load with a contingency test, as well as a function that provides a list of overloaded lines and isolated sections. The tool is implemented in two modes: verbose and non-verbose, where the verbose mode outputs the complete diagnostic and contingency report from the respective PandaPower implementations, which includes additional information, such as lines with impedances close to zero and voltage inconsistencies, while the non-verbose mode outputs if the transformer is overloaded and a list of overloaded lines as well as isolated sections due to the overloading of those lines. Since the contingency analysis tool offers information on which line is responsible for the overloading of the transformer, which could provide valuable insights into the effectiveness of specific compromised buses, this information is included in both modes as well. Algorithm 13 shows a pseudo-code description of the implementation of the grid health tool.

The metric of grid health is valued in the following manner: The *overload factor* is equal to the percentage of buses isolated due to the effects of the FDIA in each time step, divided by the number of time steps. The more buses are isolated, the higher the impact of the FDIA. If the transformer is overloaded, the complete grid will be cut off from the higher voltage levels and the overload factor for the time step will be classified as one. The grid health metric will be used as the highest metric for the effectiveness of the grid since it directly shows if the grid has been compromised.

Algorithm 13 check_grid_health

```
1: procedure CHECK_GRID_HEALTH(net, verbose=False)   ▷ False is set as the standard value for verbosity
2:     diagnostic ← pandapower.diagnostic(net)
3:     nminus1_cases ← line indices
4:     contingencies ← pandapower.contingency.run_contingency(net, nminus1_cases)
5:     if verbose then
6:         Print diagnostics
7:         Print contingencies
8:     else
9:         if 'overload' in diagnostic then
10:            Print overloads from diagnostic
11:            Print contingencies
12:        end if
13:        if isolated sections in diagnostic then
14:            Print isolated sections from diagnostics
15:            Print contingencies
16:            bus counter = 0
17:            for isolated section in isolated sections do
18:                for bus in isolated section do
19:                    bus counter ++
20:                end for
21:            end for
22:            return $\frac{\texttt{bus counter}}{\texttt{total no. of buses}}$
23:        end if
24:        if Trafo is overloaded then
25:            Print "Trafo is overloaded because of the following lines: "
26:            Print index of overload causing bus
27:            Print max. loading percent of trafo
28:            return 1.0
29:        else
30:            Print "Grid is healthy"
31:            return 0.0
32:        end if
33:    end if
34: end procedure
```

4.1.2 Deviation Metric

As a secondary metric, we will utilize the mean deviation from the original state estimation data. The state estimation will be conducted once on the original dataset without FDIA injection and then again with the injected attack vector. The differences will then be calculated for each node and measurement in each time step. From this we calculate the *Mean Nodal Deviations (MND)*, the mean of all nodal differences, as well as their standard deviations. The metrics utilized to rank the FDIA are the mean and the highest absolute value of the MND.

An FDIA will first and foremost be ranked regarding its effectiveness on the grid health factor and only secondarily on the deviation metrics. We will utilize the highest absolute values as the secondary ranking to the grid health, and the MND as the tertiary ranking. The implementation of the grid health tool and the calculation and visualization of these metrics in the form of helper functions will be outlined in the following.

To convey the effect of the deviation metric a plot is created, which shows an overview of the MND per measurement. To achieve this, a list of all differences DataFrames, whose creation will be explained in section 4.1.3, is created during the simulation and then used as input for the plotting function plot_mean_and_std. First, the function takes the list of DataFrames and calculates the mean values and their standard deviations of the four state estimation parameters per bus. It then creates a subplot for each measurement type, which shows the mean nodal deviation as a line graph and its standard deviation as an area around the line graph which is colored in a lighter shade of the line graph's color. Furthermore, the overall mean of the mean nodal values is depicted as a dotted line and its value is shown in a box in the upper right corner of each subplot. The pseudo-code for this function is provided in algorithm 14. An example of the created plot can be found in figure 4.1. [1]

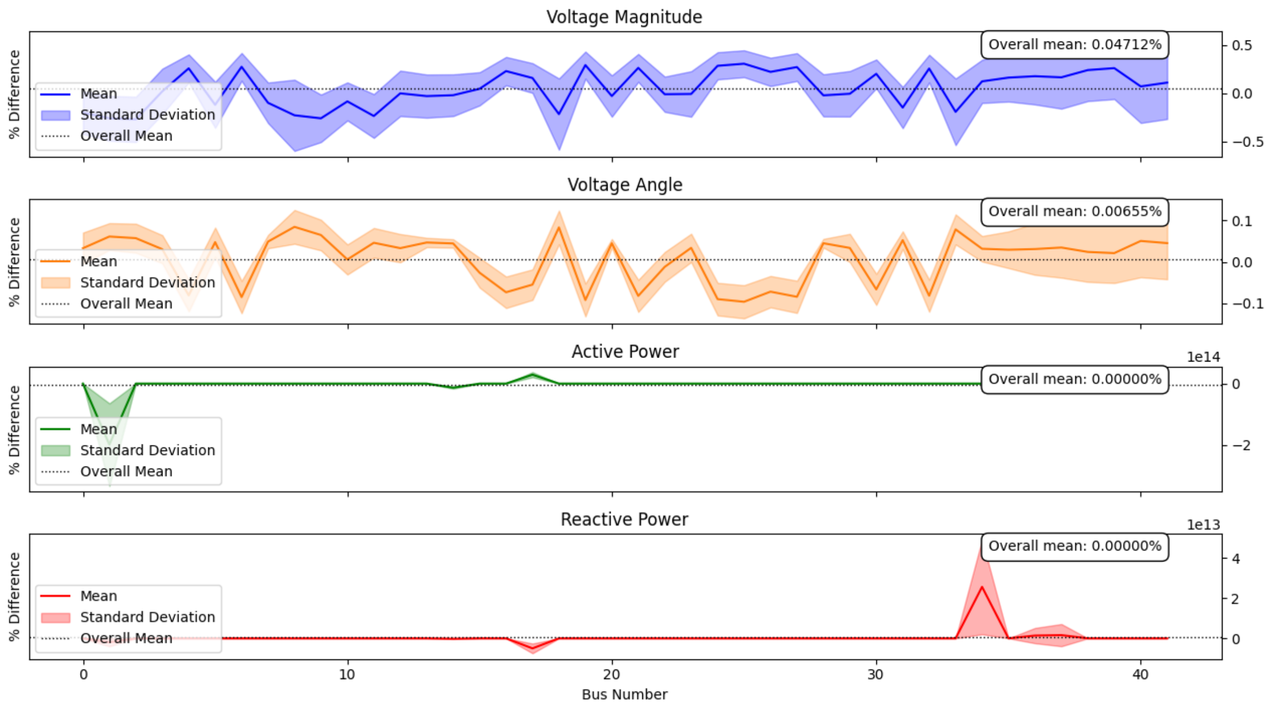

Figure 4.1: Example Plot of Mean Effect and Standard Deviation of FDIA

[1] The example plot and the following plots show a bug which depicted high value means as 0.00000 %. This bug has been fixed in the final code provided on the SD card.

Algorithm 14 plot_mean_and_std

```
1: procedure PLOT_MEAN_AND_STD(differences_list)
2:     Create DataFrame all_differences from differences_list
3:     means ← mean values per bus and measurement from all_differences
4:     stds ← standard deviations per bus and measurement from all_differences
5:     overall_means ← mean values of means DataFrame
6:     max_abs_value_info ← empty list
7:     fig, axes ← subplots of columns of means                ▷ columns = U, $\varphi$, P, Q
8:     for i, measurement in enumerate(means.columns) do
9:         mean_values ← mean values of measurement
10:        std_values ← std devs of measurement
11:        max_abs_value ← absolute max of selected means column
12:        bus_index ← index of absolute max
13:        std_bus ← std dev of absolute max index bus
14:        max_abs_value_info[measurement] ← (bus_index, max_abs_value, std_bus)
15:        ax ← measurement subplots
16:        Plot mean values as a line plot
17:        Plot standard deviations as a lightly colored area
18:        Plot dotted line for the mean value of means
19:        Plot title of measurement
20:        Plot legend in the bottom left corner
21:    end for
22:    Show plot
23:    for measurement in max absolute value info do
24:        Print "Highest abs value for measurement at bus bus_index:"
25:        Print max_abs_value ± std_bus")       ▷ Values are formatted with 5 significant digits
26:    end for
27:    return overall_means
28: end procedure
```

4.1.3 Helper Functions

Helper functions facilitate the computation of FDIA effects and provide visual comparisons of pre- and post-attack data. They allow for the analysis and graphical representation of attack impacts, which is crucial for validating the effectiveness of different FDIA strategies. The grid health tool and the plot_mean_and_std can also be classified as helper functions but were provided their sections based on their importance as effectiveness metrics for FDIA techniques.

The first helper function, compute_differences, is used to calculate the percentage impact of an FDIA on the state estimation. It takes the state estimation results without FDIA manipulation (correct_data) and the state estimation results after FDIA manipulation (fdia_data) as input. Both are Pandas DataFrames and contain the estimation results for each measurement in a separate column. The difference is computed for each bus as the difference of the manipulated data minus the original estimation result, divided by the original estimation result. This is then multiplied by

100 to reflect a percentage difference. Algorithm 15 depicts the implementation in a pseudo-code format.

Algorithm 15 Compute Percentage Differences

```
1: procedure COMPUTE_DIFFERENCES(correct_data, fdia_data)
2:     differences = new DataFrame
3:     for column in correct_data.columns do
4:         Compute percentage difference for each measurement type:
5:         $\texttt{differences[column]} = \frac{\texttt{fdia\_data[column]} - \texttt{correct\_data[column]}}{\texttt{correct\_data[column]}} \times 100$
6:         The differences show how far the FDIA data deviates from the expected data.
7:     end for
8:     return differences
9: end procedure
```

The second helper function, plot_differences, first calls the compute_differences function, and then plots the returned results in a separate subplot for each measurement value, i.e. voltage, voltage angle, active power, and reactive power, similar to the plot_mean_and_std function. Moreover, it outputs the mean difference per measurement type to provide an overview of the average effectiveness of the FDIA. These mean values are also included in the plot in the form of a dotted line. An example plot can be found in figure 4.2 and a pseudo code of the implementation is provided in algorithm 16.

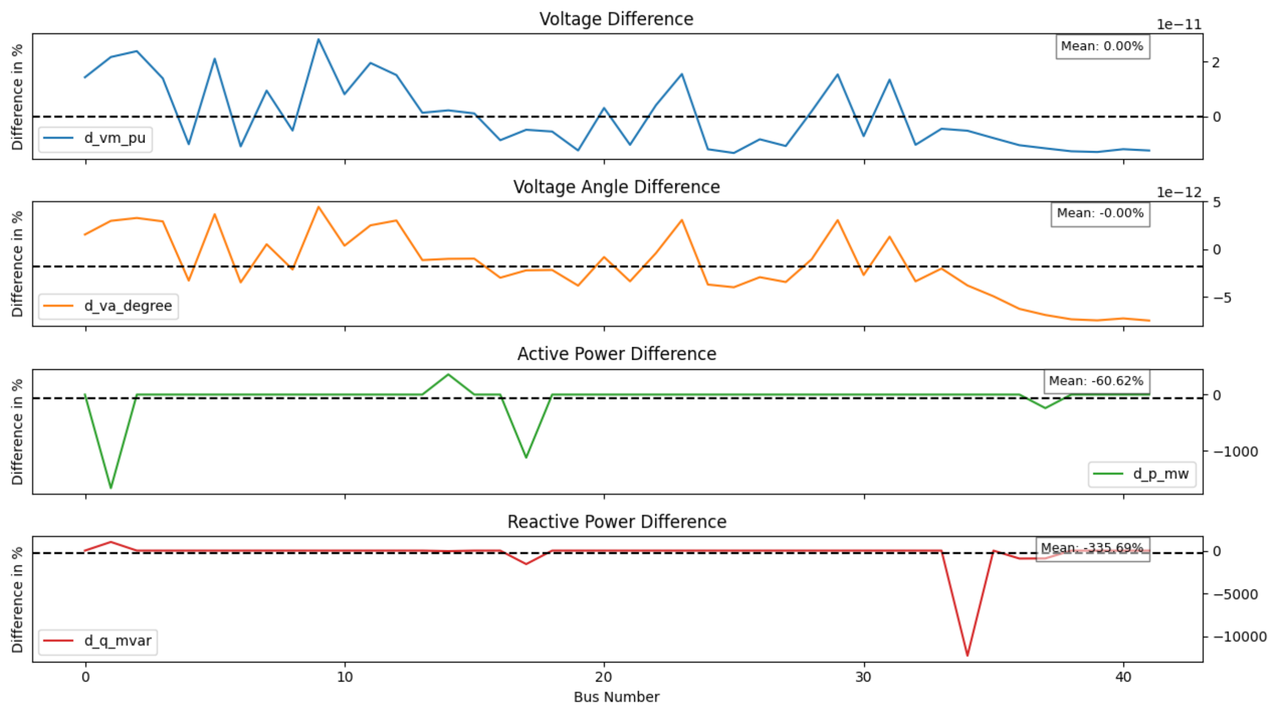

Figure 4.2: Example Plot of State Estimation Differences before and after FDIA

Algorithm 16 Plot Differences Between Correct and FDIA Data

```
1: procedure PLOT_DIFFERENCES(correct_data, fdia_data)
2:     differences ← compute_differences(correct_data, fdia_data)
3:     Print "Average Differences in %", mean values of differences per measurement
4:     Plot differences for each bus, displaying voltage, active, and reactive power differences.
5:     Plot dotted line for the mean value of each measurement
6:     Show plot
7: end procedure
```

The last helper function is used to visualize the location of compromised buses / SMGWs in the power grid. It takes the net variable and the list of attack buses as input values plots the grid with the underlying grid geodata, and represents uncompromised buses as blue dots, transmission lines as black lines, and the compromised buses as red dots to directly distinguish them from the uncompromised buses.

