Predicting Transformer Tap-Positions using Graph Neural Networks

For Distribution Power Grids

Soham Prajapati

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Predicting Transformer Tap-Positions using Graph Neural Networks

For Distribution Power Grids

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Soham Prajapati

July 16, 2025

Academic Supervisors:
Prof. Tamás Keviczky, Delft Center for Systems and Control
Prof. Jochen Cremer, Department of Electrical Sustainable Energy

Industrial Supervisors:
Dr. Nuran Cihangir Martin, Stedin B.V.
Willem van Seters, Stedin B.V.

Faculty of Mechanical Engineering (ME) · Delft University of Technology

The work in this thesis was supported by Stedin. Their cooperation is hereby gratefully acknowledged.

Abstract

Distributed energy resources challenge the situational awareness of power flows. Many distribution grid (DG) operators have not yet implemented state estimation (SE) due to the expense or privacy constraints of measurements that lead to an unobservable system, as well as inaccurate grid parameters. A key concern with the latter is the presence of medium- and low-voltage transformers with off-load tap changers, whose tap positions critically influence voltage levels across the network. Identifying transformers where the registered position is likely incorrect–and, when on-site verification is impractical, estimating a plausible tap setting–constitutes a valuable contribution to improving network observability and operational accuracy. Although operators could manually inspect each transformer, this is impractical–there are, for instance, up to 20,000 transformers in the Southern Netherlands.

To address these challenges, this thesis proposes a novel topology-aware framework for estimating state and transformer tap positions in unobservable DGs. The framework comprises two key components. First, a generative adversarial network (GAN) is used to train a generative model conditioned on the network topology and synthetic power flow data, generating realistic measurements. Second, an integrated model, referred to as the TapSEGNN model, is proposed for estimating state and transformer tap positions.

Both of these components employ a core model architecture which combines graph and simplicial complex neural networks to capture spatial dependencies between nodes, edges, and higher-order structures. To train these components, an industrial-grade data-processing pipeline was developed using a real DG topology and simulating the exact available measurement locations. The results show that balanced adversarial training of GAN accurately imputes the missing active power injection measurements, but produces high variance in imputations for voltage magnitude and active power flow measurements. The performance of the TapSEGNN model demonstrates at least tenfold higher accuracy for SE compared to conventional methods, and it predicts transformer tap positions with 100% accuracy in a computationally efficient manner. TapSEGNN also shows promising scalability for larger networks; however, it struggles with generalisability across similar real networks. Finally, the suboptimal performance of both components in certain aspects warrants further investigation, which is recommended as future work.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This thesis is more than just technical research for me. It is a personal journey of learning, growth, and collaboration. It would not have been possible without the generous support, guidance and on-time feedback from my supervisors, to whom I owe my sincerest gratitude.

First and foremost, I would like to express my gratitude to Professor Tamás Keviczky for directing me to Professor Jochen Cremer, with whom I pursued my interests in electrical power grids. Moreover, his astute observations and constructive criticism in many aspects of my work guided this work to completion.

I would also like to express my deepest gratitude to my supervisor from the Intelligent Electrical Power Grid department, Professor Jochen Cremer, who recommended me to Stedin to conduct this work as an industrial thesis. He encouraged me to assess my ideas critically, guided my brainstorming in the right direction, and supported me not only academically but also emotionally throughout this journey.

I would also like to extend my heartfelt thanks to my industry supervisors, Mr. Willem van Seters and Dr. Nuran Cihangir Martin, for their unwavering support throughout this project. Their thoughtful recommendations always pointed me to the right people for each challenge I faced. Their clear direction allowed me to navigate the large and complex organisation of Stedin.

A very special and deeply personal acknowledgement is owed to my late daily supervisor, Benjamin Habib, whom we sadly lost. He was the one who suggested that I experiment with simplicial complex neural networks for my work. His dedication to work until his last moments showed the strength of his character, which I will remember for my lifetime.