Algorithm 17 Visualize Attacks with Bus Geodata

```
1: procedure PLOT_ATTACK(net, attack_buses)
2:     Extract geodata for attack and non-attack buses.
3:     for line in net do
4:         Plot each line considering start and end bus positions.
5:     end for
6:     Plot attacked buses as red dots and non-attacked as blue dots.
7:     Annotate each bus with its number to identify regions affected by FDIA.
8:     Show the plot, which provides an overview of attack placement across the grid.
9: end procedure
```

4.2 Implementation of False Data Injection Attacks

This section details the implementation of selected FDIA intended to compromise the state estimation processes in power grid systems. These attacks are strategically designed to inject malicious data into measurement systems, thereby altering perceived system states, without triggering standard detection mechanisms as outlined in section 2.4. The implementations leverage both mathematical approaches and advanced machine-learning techniques.

4.2.1 Random Data FDIA

This function implements random false data injection, aiming to obscure system states through noise addition. This technique is used to create a baseline for the effectiveness of other FDIA techniques. To create an attack vector that can bypass bad data detection, the boundaries of the randomly generated values have to be experimentally approached. The highest values for the bounds that were not flagged by the bad data detection in any iteration can be found in table 4.1.

Table 4.1: Minimum and Maximum Values of Parameters

Parameter	Minimum Value	Maximum Value
U	0.95 kV	1.05 kV
P	10^{-6} MW	10^{-2} MW
Q	-10^{-5} Mvar	10^{-5} Mvar

The experiments to approach the bad data detection boundaries were tested with the lowest and highest value for each measurement from the dataset created for the training of the machine learning model, which will be explained in an upcoming section. In the case of the power measurements, the exponents were then decreased, until the measurements passed the bad data detection for all time steps of a full day simulation. The voltage values provide a smaller tolerance. Their boundaries were decreased in steps of 0.05, resulting in the final boundaries from the table. In future works, the boundaries could be approached with smaller incrementation steps or other techniques to create higher impacts of the FDIA techniques utilizing those boundaries.

Algorithm 18 Random Data FDIA Implementation

```
1: procedure RANDOM_FDIA(buses, measurements)
2:     for each bus in buses do
3:         for each measurement in measurements do
4:             if measurement corresponds to the specified bus then
5:                 Generate a random value for ActivePower within (10e-6, 10e-2).
6:                 Generate a random value for ReactivePower within (-10e-5, 10e-5).
7:                 Generate a random value for Voltage within (0.95, 1.05).
8:                 Update the measurement's data with these random values.
9:             end if
10:        end for
11:    end for
12:    return modified measurements
13: end procedure
```

4.2.2 Random Non-Generalized FDIA using Liu's Method

Utilizing Liu's approach, this function constructs random FDIA vectors that better align with system vulnerabilities identified in extensive grid studies. This method is dependent on altering measurement data in a manner that avoids easy detection.

Algorithm 19 Random Non-Generalized FDIA Liu's Method

```
1: procedure RANDOM_FDIA_LIU(buses, measurements, net, H)
2:     Identify I_meter, the indices of all buses minus the attack buses
3:     for each index j in I_meter do
4:         Find a non-zero element in H's column for row j.
5:         if such non-zero element exists then
6:             Swap this column with the first column of H.
7:             for each column i from 1 to n do
8:                 Zero out the element at H[j, i] using elementary row operations.
9:             end for
10:        end if
11:        Repeat row operations for all j in I_meter.
12:    end for
13:    Select an attack vector column from H that satisfies conditions.
14:    for each bus in buses, each corresponding measurement do
15:        Apply selected attack vector to measurements.
16:    end for
17:    return modified measurements
18: end procedure
```

4.2.3 Random Generalized FDIA using Liu's Method

This function expands on the Liu method by generalizing the attack, allowing for greater flexibility in impact via adapting the measurement matrix and considering permissible manipulation ranges.

Algorithm 20 Random Generalized FDIA Liu's Method

```
 1: procedure RANDOM__GENERALIZED__FDIA__LIU(buses, measurements, net, H)
 2:     Calculate State Estimation Residual $\tau_a$
 3:     Prepare I_meter and H as in algorithm 19.
 4:     for each index j in the reduced I_meter do
 5:         Perform column swaps and zeroing techniques to manipulate H.    ▷ Target vector
        formation ensures columns can be reoriented for optimal attack efficiency.
 6:     end for
 7:     Generate attack vector $\mathbf{t}$ filling random indices constrained by $\tau_a$.
 8:     Compute generalized vector $\mathbf{a}'$ that solves $\mathbf{B}'\mathbf{a}' = \mathbf{B}\mathbf{t}$.
 9:     for each bus, each corresponding measurement do
10:         Inject components of $\mathbf{a}'$ to measurements.
11:     end for
12:     return measurements
13: end procedure
```

4.2.4 Machine Learning and Genetic Algorithm FDIA

The machine learning and genetic algorithm FDIA (MLGA-FDIA) leverages ANNs to model and exploit vulnerabilities in power system state estimation. This method involves constructing datasets from measurement data, training predictive models on these datasets, and using the models to craft optimal attack vectors. The approach enhances the stealth and effectiveness of FDIAs by learning complex relationships between variables that might not be easily spotted by traditional detection methods. The process involves several key steps: dataset creation, feature selection and preprocessing, model training, and optimization using genetic algorithms.

Building the Dataset

The dataset serves as the foundational component, providing the input and output data required for training the machine learning model. For each of the 43 buses in the power grid, three measurement parameters have to be considered: voltage, active power, and reactive power. Consequently, the dataset includes a total of 129 input features. The outputs to be predicted by the model encompass four parameters for each node: voltage estimate U_{est}, voltage angle estimate φ_{est}, active power estimate P_{est}, and reactive power estimate Q_{est}, resulting in a total of 172 output variables. The dataset is created by running a random data FDIA attack on the attack buses ten times for each time step in a full-day simulation, resulting in 960 total sets of input and output parameters. The implementation of the dataset build process is shown in algorithm 21.

Algorithm 21 Deep Learning FDIA Dataset Construction

```
1: procedure DEEP_LEARNING_FDIA_BUILD_DATASET(measurements, original_vals, net)
2:     Initialize dataset to store measurement differences and state estimates.
3:     for each bus in the network excluding transformer buses do
4:         for each measurement matching the bus do
5:             Compute voltage difference:
6:             measurement["Voltage"] - U of original_vals[bus].
7:             Compute active power difference:
8:             measurement["ActivePower"] - P of original_vals[bus].
9:             Compute reactive power difference:
10:            measurement["ReactivePower"] - Q of original_vals[bus].
11:            Append these differences to the dataset vector.
12:        end for
13:        Append state estimates from net.res_bus_est for each bus.
14:    end for
15:    return constructed dataset vector
16: end procedure
```

In future research, the number of data points in the dataset could be increased by running the function over the full load profile set to improve the performance of the model.

Feature Selection & Model Training

Since the evaluation is focussed on using a fraction of malicious buses for the FDIA, the measurements of a subset of critical buses, whose selection process will be described in the following section, will be selected as features the model can influence. The step of preprocessing also includes normalization of the data, which transforms input data to a common scale, between 0 and 1, which aids in faster convergence and stability during model training. Furthermore, the boundaries created for the random data FDIA will be applied in the model training to ensure that the model does not produce unrealistic measurements. To avoid overfitting, the dataset is split into training and testing subsets.

The core of the optimization process is a neural network model that learns the mapping from system inputs to outputs. A multi-layer perceptron (MLP) architecture is employed, implemented using the TensorFlow Python library. This model comprises several dense layers, each acting as a set of neurons that transform the input data. Specifically, the architecture features two hidden layers with 256 neurons each, utilizing the Rectified Linear Unit (ReLU) activation function, which introduces non-linearity essential for capturing complex patterns. Training is conducted over 100 epochs with a batch size of 32, using the Adam optimizer, a popular choice for its efficiency and adaptive learning rate capabilities (Deru and Ndiaye, 2020). The model minimizes the mean squared error loss, similar to the process of state estimation itself. The feature selection and model training are implemented as depicted in algorithm 22.

Algorithm 22 Training the ANN Model for FDIA

```
 1: procedure DEEP__LEARNING__FDIA__TRAIN__MODEL
 2:     Load the dataset and extract input/output features corresponding to attack buses.
 3:     Define bounds for inputs, ensuring that generated vectors fall within realistic measurement
        deviations.
 4:     Normalize data with StandardScaler to center it around zero with a unit variance.
 5:     Split data into training (80%) and testing (20%) subsets.
 6:     Construct a neural network with two hidden Dense layers using ReLU activations to capture
        non-linearities.
 7:     Train the model using Adam optimizer and mean_squared_error loss over 100 epochs, with
        validation splits to monitor overfitting.
 8:     Evaluate the trained model on the test set to determine its performance.
 9:     return the trained model and input bounds
10: end procedure
```

Predicting Attack Vectors Using Genetic Algorithms

Once trained, the neural network is used to estimate system outputs for given inputs, forming the basis for optimization. GA is then applied to identify the input configurations that maximize desired output parameters. GA is a class of optimization algorithms inspired by natural selection and genetic evolution. (Grefenstette, 1993) They operate by initializing a population of candidate solutions (input vectors), which evolve over successive generations. Each candidate solution is evaluated using the pre-trained model to predict the output it would produce, which serves as a fitness score for optimization. The algorithm iteratively applies genetic operators such as selection (choosing the best candidates), crossover (combining parts of different candidates to form new ones), and mutation (introducing random changes), driving the population toward optimal solutions. For each generation, a repair function is applied that removes candidate solutions exceeding the boundaries for the measurements to ensure that the final attack vector can bypass the bad data detection. The prediction function's implementation is outlined in algorithm 23.

Algorithm 23 GA-Based Prediction of Optimal FDIA Vectors

```
1: procedure DEEP_LEARNING_FDIA_PREDICT(model, bounds)
2:     Define a fitness evaluation function that uses the neural network to score potential attack vectors.
3:     Establish initialization parameters for the genetic algorithm (population size, mutation rate).
4:     Create Individual and FitnessMax classes to handle solution encoding.
5:     Setup GA operations, including mutation bounded within realistic data limits and tournament selection for reproduction.
6:     for each generation in GA (totaling ngen) do
7:         Apply crossover and mutation to evolve potential solutions.
8:         Evaluate offspring using the ANN to update their fitness scores.
9:         Select the best-performing individuals for subsequent generations.    ▷ This iterative refinement homes in on input vectors that achieve maximal disruptive impact.
10:    end for
11:    return the optimal attack vector produced by the GA
12: end procedure
```

Injecting Optimized Attack Vectors

The final step involves deploying the crafted optimal attack vector to alter system measurements. By applying this vector to the targeted measurements, the adversary can induce substantial state estimation errors, potentially disrupting grid operations without immediate detection.

Algorithm 24 Injection of Optimized FDIA Vectors

```
1: procedure DEEP_LEARNING_FDIA_INJECT(att_vector, buses, measurements)
2:     Initialize a copy of att_vector for manipulation.
3:     for each bus in buses and corresponding measurement do
4:         Retrieve and apply sequential elements from the att_vector to update:
5:         measurement["ActivePower"], measurement["ReactivePower"], and measurement["Voltage"].    ▷ The application is performed in reverse order to synchronize with att_vector's pop operation.
6:     end for
7:     return modified measurements augmented by the optimized attack vector
8: end procedure
```

4.3 Evaluation of FDIA techniques

In this section, we will test the implemented FDIA techniques with the co-simulation and evaluate them based on the metrics we developed. First, we will evaluate the impact of single buses and arrays of buses to make sure we can create the highest possible impact on the power grid. Afterward, we will simulate the FDIA techniques in two scenarios: First, we will let the co-simulation run as described in section 3.2, where the load profile data is used as input for the power flow calculation and subsequently the creation of the measurement data. Secondly, we will simulate the FDIA in a scenario where we take the state estimation results of the previous iteration as input values for the power flow calculation. By inputting the state estimation values directly into the `net` variable, the grid health tool can take the FDIA into account, which in the first scenario is not possible, since the state estimation results are never put into the "real grid"[2].

4.4 Finding the Ideal Attack Buses

To find out where an attacker might have the biggest influence on the variables in the state estimation, a random FDIA was carried out for each bus in every available load profile timestep. Additionally, a state estimation without any FDIA was carried out, and the differences between the values of the augmented state estimation and the original state estimation were calculated using the compute_differences function. For each timestep, the bus with the highest absolute mean impact was selected for each of the four output values. After simulating a complete daily simulation[3], the buses that had the highest impact on each respective value the most times were selected. The results of this simulation are depicted in table 4.2, which shows that the results were almost the same in each of the five simulation iterations.

Table 4.2: Bus with the most influence on estimated values

Iteration	U	φ	P	Q
1	42	32	0	0
2	42	32	0	0
3	42	32	7	$\{0,1\}$
4	42	32	0	0
5	42	32	0	0

The most effective buses for each measurement, bus 42 for the voltage magnitude, bus 32 for the voltage angle, and bus 0 for active and reactive power, are marked as red in figure 4.3, which depicts their position in the grid using the plot_attack function described earlier.

[2]Meaning they are not used as input for any of PandaPower's underlying functions, except for being overwritten in the next iteration

[3]i.e. 96 time steps

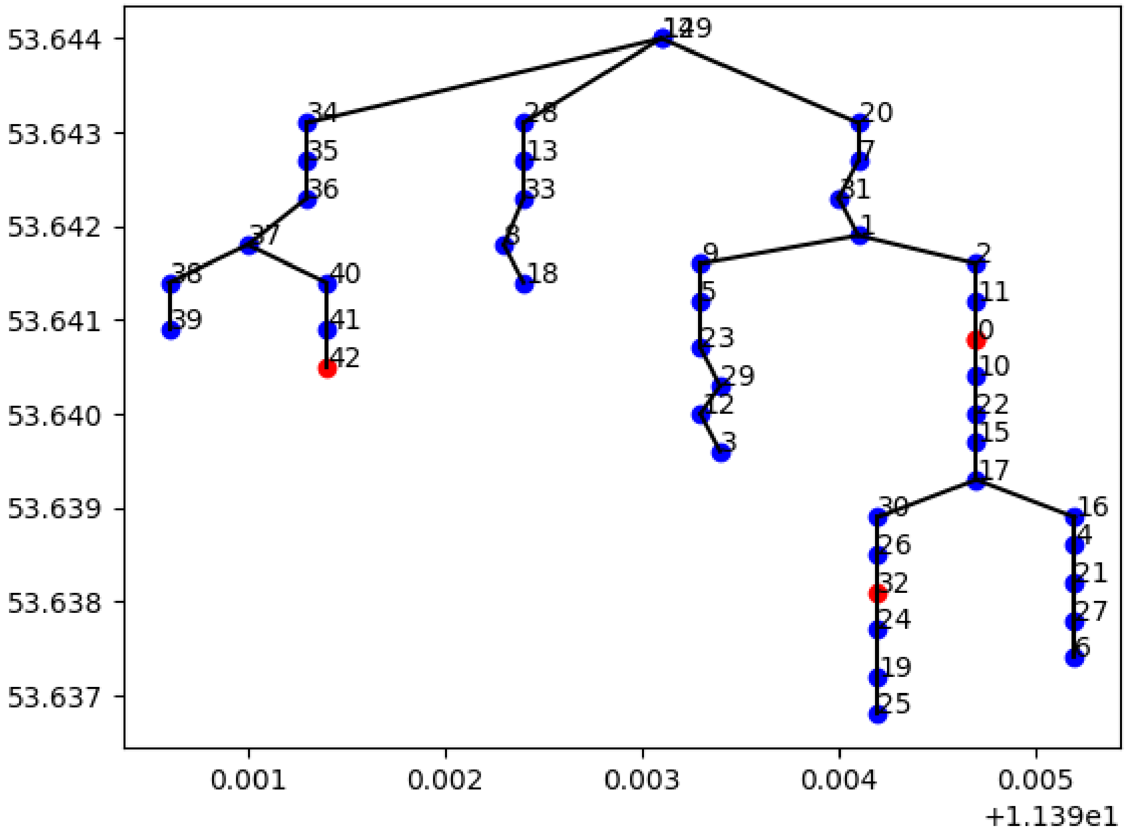

Figure 4.3: Positions of the buses from table 4.2 in the Grid

To use the random non-generalized FDIA method described by Liu et al. (2011), testing showed that at least six attacking buses are needed for the successful calculation of an attack vector in the specific grid configuration. To find the six most effective buses, the most intuitive way would be to try all possible combinations and compare the results analogously to the previous simulation. Since the grid has 43 buses, this would however yield over 8 million different combinations, which would make the simulation take a tremendous time to finish. An alternative solution, which was pursued in this work, would be to select the best bus for the selected variable as we have done before, then remove it from the list of buses that are tested and add it to the list of attacking buses. This yields the six best attacking buses for a specific value. The resulting attacking vector for impacting the active power P, which was chosen because the overloading of the transformer was designated as the highest goal for the attacker, consists of the buses 0, 1, 2, 8, 9, and 40, which are the buses utilized for testing all FDIA techniques in both testing scenarios. The buses are shown in figure 4.4.

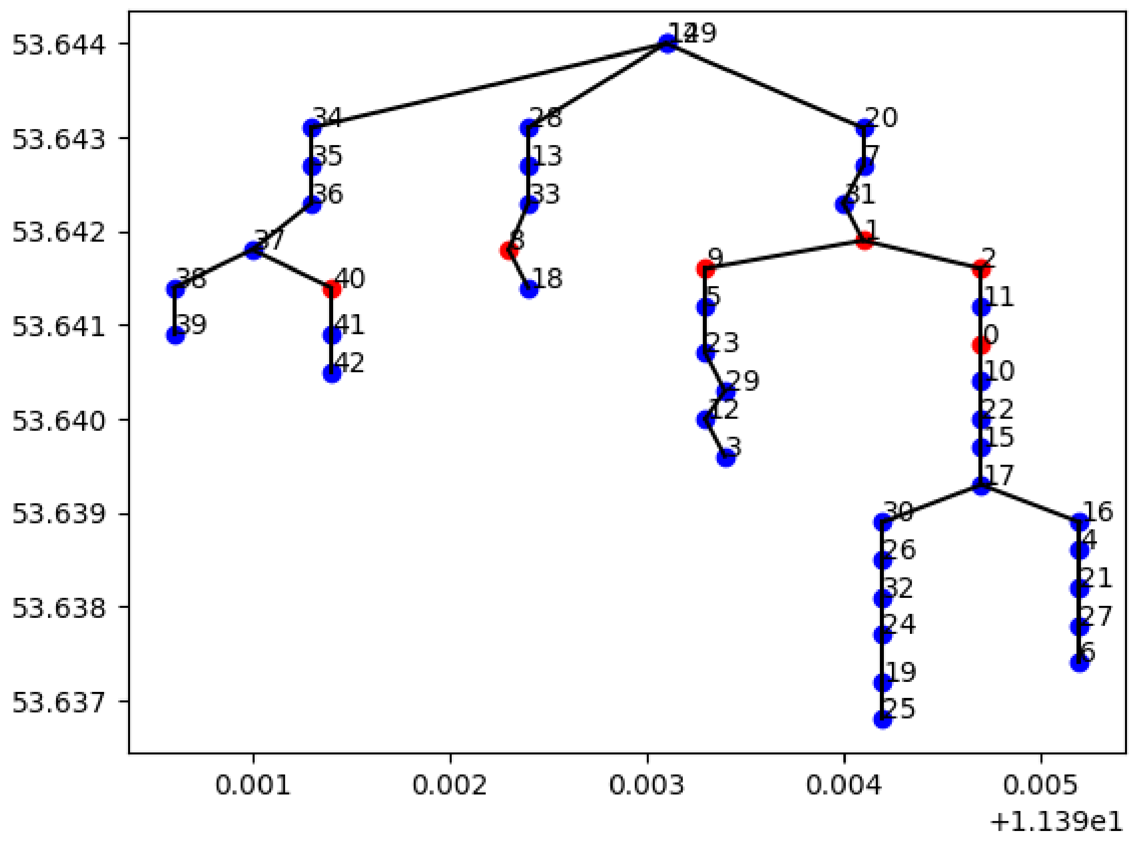

Figure 4.4: Ideal Attacking buses for impacting P

4.4.1 Results of FDIA with Power Flow based on load profile data

This section shows the results of the FDIA techniques using the load profile data as inputs for the power flow in each time step. Since the `net` variable is not impacted by the state estimation results in this case, except of course the `net.res_bus_est`, the grid health tool can not be utilized in the evaluation of the attack techniques. Thus, the highest absolute mean deviation per value will be used as the primary metric to rank the techniques' effectiveness, with the means of the MND as a secondary metric.

Random Data FDIA

The random data FDIA was evaluated creating a new random data attack vector for each timestep. As figure 4.5 shows, the voltage magnitude and voltage angle differences for the random data FDIA are almost mirror images of one another. The highest absolute MND can be seen at bus 25 for voltage and voltage angle, at bus 17 for the active power, and at bus 14 for the reactive power. The standard deviation seems to be relatively uniformly distributed for the voltage and voltage angle, which are affected across all buses, whereas a deviation for active and reactive power can only be seen at several buses, with no visible standard deviations at the non-affected buses. This could suggest that the active and reactive power can only be influenced by a specific number of buses for each attack bus configuration. The underlying shape of the plot seems to be similar through the iterations, which could be used as a characteristic for distinguishing between the FDIA techniques.

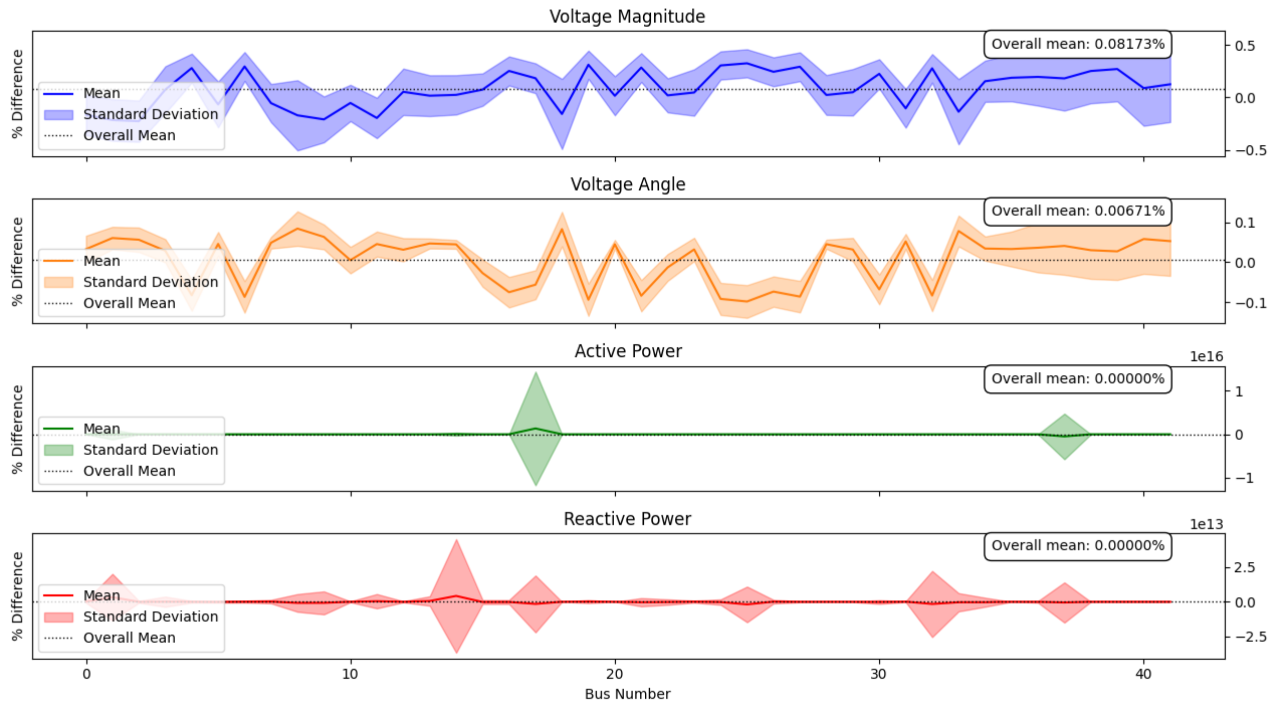

Figure 4.5: Impact of random data FDIA

The means of the MND for the random data FDIA are listed in table 4.3. The mean values range between 10^{-1} and 10^{-3}% for voltage magnitude and angle, while the power values range between 10^{10} and 10^{14}%. A ranking of the techniques based on these values will be done at the end of the section.