Finally, I would like to thank my family and friends, both here in the Netherlands and back home in India, for their constant support and, most importantly, for patiently listening to me blabber about power grids at every opportunity.

Delft, University of Technology — Soham Prajapati
July 16, 2025

પ્રભુ! મેં જે કંઇ કામ કર્યું છે તે તારી
શક્તિથી કર્યું છે.

હું તો નિમિત્ત માત્ર છું, પ્રભુ, કર્તૃત્વ તારું
છે.

Chapter 1

Introduction

1-1 Background and Motivation

One-third of electricity in the world comes from renewables, pushing the energy transition forward at an unprecedented pace to align with climate goals and sustainable practices [7]. The growing adoption of renewable energy sources (RES) by both producers and consumers is accelerating this transformation. Economically and socially, this transition offers opportunities for market growth and energy democratisation. While plugging the RES into the current grid marks a significant engineering achievement, it introduces several issues, such as higher generation, bidirectional power flows, and voltage violations, leading to grid congestion and potential blackouts. To address such problems, system operators require accurate network models to make informed decisions for planning and expanding the grid.

Tracing the origins of the power industry [8], under complex system operations or adverse conditions, the operator would have to manually analyse the situation and input the data into a power flow (PF) program to get results and insights into the problem. However, since the data were treated deterministically at that time, the results would most likely be unsatisfactory. With the advent of technological capability, the Energy Management System (EMS) of modern control centres allow operators to simulate, plan, operate, and monitor the power grid optimally using tools like state estimation (SE), PF analysis, contingency analyses, fault calculations, and voltage stability. Among these, SE precedes all other analyses in the pipeline. Since SE enables operators to closely monitor the underlying assets of the power system, preferably in real-time, numerous measurements are desired across the network. However, in general, the measurement infrastructure differs between the transmission grid (TG) and the distribution grid (DG) for any nation.

The TG manages extra-high and high voltage (EHV and HV) lines. It is considered the backbone of the power grid because it transports bulk electricity from the synchronised and asynchronous generation sources to the DG. The power fed to the DG gets distributed among the industrial and residential consumers through medium and low voltage (MV and LV) lines. Because of the critical role of the TG, any disturbances in it could cause a nationwide blackout,

affecting the lives of millions. Thus, the operational reliability standards of the TG always include "n-1" reserve: there is always a reserve for a single fault. Moreover, these regulations also ensure full observability of the network, i.e., the number of observations in the grid is sufficient to determine an accurate network model. However, these regulations do not fully apply to the DG. First, because the DG is not as critical as the TG, and second, it is significantly larger [6]. As a result, national investment in the DG measurement infrastructure is limited, leading to fewer measurements and often no fault reserves [6]. Table 1-1 gives an impression of the size of the DG as compared to the TG in the Netherlands. Comparing the medium voltage (MV) and low voltage (LV) networks in the DG, the MV network often has better observability as compared to the LV network, as consumer privacy contracts restrict access to the load profiles.

Table 1-1: Transmission and distribution networks in the Netherlands, 2009 [6].

Voltage level	Station type	Number of stations	Total Line Length
EHV (220/380 kV)	EHV/HV	Dozens	2685 km
HV (50/110/150 kV)	HV/MV	∼200	9252 km
MV (3-25 kV)	MV/LV	∼20,000	101,965 km
LV (0.4 kV)	LV/consumers	∼50-200	145,339 km

Despite these infrastructural differences, the integration of RES throughout the grid causes congestion in both the TG and the DG. For the TG, full observability allows well-established SE tools [9] to provide accurate network models for optimal decision-making. However, the same does not hold for the DG, where adapting SE methods designed for the TG is difficult due to limited observability. In addition to that, the quality of the network model is also susceptible to the input model parameters [10]. Conventional TG-based methods, such as residual analysis, cannot differentiate between a faulty measurement and one that results from inaccurate model parameters. This makes it challenging to deal with the limited observability in state and parameter estimation (SPE) for the DG.