Table 4.3: Means of MND (Random Data)

ΔU	$\Delta\varphi$	ΔP	ΔQ
$8.173049 \cdot 10^{-2}$	$6.713094 \cdot 10^{-3}$	$1.939225 \cdot 10^{13}$	$7.659167 \cdot 10^{10}$

Random Non-Generalized Liu FDIA

Similar to the random data FDIA, the random non-generalized Liu FDIA was conducted with newly calculated attack vectors for each time step. Analyzing the shape of the plots, the random non-generalized Liu FDIA also seems to have a characteristic shape that is only changed in magnitude through the iterations. In contrast, however, the voltage magnitude and angle are not mirrored as in the random data FDIA. However, the amount of nodes affected by its active and reactive power measurements is also limited. The highest absolute MND can be found at bus 1 for voltage magnitude, active and reactive power, and at bus 2 for voltage angle.

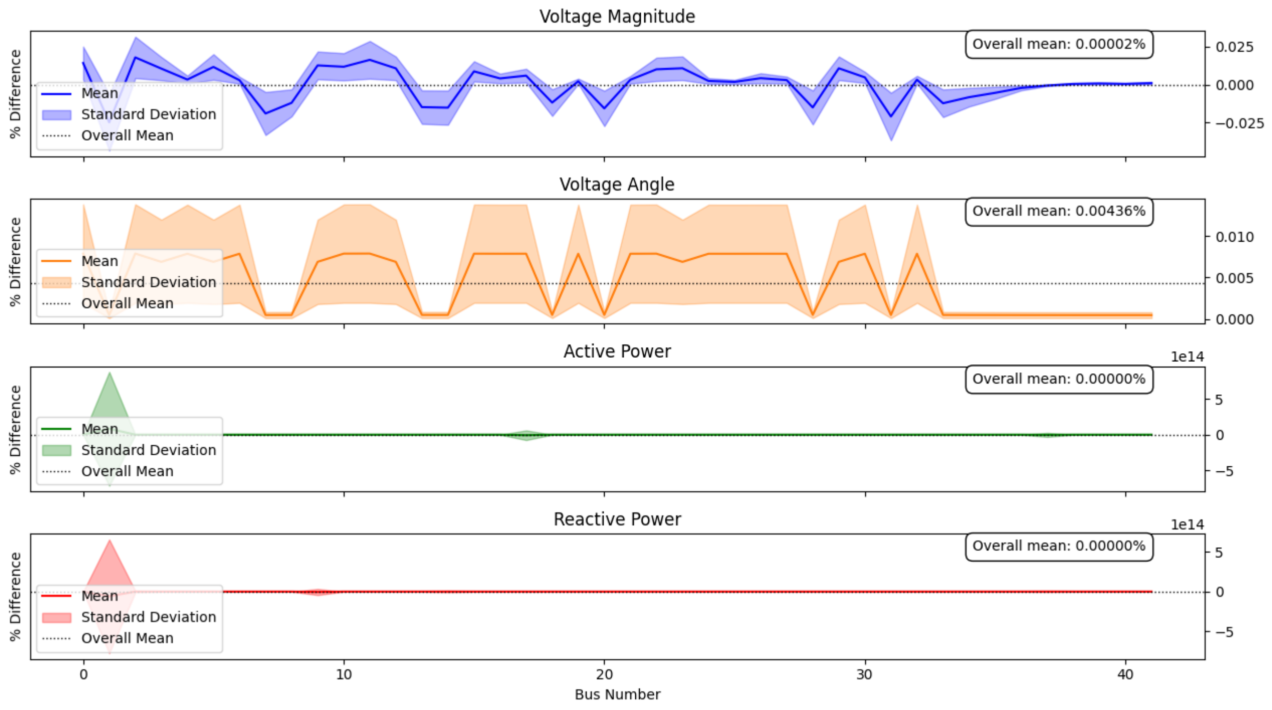

Figure 4.6: Impact of Random Non-Generalized Liu

The means of MND, depicted in table 4.4, show that in the secondary metric, the non-generalized Liu FDIA outperforms the random data FDIA only in the reactive power deviation.

Table 4.4: Means of MND (Random Non-Generalized Liu)

ΔU	$\Delta \varphi$	ΔP	ΔQ
$1.771321 \cdot 10^{-5}$	$4.356067 \cdot 10^{-3}$	$1.660787 \cdot 10^{12}$	$-1.421976 \cdot 10^{12}$

Random Generalized Liu FDIA

The random generalized Liu FDIA was conducted with newly calculated attack vectors for each time step, as were the previous two techniques. The plots for voltage magnitude and angle seem to be related and have the same underlying shape for each iteration, as was the case for the random non-generalized Liu FDIA. Furthermore, this technique only impacts a select number of buses' active and reactive power values, as seen before. The highest absolute MND can be seen at bus 9 for the voltage magnitude, at bus 25 for the voltage angle, at bus 17 for active power, and at bus 1 for reactive power.

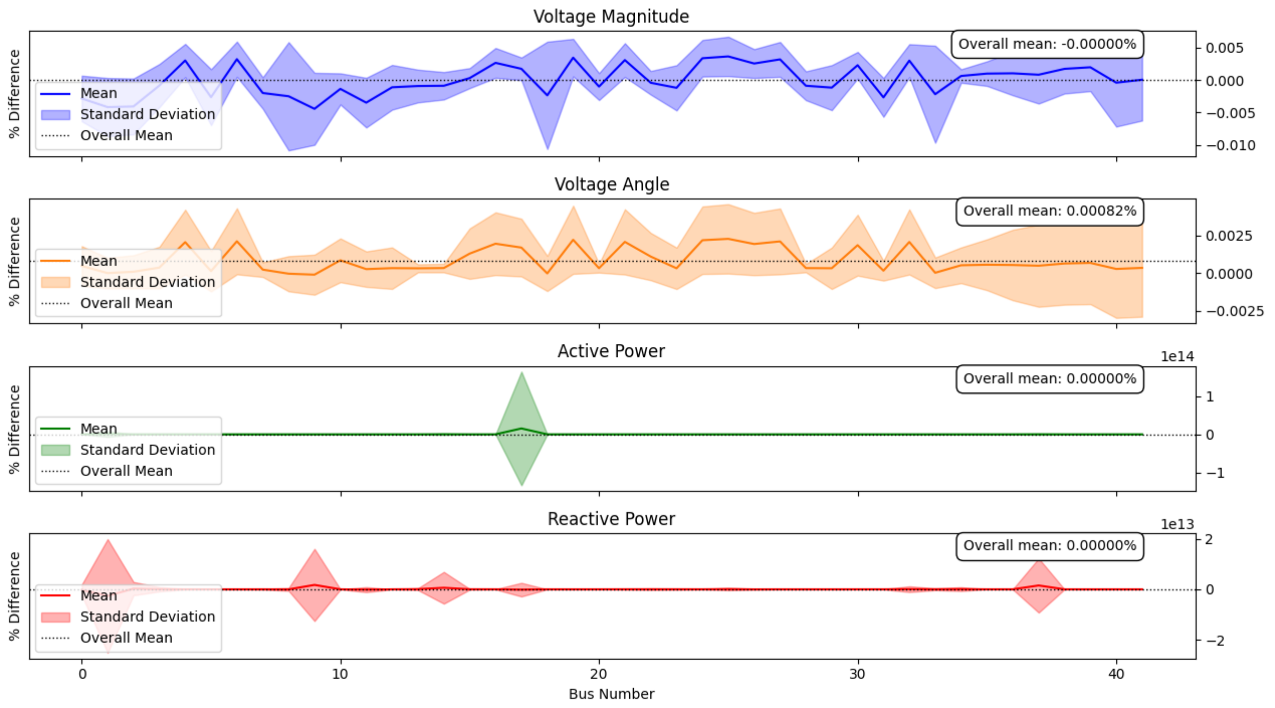

Figure 4.7: Impact of Random Generalized Liu

As table 4.5 shows, the technique underperforms both previous techniques in all other measurements. Since the generalized FDIA should have fewer constraints than the generalized one, this result is surprising and will be evaluated again after the testing in the second scenario.

Table 4.5: Means of MND (Random Generalized Liu)

ΔU	$\Delta \varphi$	ΔP	ΔQ
$-2.995578 \cdot 10^{-7}$	$8.197380 \cdot 10^{-4}$	$3.808114 \cdot 10^{11}$	$6.312517 \cdot 10^{10}$

MLGA-FDIA

The evaluation of the MLGA-FDIA was conducted in a different manner than the other three techniques. Since the creation of a single attack vector takes on average just under 10 minutes on the system it was tested on[4], which makes it unfeasible to calculate a new attack vector for every time step. However, as the simulation shows, one computed attack vector can be used multiple times and still create large impacts, as long as this behavior is not noticed and stopped by the grid

[4]i7-11700 16 core processor at 2.5 GHz, 32 GB of RAM and a GTX 3060Ti graphics card

operator. To validate the results, the simulation was carried out five times to see if the results were similar.

As figure 4.8 shows, the standard deviations are significantly lower in the plots for voltage magnitude and voltage angle, which can be attributed to the usage of the same attack vector in each iteration. However, the standard deviations for the active and reactive power at the affected nodes are quite high in comparison with the other plots. The highest absolute MND for the MLGA-FDIA is produced at bus 18 for the voltage magnitude, bus 40 for the voltage angle, bus 17 for the active power, and bus 1 for the reactive power.

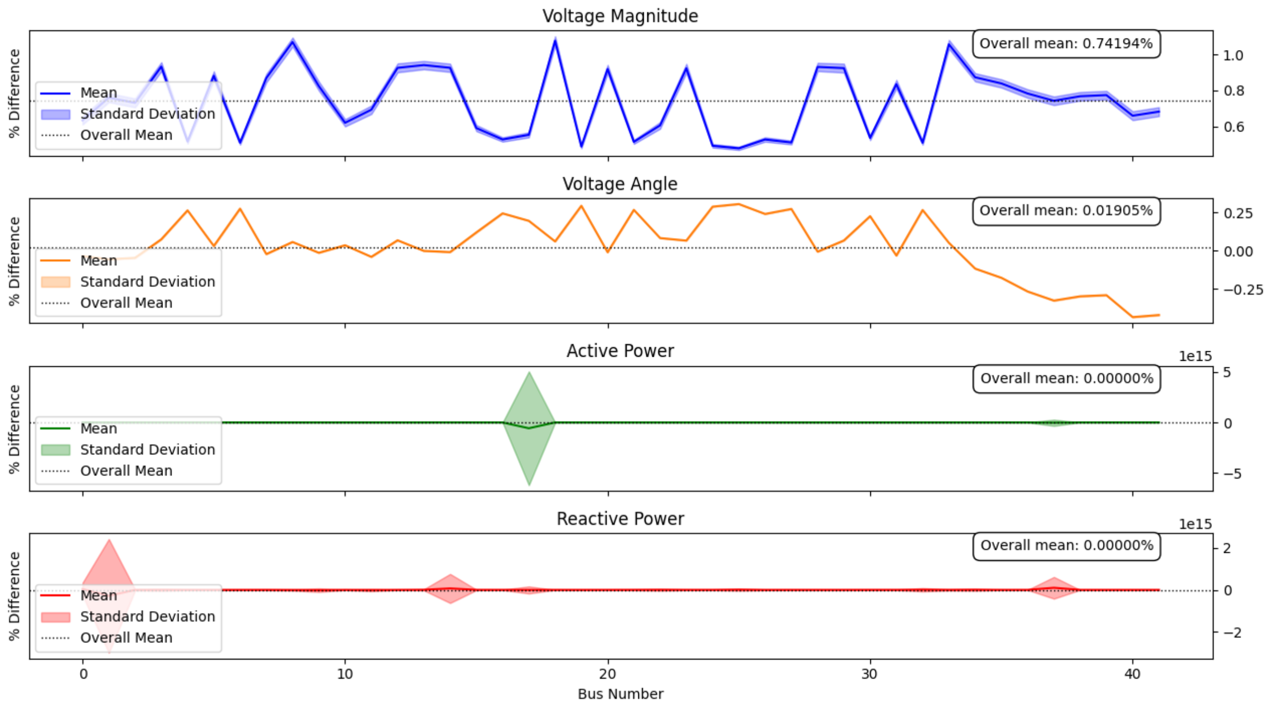

Figure 4.8: Impact of MLGA FDIA

As can be seen in table 4.6, the results for means of MND do not differ significantly between the five evaluation runs. The MLGA-FDIA outperforms the other three techniques in all categories, except for the active power, where it falls just under the random data FDIA.

Table 4.6: Means of MND (MLGA)

Iteration	ΔU	$\Delta\varphi$	ΔP	ΔQ
0	$7.419424 \cdot 10^{-1}$	$1.904808 \cdot 10^{-2}$	$-1.373696 \cdot 10^{13}$	$1.683446 \cdot 10^{12}$
1	$7.421366 \cdot 10^{-1}$	$3.392710 \cdot 10^{-2}$	$-1.373696 \cdot 10^{13}$	$7.582626 \cdot 10^{12}$
2	$7.419482 \cdot 10^{-1}$	$1.898896 \cdot 10^{-2}$	$-1.373373 \cdot 10^{13}$	$1.681197 \cdot 10^{12}$
3	$7.419444 \cdot 10^{-1}$	$1.898418 \cdot 10^{-2}$	$-1.371866 \cdot 10^{13}$	$1.679924 \cdot 10^{12}$
4	$7.419346 \cdot 10^{-1}$	$1.911865 \cdot 10^{-2}$	$-1.372116 \cdot 10^{13}$	$1.683592 \cdot 10^{12}$
Mean	$7.4198124 \cdot 10^{-1}$	$2.2013394 \cdot 10^{-2}$	$-1.3729494 \cdot 10^{13}$	$2.862157 \cdot 10^{12}$

Comparison of Techniques

After evaluating all four implemented FDIA techniques in the first scenario, the techniques will be ranked according to their performance. As described earlier, the highest absolute MND is the primary decision factor in this scenario. An overview of these values for all four techniques can be seen in tables 4.7 and 4.8. The rankings for the highest absolute MND and means of MND are shown in tables 4.9. As the tables show, the ranking is the same for both the primary and the secondary metric, with the MLGA-FDIA performing the best, the random data FDIA performing the second best, Liu's random non-generalized technique performing the third best and Liu's random generalized technique consistently performing the worst out of the four techniques.

Interestingly, the highest absolute MND for active and reactive power was recorded three out of four times at bus 17 and 1 for the respective measurements. While bus 1 was part of the attack buses, bus 17 is the next bus behind bus 1 where two grid branches meet.

Table 4.7: Comparison of FDIA Techniques (Highest Absolute MND) - 1

Technique	**Success Rate**	**Bus**	ΔU	**Bus**	$\Delta\varphi$
Random	1.0	25	0.32586 ± 0.13411	25	0.09861 ± 0.04097
Random Non-Generalized Liu	1.0	1	0.02483 ± 0.01855	2	0.00789 ± 0.00595
Random Generalized Liu	1.0	9	0.00442 ± 0.00554	25	0.00227 ± 0.00230
MLGA	1.0	18	1.07632 ± 0.02506	40	0.43211 ± 0.00306

Table 4.8: Comparison of FDIA Techniques (Highest Absolute MND) - 2

Technique	**Bus**	ΔP	**Bus**	ΔQ
Random	17	$1.337 \cdot 10^{15} \pm 1.309 \cdot 10^{16}$	14	$4.232 \cdot 10^{12} \pm 4.116 \cdot 10^{13}$
Random Non-Generalized Liu	1	$8.244 \cdot 10^{13} \pm 7.914 \cdot 10^{14}$	1	$5.982 \cdot 10^{13} \pm 7.195 \cdot 10^{14}$
Random Generalized Liu	17	$1.535 \cdot 10^{13} \pm 1.490 \cdot 10^{14}$	1	$2.621 \cdot 10^{12} \pm 2.255 \cdot 10^{13}$
MLGA	17	$5.761 \cdot 10^{14} \pm 5.630 \cdot 10^{15}$	1	$2.842 \cdot 10^{14} \pm 2.716 \cdot 10^{15}$

Table 4.9: FDIA Ranking by Highest Absolute and Mean MND

Technique	U_{max_abs}	φ_{max_abs}	P_{max_abs}	Q_{max_abs}	$\overline{U}$	$\overline{\varphi}$	$\overline{P}$	$\overline{Q}$
Random Data FDIA	2	2	1	3	2	2	1	3
Non-Generalized Liu FDIA	3	3	3	2	3	3	3	2
Generalized Liu FDIA	4	4	4	4	4	4	4	4
MLGA-FDIA	1	1	2	1	1	1	2	1

4.4.2 Results of FDIA with Power Flow based on State Estimation Results

To estimate the effects of the implemented FDIA techniques on grid health, we need to use a different approach of simulation, where the power flow calculation is conducted based on the values from the state estimation. In the standard simulation scenario, we would take the load data for each time step and do the power flow calculation based on this data. However, since the state estimation data is not integrated in such a way that it has an impact on the internal values of lines, buses, etc., an FDIA that impacts the state estimation in one time step would not impact the next time step whatsoever. This is why we conduct a second evaluation, which takes the results of the state estimation as a power flow calculation basis for the next time step. The results of this evaluation are shown in the following. For each technique, the simulation is run with 96 time steps to represent one whole day at 15-minute intervals. The results of the grid health tool and the deviation metric will be used to evaluate the results.

Random Data FDIA

Analogous to the first scenario, the random data FDIA was carried out using a newly calculated attack vector for each iteration. The shape of the plots for voltage magnitude and angle is similar to the first scenario. The highest absolute MND values are found at bus 25 for voltage magnitude and angle, at bus 1 for active power, and at bus 34 for reactive power. While bus 25 remains the most affected for the voltage measurements, the other two buses were not part of the most affected in the first scenario.

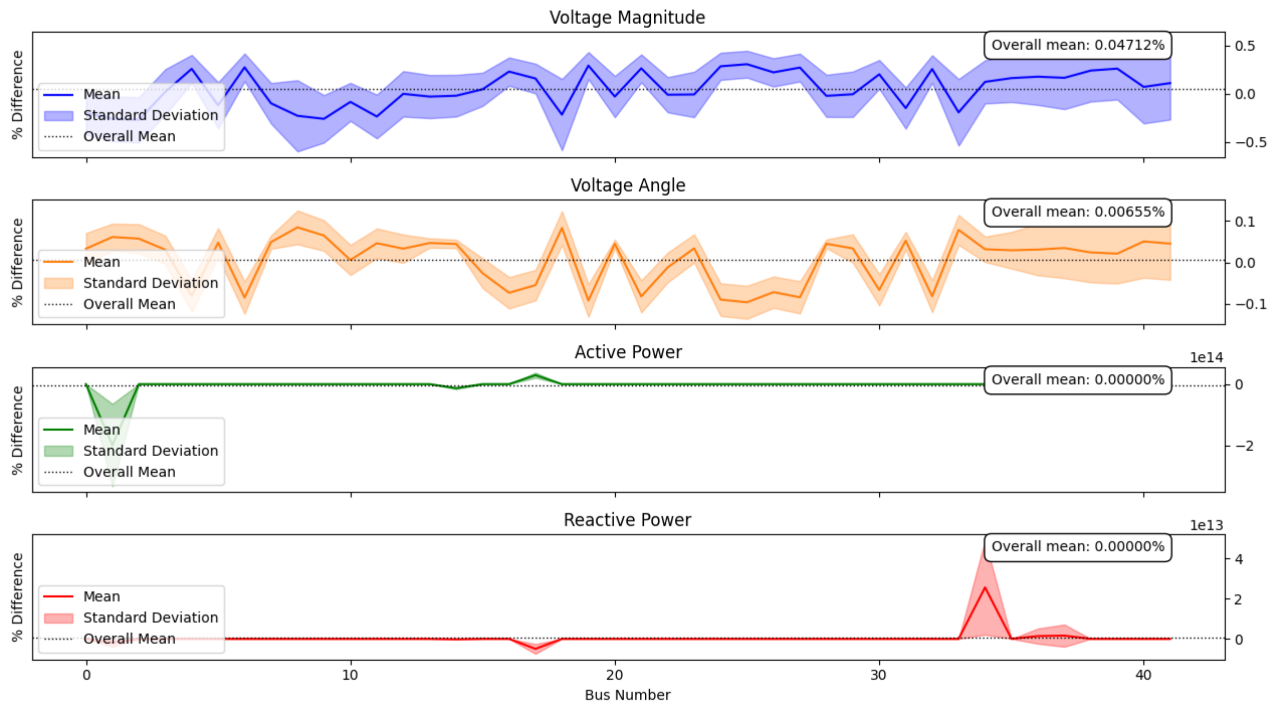

Figure 4.9: Results of Random Data FDIA

The means of the MND are slightly improved compared to the first scenario, although the order of magnitude does not change significantly. During all 96 time steps, the grid health tool did not record a single grid health violation.

Table 4.10: Means of MND (Random FDIA)

ΔU	$\Delta\varphi$	ΔP	ΔQ
$4.712 \cdot 10^{-2}$	$6.553 \cdot 10^{-3}$	$-3.595 \cdot 10^{12}$	$4.850 \cdot 10^{11}$

Random Non-Generalized Liu FDIA

Although the random non-generalized Liu FDIA was conducted in the same manner as in the first scenario, a stark difference can be observed in the lack of standard deviation zones in figure 4.10. This could be connected to the manner of calculation of the attack vector, but this cannot be said for certain at this point. The buses with the highest absolute MND are Bus 2 for the voltage angle and Bus 1 for all other measurements, as was the case in the first scenario.

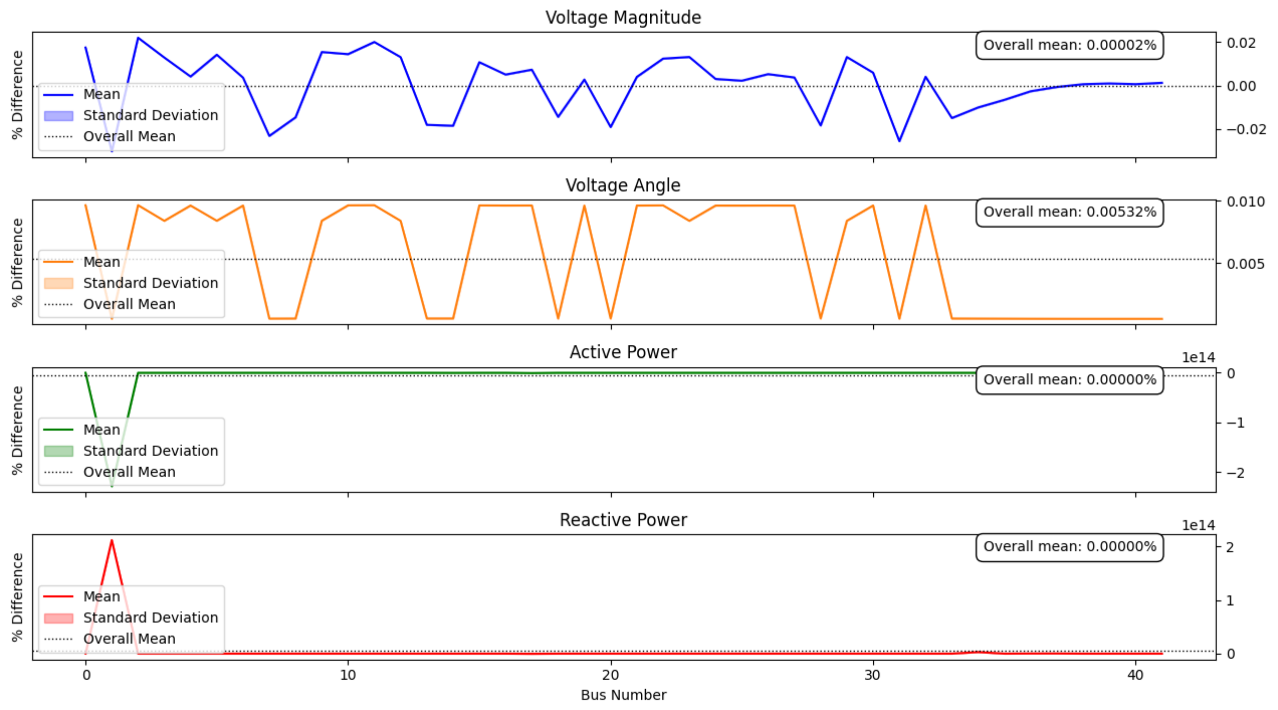

Figure 4.10: Results of Random Liu FDIA

Similar to the random data FDIA, the means of the MND change slightly in comparison to the first scenario but remain in the same order of magnitude. Like the random data FDIA, the random non-generalized Liu FDIA did not trigger any alarms from the grid health tool.

Table 4.11: Means of MND (Random Liu)

ΔU	$\Delta\varphi$	ΔP	ΔQ
$2.122799 \cdot 10^{-5}$	$5.322951 \cdot 10^{-3}$	$-5.203020 \cdot 10^{12}$	$4.895195 \cdot 10^{12}$

Random Generalized Liu FDIA

While the shape of the voltage measurements' plots remains largely the same for the random generalized Liu FDIA, similar to the random data FDIA the positions for the peaks in active and reactive power change during the second scenario. This is reflected in the highest absolute MND, where the bus for the voltage angle remains the same, bus 25, bus has the highest voltage magnitude value, and buses 1 and 34 have the highest active and reactive power values respectively.

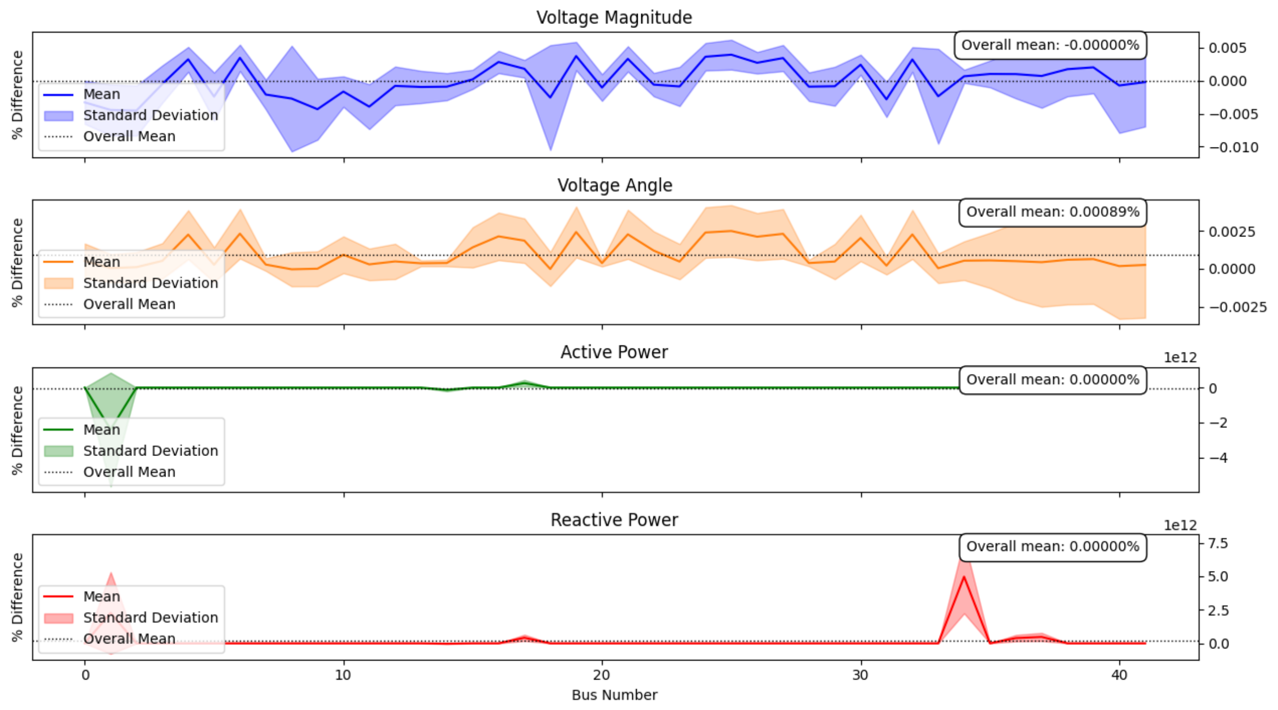

Figure 4.11: Results of Random Generalized Liu FDIA

Akin to the first two techniques, the means of the MND do not change significantly in magnitude and the grid health tool did not report any grid health violations. One possible reason for the techniques' surprisingly bad performance in light of the theoretical advantage compared to the non-generalized technique could be the calculation of the residual τ, which, when not calculated properly could result in tight constraints for finding an attack vector. This possibility should be studied in future works.

Table 4.12: Means of MND (Random Generalized Liu)

ΔU	$\Delta\varphi$	ΔP	ΔQ
$-4.958790 \cdot 10^{-7}$	$8.873958 \cdot 10^{-4}$	$-4.618380 \cdot 10^{10}$	$1.927707 \cdot 10^{11}$

MLGA-FDIA

Since the first scenario showed that the different calculated attack vectors did not change the outcome significantly, in the second scenario only one attack vector was created and used for each iteration. Where the plots of the first scenario showed standard deviations for the impact, no standard deviation was found in the second scenario, although the plots were both created using only one attack vector. This together with the plot for the random non-generalized FDIA suggests that in this scenario, a single attack vector has the same impact in each iteration. The highest absolute MND buses are the same in the second scenario as in the first, namely bus 18 for voltage magnitude, bus 40 for voltage angle, and bus 1 for reactive power, except for the active power, which was most influenced at bus 1 in the second scenario.

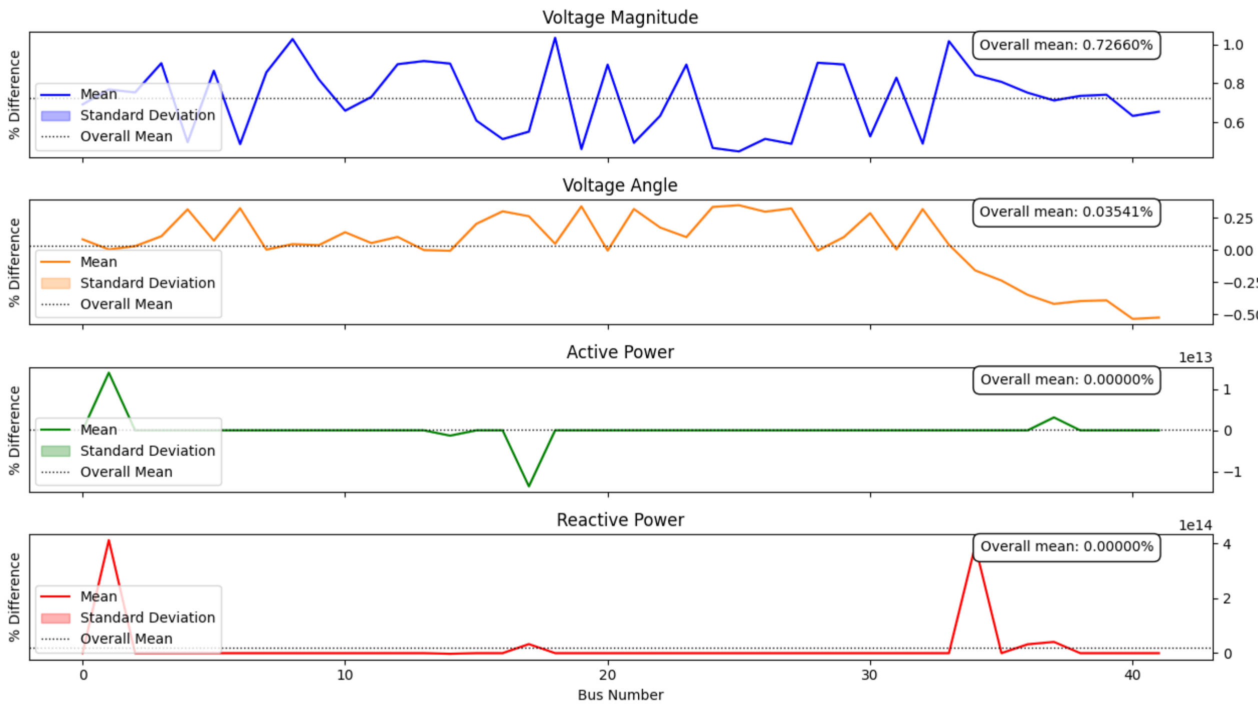

Figure 4.12: Results of MLGA-FDIA

As for the previous techniques, the magnitude of the means of the MND remains largely the same and the grid health tool was not triggered in any iteration.

Table 4.13: Means of MND (MLGA)

ΔU	$\Delta\varphi$	ΔP	ΔQ
$7.265988 \cdot 10^{-1}$	$3.541319 \cdot 10^{-2}$	$5.119004 \cdot 10^{10}$	$2.061555 \cdot 10^{13}$

Comparison of Techniques

Although the means of the MND did not change significantly, the ranking of the techniques according to the metrics has slightly different results in comparison to the first scenario. The primary metric, the grid health tool, was not triggered by any technique, which means no technique managed to overload a transmission line or the transformer. This result shows the impact of these FDIA techniques may not be as high as initially anticipated, though this cannot be said for certain with the available information.

An overview of the highest absolute MND can again be seen in the following two tables. The ranking according to this secondary metric can be found in tables 4.16. While the MLGA-FDIA is ranked one place lower in the active power category, overall it still performs the best. The random data FDIA also loses its first place in the active power category to the non-generalized Liu FDIA, which makes the ranking closer. However, the overall ranking remains the same as in the first scenario, with the MLGA-FDIA performing the best, the random data FDIA performing the second best, then the random non-generalized Liu FDIA, and the worst the random generalized Liu FDIA.

According to the tertiary metric of means of the MND, the overall ranking remains the same as for the secondary metric and the first evaluation scenario.

Table 4.14: Comparison of FDIA Techniques (Highest Absolute MND) - 1

Technique	**Success Rate**	**Bus**	ΔU	**Bus**	$\Delta \varphi$
Random	1.0	25	0.30584 ± 0.13942	25	0.09605 ± 0.03986
Random Non-Generalized Liu	1.0	1	0.03040 ± 0.00000	2	0.00966 ± 0.00000
Random Generalized Liu	1.0	2	0.00453 ± 0.00380	25	0.00248 ± 0.00170
MLGA	1.0	18	1.03529 ± 0.00000	40	0.53631 ± 0.00000

Table 4.15: Comparison of FDIA Techniques (Highest Absolute MND) - 2

Technique	**Bus**	ΔP	**Bus**	ΔQ
Random	1	$1.996 \cdot 10^{14} \pm 1.349 \cdot 10^{14}$	34	$2.555 \cdot 10^{13} \pm 2.351 \cdot 10^{13}$
Random Non-Generalized Liu	1	$2.286 \cdot 10^{14} \pm 0.00000$	1	$2.123 \cdot 10^{14} \pm 0.00000$
Random Generalized Liu	1	$2.404 \cdot 10^{12} \pm 3.264 \cdot 10^{12}$	34	$4.968 \cdot 10^{12} \pm 2.731 \cdot 10^{12}$
MLGA	1	$1.402 \cdot 10^{13} \pm 0.00000$	1	$4.120 \cdot 10^{14} \pm 0.00000$

Table 4.16: FDIA Ranking by Highest Absolute and Mean MND

Technique	U_{max_abs}	φ_{max_abs}	P_{max_abs}	Q_{max_abs}	$\overline{U}$	$\overline{\varphi}$	$\overline{P}$	$\overline{Q}$
Random Data FDIA	2	2	2	3	2	2	2	3
Non-Generalized Liu FDIA	3	3	1	2	3	3	1	2
Generalized Liu FDIA	4	4	4	4	4	4	3	4
MLGA-FDIA	1	1	3	1	1	1	4	1

5 Conlusions and Outlook

In this thesis, we first gave the motivation to show the relevance of the research field of FDIA and posed three research questions in chapter 1. After presenting the state of the art of research in the fields of power grid components, communication, and calculation, FDIA and simulations, as well as co-simulation of power grids and communication networks in chapter 2, requirements for the development of a co-simulation between the power grid and communication network were developed and simulation software to fulfill the requirement was designed and implemented in chapter 3. In chapter 4, we developed metrics to evaluate the impact of FDIA and implemented a selection of FDIA techniques, which were subsequently evaluated in two evaluation scenarios and ranked by their effectiveness.

In this section, we will first discuss the results of this work and deliver answers to the research questions posed in chapter 1. In the second part, an outlook and open questions as well as research gaps will be addressed.

5.1 Conclusions

The goal we set at the beginning of this thesis was to create a simulation framework to test and evaluate the impact of FDIA. This goal was divided into three research questions, which were answered in the scope of this work.

The research questions are:

RQ 1: How can a power grid and its communication infrastructure be simulated so that false data injection attacks and their impact on the grid can be tested and evaluated?

RQ 2: Which metrics can be used to evaluate the impact of false data injection attacks on the power grid?

RQ 3: What kinds of FDIA can be conducted on the power grid and how large is their impact on the power grid based on the metrics from RQ2?

To answer RQ 1, we first developed requirements for the successful simulation of FDIA on the power grid and communication infrastructure in section 3.1, where we used requirements from the technical guidelines for SMGWs regarding communication behavior, data format, and scalability, among others, which were complemented by self-developed requirements such as the requirements for the grid health tool. Those requirements were then developed into a simulation architecture, which is described in great detail in section 3.2. This simulation was then tested according to testing standards for unit and integration tests and utilized to facilitate the subsequent evaluation of FDIA techniques.

RQ 2 was answered mainly in the scope of section 4.1, where the grid health tool, the highest absolute mean nodal values, and the means of the mean nodal values were chosen as the three metrics for evaluating FDIA effects. Furthermore, the plotting functions provided in the same section provide a visual representation of the impact of FDIA, which makes the secondary and tertiary metrics easier to grasp. While the secondary and tertiary metrics could be successfully utilized in the evaluation process, the grid health tool can be improved in the future to provide more information even when no grid health violations occur.

RQ3 is formulated as a two-part question, whose first part was mainly answered in section 2.4. Though the list of possible FDIA techniques cannot be considered exhaustive, the provided taxonomy and example techniques gave a descriptive overview of the possible FDIA. A novel FDIA technique was developed in sections 2.4 and 4.2, showing the possibilities of creating FDIA techniques. Several FDIA techniques were tested with the simulation framework created for RQ1 and evaluated with the metrics developed for RQ2. The results show that modern techniques, such as the MLGA-FDIA, with machine learning techniques can outperform the traditional mathematic techniques as described by Liu et al. (2011) when first coining the term FDIA.

In summary, all three of the research questions posed at the beginning of this thesis could be answered, although all of them can and should be studied further in future works.

5.2 Outlook

While this thesis laid a foundation for the testing and evaluation of FDIA on power grids and their communication infrastructure, the research on this topic can be expanded in various directions:

While the developed simulation framework simulated a power grid as well as the respective communication network, both components were highly abstracted in the scope of this thesis. The power grid simulation could be extended regarding the system boundaries, e.g. by including the transformer or connecting different power grids in the same simulation. Moreover, additional measurement sources such as PMUs could be included in the simulation. Regarding the communication network simulation, the inclusion of encryption and communication channels in the simulation or the integration of hardware-in-the-loop devices could be conducted to extend the simulation framework, just to name a few.

Concerning the FDIA techniques, additional techniques could be implemented and developed to compare their effectiveness on the simulation framework. Moreover, the implemented techniques could be refined, e.g. the implementation of τ for the random generalized Liu FDIA, to increase their effectiveness. Since targeted FDIA techniques were not included in the scope of this work, the study of those could be of interest.

As previously mentioned, the metrics for the effect of FDIA can also be improved and extended. Especially the grid health tool, which in the current state is devised to be a sort of alarm tool, could be extended to be a real-time monitoring tool for the grid state.

Finally, while this thesis deals with the impact and simulation of FDIA, it does not consider how a GO could defend against them. Although there are studies on how to defend against FDIA, e.g. by utilizing PMUs to increase state estimation accuracy (Pei et al., 2020) or by using neural networks (Ayad et al., 2018), the simulation framework could be utilized directly to assess the effectiveness of defensive measures against FDIA.

Literature

PYPOWER: Solves power flow and optimal power flow problems, Mar. 2023. URL https://github.com/rwl/PYPOWER.

About MATPOWER – MATPOWER, May 2024. URL https://matpower.org/about/.

F. E. Abrahamsen, Y. Ai, and M. Cheffena. Communication Technologies for Smart Grid: A Comprehensive Survey. *Sensors*, Mar. 2021. URL https://www.mdpi.com/1424-8220/21/23/8087.

A. Abur and A. G. Expósito. *Power System State Estimation: Theory and Implementation*. CRC Press, Mar. 2004. ISBN 978-0-203-91367-3. Google-Books-ID: NQhbtFC6_40C.

H. Adjerid and A. R. Maouche. Multi-Agent system-based decentralized state estimation method for active distribution networks. *Computers & Electrical Engineering*, 86:106652, Sept. 2020. ISSN 0045-7906. doi: 10.1016/j.compeleceng.2020.106652. URL https://www.sciencedirect.com/science/article/pii/S0045790620305073.

M. Affenzeller, S. Wagner, S. Winkler, and A. Beham. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman and Hall/CRC, New York, Apr. 2009a. ISBN 978-0-429-14197-3. doi: 10.1201/9781420011326.

M. Affenzeller, S. Winkler, S. Wagner, and A. Beham. Simulating Evolution: Basics about Genetic Algorithms. In *Genetic Algorithms and Genetic Programming*. Chapman and Hall/CRC, 2009b. ISBN 978-0-429-14197-3. Num Pages: 23.

A. Ayad, H. E. Z. Farag, A. Youssef, and E. F. El-Saadany. Detection of false data injection attacks in smart grids using Recurrent Neural Networks. In *2018 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, Feb. 2018. doi: 10.1109/ISGT.2018.8403355. URL https://ieeexplore.ieee.org/document/8403355/?arnumber=8403355. ISSN: 2472-8152.

D. Baimel, S. Tapuchi, and N. Baimel. Smart grid communication technologies- overview, research challenges and opportunities. In *2016 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 116–120, June 2016. doi: 10.1109/SPEEDAM.2016.7526014. URL https://ieeexplore.ieee.org/document/7526014/?arnumber=7526014.

M. E. Baran and A. W. Kelley. State Estimation for Real-Time Monitoring of Distribution Systems. *IEEE Transactions on Power Systems*, 9(3):1601–1609, 1994. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=336098&casa_token=dhZwecZuc90AAAAA:tvqKCWtlViz2Ve6RbbR4oCGyN6RXQmuzBQ3igTzzgIC2KG7T4mFCFRrAPFnnLzMbSd9hmnXluuHN.

BSI. Errata für die BSI TR-03109-1 V 1.0.1 1 TAF 14 Hochfrequente Messwertbereitstellung für Mehrwertdienste, Dec. 2019. URL `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03109/TR-03109-1_Errata-TAF14.pdf?__blob=publicationFile&v=1`.

BSI. Technische Richtlinie (BSI-TR-03109-1), Sept. 2021. URL `https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Smart-metering/Smart-Meter-Gateway/TechnRichtlinie/TR-03109-1.html?nn=449854`.

H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, Oct. 2014. ISSN 0743-7315. doi: 10.1016/j.jpdc.2014.06.008. URL `https://www.sciencedirect.com/science/article/pii/S0743731514001105`.

G. Cheng, Y. Lin, A. Abur, A. Gómez-Expósito, and W. Wu. A Survey of Power System State Estimation Using Multiple Data Sources: PMUs, SCADA, AMI, and Beyond. *IEEE Transactions on Smart Grid*, 15(1):1129–1151, 2024. URL `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10153770`.

R. Czekster. *Analysis of selected frameworks in Smart Grid co-simulation*. Feb. 2021. doi: 10.13140/RG.2.2.18187.21288/2.

J. Day and H. Zimmermann. The OSI reference model. *Proceedings of the IEEE*, 71(12):1334–1340, Dec. 1983. ISSN 1558-2256. doi: 10.1109/PROC.1983.12775. URL `https://ieeexplore.ieee.org/document/1457043`. Conference Name: Proceedings of the IEEE.

M. Deru and A. Ndiaye. *Deep Learning mit TensorFlow, Keras und TensorFlow.js*. Apr. 2020. ISBN 978-3-8362-7427-2. URL `https://content-select.com/de/portal/media/view/668e600f-c738-4f89-bbd8-0488ac1b0005?forceauth=1`.

DIgSILENT. PowerFactory Features, July 2024. URL `https://www.digsilent.de/en/features.html`.

M. Emsalifalak, H. Nguyen, R. Zheng, L. Xie, L. Song, and Z. Han. A Stealthy Attack Against Electricity Market Using Independent Component Analysis. *IEEE Systems Journal*, 12(1):297–307, 2018. URL `https://ieeexplore.ieee.org/abstract/document/7307141?casa_token=bjDYM3rpswQAAAAA:7BVC-_TQoXdrxaNxx9XpbWMUmAjedd07vRTpsWuaogGdV4EiSsP7RftzqaLDzQQC7RYVMKZ0Z1Y`.

X. Fang, S. Misra, G. Xue, and D. Yang. Smart Grid — The New and Improved Power Grid: A Survey. *IEEE Communications Surveys & Tutorials*, 14(4):944–980, 2012. ISSN 1553-877X. doi: 10.1109/SURV.2011.101911.00087. URL `https://ieeexplore.ieee.org/document/6099519/?arnumber=6099519`. Conference Name: IEEE Communications Surveys & Tutorials.

D. L. Fernandes, A. L. M. Leopoldino, J. de Santiago, C. Verginis, A. A. Ferreira, and J. G. de Oliveira. Distributed control on a multi-agent environment co-simulation for DC bus voltage control. *Electric Power Systems Research*, 232:110408, July 2024. ISSN 0378-7796. doi: 10.1016/j.epsr.2024.110408. URL `https://www.sciencedirect.com/science/article/pii/S0378779624002967`.

FfE. Der Smart Meter Rollout in Deutschland und Europa, July 2023. URL https://www.ffe.de/veroeffentlichungen/smart-meter-rollout-in-deutschland-und-europa/.

C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe. Co-Simulation: A Survey. *ACM Computing Surveys*, 51(3):1–33, May 2019. ISSN 0360-0300, 1557-7341. doi: 10.1145/3179993. URL https://dl.acm.org/doi/10.1145/3179993.

A. Gougeon, F. Lemercier, A. Blavette, and A.-C. Orgerie. Influence of Communication Technologies in Smart Grid Power Congestion Management. In *2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 212–221, Aug. 2022. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00065. URL https://ieeexplore.ieee.org/abstract/document/9903123.

J. J. Grefenstette. Genetic algorithms and machine learning. *Proceedings of the sixth annual conference on Computational learning theory*, pages 3–4, 1993.

C. Hauser, D. Bakken, and A. Bose. A failure to communicate: next generation communication requirements, technologies, and architecture for the electric power grid. *IEEE Power and Energy Magazine*, 3(2):47–55, Mar. 2005. ISSN 1558-4216. doi: 10.1109/MPAE.2005.1405870. URL https://ieeexplore.ieee.org/document/1405870/?arnumber=1405870. Conference Name: IEEE Power and Energy Magazine.

ISO/IEC/IEEE. ISO/IEC/IEEE International Standard - Software and systems engineering –Software testing –Part 1:General concepts. *ISO/IEC/IEEE 29119-1:2022(E)*, pages 1–60, Jan. 2022. doi: 10.1109/IEEESTD.2022.9698145. URL https://ieeexplore.ieee.org/document/9698145/?arnumber=9698145. Conference Name: ISO/IEC/IEEE 29119-1:2022(E).

A. Jain and S. Bhullar. Network performance evaluation of smart distribution systems using smart meters with TCP/IP communication protocol. *Energy Reports*, 8:19–34, Nov. 2022. ISSN 2352-4847. doi: 10.1016/j.egyr.2022.05.108. URL https://www.sciencedirect.com/science/article/pii/S2352484722009568.

J. H. Kazmi, A. Latif, I. Ahmad, P. Palensky, and W. Gawlik. A flexible smart grid co-simulation environment for cyber-physical interdependence analysis. In *2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6, Apr. 2016. doi: 10.1109/MSCPES.2016.7480226. URL https://ieeexplore.ieee.org/document/7480226.

J. Kim, L. Tong, and R. J. Thomas. Subspace Methods for Data Attack on State Estimation: A Data Driven Approach. *IEEE Transactions on Signal Processing*, 63(5):1102–1113, Jan. 2015. URL https://ieeexplore.ieee.org/abstract/document/6996007.

M.-T. Le, T.-L. Nguyen, Q.-T. Tran, Y. Besanger, and T.-T. Hoang. A co-simulation approach for validating agent-based distributed algorithms in smart grid. In *2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON)*, pages 529–534, Palermo, Italy, June 2020. IEEE. ISBN 978-1-72815-200-4. doi: 10.1109/MELECON48756.2020.9140496. URL https://ieeexplore.ieee.org/document/9140496/.

R. Leal-Arcas, N. Akondo, and A. Rios. Energy Decentralization in the European Union Energy decentralization in the EU Energy decentralization in the European Union. Feb. 2019.

W. Li, M. Ferdowsi, M. Stevic, A. Monti, and F. Ponci. Cosimulation for Smart Grid Communications. *IEEE Transactions on Industrial Informatics*, 10(4):2374–2384, Nov. 2014. ISSN 1941-0050. doi: 10.1109/TII.2014.2338740. URL https://ieeexplore.ieee.org/document/6853359/?arnumber=6853359. Conference Name: IEEE Transactions on Industrial Informatics.

Y. Li and Y. Wang. False Data Injection Attacks With Incomplete Network Topology Information in Smart Grid. *IEEE Access*, 7:3656–3664, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2888582. URL https://ieeexplore.ieee.org/document/8581440/?arnumber=8581440. Conference Name: IEEE Access.

H. Liu and H. Yu. Decentralized state estimation for a large-scale spatially interconnected system. *ISA Transactions*, 74:67–76, Mar. 2018. ISSN 0019-0578. doi: 10.1016/j.isatra.2018.01.007. URL https://www.sciencedirect.com/science/article/pii/S0019057818300077.

L. Liu, M. Emsalifalak, Q. Ding, V. A. Emesih, and Z. Han. Detecting False Data Injection Attacks on Power Grid by Sparse Optimization. *IEEE Transactions on Smart Grid*, 5(2):612–621, 2014. URL https://ieeexplore.ieee.org/abstract/document/6740901.

Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 14(1):13:1–13:33, June 2011. ISSN 1094-9224. doi: 10.1145/1952982.1952995. URL https://dl.acm.org/doi/10.1145/1952982.1952995.

S. Meinecke, A. Klettke, D. Sarajlic, J. Dickert, M. Hable, F. Fischer, N. GmbH, M. Braun, A. Moser, and C. Rehtanz. GENERAL PLANNING AND OPERATIONAL PRINCIPLES IN GERMAN DISTRIBUTION SYSTEMS USED FOR SIMBENCH. 2019.

K. Mets, J. A. Ojea, and C. Develder. Combining Power and Communication Network Simulation for Cost-Effective Smart Grid Analysis. *IEEE Communications Surveys & Tutorials*, 16(3):1771–1796, 2014. ISSN 1553-877X. doi: 10.1109/SURV.2014.021414.00116. URL https://ieeexplore.ieee.org/document/6766918/?arnumber=6766918. Conference Name: IEEE Communications Surveys & Tutorials.

H. B. Netzer and P. Miller. What are race conditions? *ACM Letters on Programming Languages and Systems*, 1(1):74–88, 1992.

T. Overbye, X. Cheng, and Y. Sun. A comparison of the AC and DC power flow models for LMP calculations. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pages 9 pp.–, Jan. 2004. doi: 10.1109/HICSS.2004.1265164. URL https://ieeexplore.ieee.org/document/1265164.

C. Pei, Y. Xiao, W. Liang, and X. Han. PMU Placement Protection Against Coordinated False Data Injection Attacks in Smart Grid. *IEEE Transactions on Industry Applications*, 56(4):4381–4393, July 2020. ISSN 1939-9367. doi: 10.1109/TIA.2020.2979793. URL https://ieeexplore.ieee.org/document/9031416/?arnumber=9031416. Conference Name: IEEE Transactions on Industry Applications.

M. Pipattanasomporn, H. Feroze, and S. Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *2009 IEEE/PES Power Systems Conference and Exposition*, pages 1–8, Seattle, WA, USA, Mar. 2009. IEEE. ISBN 978-1-4244-3810-5. doi: 10.1109/PSCE.2009.4840087. URL `http://ieeexplore.ieee.org/document/4840087/`.

R. Rashed Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar. A survey on Advanced Metering Infrastructure. *International Journal of Electrical Power & Energy Systems*, 63:473–484, Dec. 2014. ISSN 0142-0615. doi: 10.1016/j.ijepes.2014.06.025. URL `https://www.sciencedirect.com/science/article/pii/S0142061514003743`.

H. T. Reda, A. Anwar, and A. Mahmood. Comprehensive survey and taxonomies of false data injection attacks in smart grids: attack models, targets, and impacts. *Renewable and Sustainable Energy Reviews*, 163:112423, July 2022. ISSN 1364-0321. doi: 10.1016/j.rser.2022.112423. URL `https://www.sciencedirect.com/science/article/pii/S1364032122003306`.

S. K. Salman. *Introduction to the Smart Grid: Concepts, Technologies and Evolution*. IET Digital Library, May 2017. ISBN 978-1-78561-120-9. doi: 10.1049/PBPO094E. URL `https://digital-library.theiet.org/content/books/po/pbpo094e`.

F. C. Schweppe and J. Wildes. Power System Static-State Estimation, Part I: Exact Model. *IEEE Transactions on Power Apparatus and Systems*, PAS-89(1):120–125, Jan. 1970. ISSN 0018-9510. doi: 10.1109/TPAS.1970.292678. URL `https://ieeexplore.ieee.org/document/4074022/?arnumber=4074022`. Conference Name: IEEE Transactions on Power Apparatus and Systems.

K. F. Schäfer. *Netzberechnung: Verfahren zur Berechnung elektrischer Energieversorgungsnetze*. Springer Fachmedien Wiesbaden, Wiesbaden, 2023. ISBN 978-3-658-40876-3 978-3-658-40877-0. doi: 10.1007/978-3-658-40877-0. URL `https://link.springer.com/10.1007/978-3-658-40877-0`.

S. Schütte, S. Scherfke, and M. Tröschel. Mosaik: A framework for modular simulation of active components in Smart Grids, 2011. URL `https://ieeexplore.ieee.org/abstract/document/6089027/`.

P. Sharma. Discrete-Event Simulation. 4(04), 2015.

V. Sharma. A Study on Data Scaling Methods for Machine Learning. *International Journal for Global Academic & Scientific Research*, 1(1):31–42, Feb. 2022. ISSN 2583-3081. doi: 10.55938/ijgasr.v1i1.4. URL `http://journals.icapsr.com/index.php/ijgasr/article/view/4`. Number: 1.

A. K. Singh and B. C. Pal. Decentralized Dynamic State Estimation in Power Systems Using Unscented Transformation. *IEEE Transactions on Power Systems*, 29(2):794–804, Mar. 2014. ISSN 1558-0679. doi: 10.1109/TPWRS.2013.2281323. URL `https://ieeexplore.ieee.org/document/6616007/?arnumber=6616007`. Conference Name: IEEE Transactions on Power Systems.

S. J. Steffel. Distribution grid considerations for large scale solar and wind installations. In *IEEE PES T&D 2010*, pages 1–3, Apr. 2010. doi: 10.1109/TDC.2010.5484387. URL `https://ieeexplore.ieee.org/document/5484387/?arnumber=5484387`. ISSN: 2160-8563.

L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, Nov. 2018. ISSN 1558-0679. doi: 10.1109/TPWRS.2018.2829021. URL `https://ieeexplore.ieee.org/document/8344496`. Conference Name: IEEE Transactions on Power Systems.

W. F. Tinney and C. E. Hart. Power Flow Solution by Newton's Method. *IEEE Transactions on Power Apparatus and Systems*, PAS-86(11):1449–1460, Nov. 1967. ISSN 0018-9510. doi: 10.1109/TPAS.1967.291823. URL `https://ieeexplore.ieee.org/document/4073219/?arnumber=4073219`. Conference Name: IEEE Transactions on Power Apparatus and Systems.

M. Viberg and B. Ottersten. Sensor array processing based on subspace fitting. *IEEE Transactions on Signal Processing*, 39(5):1110–1121, 1991. URL `https://ieeexplore.ieee.org/abstract/document/80966`.

M. Vrtal, J. Benedikt, R. Fujdiak, D. Topolanek, P. Toman, and J. Misurec. Investigating the Possibilities for Simulation of the Interconnected Electric Power and Communication Infrastructures. *Processes*, 10(12):2504, Dec. 2022. ISSN 2227-9717. doi: 10.3390/pr10122504. URL `https://www.mdpi.com/2227-9717/10/12/2504`. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

Y. Wang. Gauss–Newton method. *WIREs Computational Statistics*, 4(4):415–420, July 2012. ISSN 1939-5108, 1939-0068. doi: 10.1002/wics.1202. URL `https://wires.onlinelibrary.wiley.com/doi/10.1002/wics.1202`.

A. G. Wermann, M. C. Bortolozzo, E. Germano da Silva, A. Schaeffer-Filho, L. Paschoal Gaspary, and M. Barcellos. ASTORIA: A framework for attack simulation and evaluation in smart grids. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 273–280, Apr. 2016. doi: 10.1109/NOMS.2016.7502822. URL `https://ieeexplore.ieee.org/abstract/document/7502822?casa_token=xMBpJ9IwFGoAAAAA:CSBmz9AKf1ysGF4Fbg5_jCqRQHFeV9rF8BTZjbUn20opfm3_Lpc2YUWE5GRozQR4iPImWZOK`. ISSN: 2374-9709.

X. Yin, J. Zeng, and J. Liu. Forming Distributed State Estimation Network From Decentralized Estimators. *IEEE Transactions on Control Systems Technology*, 27(6):2430–2443, Nov. 2019. ISSN 1558-0865. doi: 10.1109/TCST.2018.2866556. URL `https://ieeexplore.ieee.org/abstract/document/8456677?casa_token=FwgkmBABTfcAAAAA:Cbp_XVh1gp4hWk6ICQsvAKmOOQKLRwrF3NwNUDHGMuPuoLNQ45dT3t_hAV2HEQl73j52v03srA`. Conference Name: IEEE Transactions on Control Systems Technology.

Z.-H. Yu and W.-L. Chin. Blind False Data Injection Attack Using PCA Approximation Method in Smart Grid. *IEEE Transactions on Smart Grid*, 6(3):1219–1226, May 2015. ISSN 1949-3061. doi: 10.1109/TSG.2014.2382714. URL `https://ieeexplore.ieee.org/document/7001709?denied=`. Conference Name: IEEE Transactions on Smart Grid.

J. Zhang, Z. Chu, L. Sankar, and O. Kosut. False data injection attacks on power system state estimation with limited information. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5, July 2016. doi: 10.1109/PESGM.2016.7741928. URL `https://ieeexplore.ieee.org/document/7741928?denied=`. ISSN: 1944-9933.

R. Zivanovic and C. Cairns. Implementation of PMU technology in state estimation: an overview. pages 1006–1011, Stellenbosch, South Africa, 1996. IEEE. doi: 10.1109/AFRCON.1996.563034. URL https://ieeexplore.ieee.org/abstract/document/563034.