Another challenge is dealing with the uncertainties in the grid caused by ageing infrastructure and cybersecurity concerns [11, 12]. Taking into account the current computational capabilities, this issue can be addressed with a data-driven approach for SPE, aligning with the philosophical framework of inductive reasoning [13]. Enhanced sensing capabilities create opportunities to rigorously test hypotheses against observable phenomena–scientific progress as a cycle of conjectures and refutations [13]. Data-driven or machine learning (ML) thrives with this vision by identifying and refining patterns through iterative hypothesis testing, where models are continuously updated to accommodate new evidence. Moreover, integrating synthetic data distribution and domain-specific physical models as prior knowledge further streamlines the training process. These priors constrain the search space for optimal functional mapping, leading to more efficient model convergence. The remarkable advancements in image generation with generative adversarial network (GAN) architecture serve as compelling evidence of the efficacy of machine learning approaches.

In light of the challenges above, this thesis proposes a novel ML framework to tackle the inherent limitations in the observability of the DG and to extend SPE capabilities for enhanced grid monitoring. Specifically, the proposed framework is built around graph neural network (GNN), a class of inductive neural networks that inherently encode the structural topology of the power grid. GNNs provide a computationally efficient and topology-aware solution that

is well-suited for learning from sparse and graph-structured data, such as that found in the power grids.

The framework applies the GNN architecture in two complementary stages. First, to mitigate the challenge of limited observability, the GNN is integrated within a GAN setup to infer missing measurements and reconstruct an otherwise unobservable or partially observable network model into a fully observable one. In the second stage, this reconstructed network model serves as an input for SPE, where the GNN operates in a supervised regression setting to accurately estimate the system states and required parameters. This two-stage framework highlights the adaptability and effectiveness of GNNs in addressing both observability and estimation challenges outlined earlier. In the next section, an industry-specific perspective of the challenges outlined above is delineated.

1-2 Industrial Motivation

A metaphor commonly used in the industrial perspective for power grid SE is [14],

"We are currently regulating traffic without knowing where the traffic is, SE will allow us to know where the traffic is."

Congestion is a significant issue worldwide, resulting in thousands of hours of lost time annually [15]. For example, the current congestion situation in the Dutch power grid is shown in Figure 1-1. Consequently, the grid operators are actively working to address this challenge by optimising and extending the capabilities of the existing infrastructure. In the Netherlands, among the six distribution system operators (DSOs), Stedin is conducting research to develop an SE toolbox for its DGs. One such study focused on a subnet of an MV network in the southern Netherlands, where conventional SE methods were applied. The choice to focus on a subnet–rather than the entire network–was driven by limited observability, which made full-network analysis impractical. For this isolated subnet, the conventional SE method called weighted least squares (WLS) was applied. The result of SE were erroneous, instead of a voltage drop along the feeder line–as expected due to line impedance and load–the estimation showed increasing voltage levels. Upon investigation, two possible issues were identified: poor data quality and incorrect input model parameters, particularly the tap position of the transformer.

The tap-position of a transformer, as illustrated in Figure 1-2, plays a critical role in regulating voltage levels across the DG. In HV/MV transformers, on-load tap changers are commonly used to dynamically adjust voltage in response to load variations. However, in MV/LV transformers with off-load tap changers, tap positions are often fixed and manually set. When these tap settings are not accurately known or reflected in the network model, it can lead to incorrect SE at various nodes within the LV network. This inaccuracy makes the LV network more susceptible to undetected voltage violations. For instance, the system might fail to recognise an overvoltage or undervoltage condition, or worse, misclassify it. As a result, operators may rely on faulty estimates and make misguided operational decisions. While operators could manually inspect and record tap settings at each transformer station, this is not practical. For example, as shown in Figure 1-3, some DGs in the southern Netherlands contain up to 500 transformers, making manual inspection infeasible for the operators.

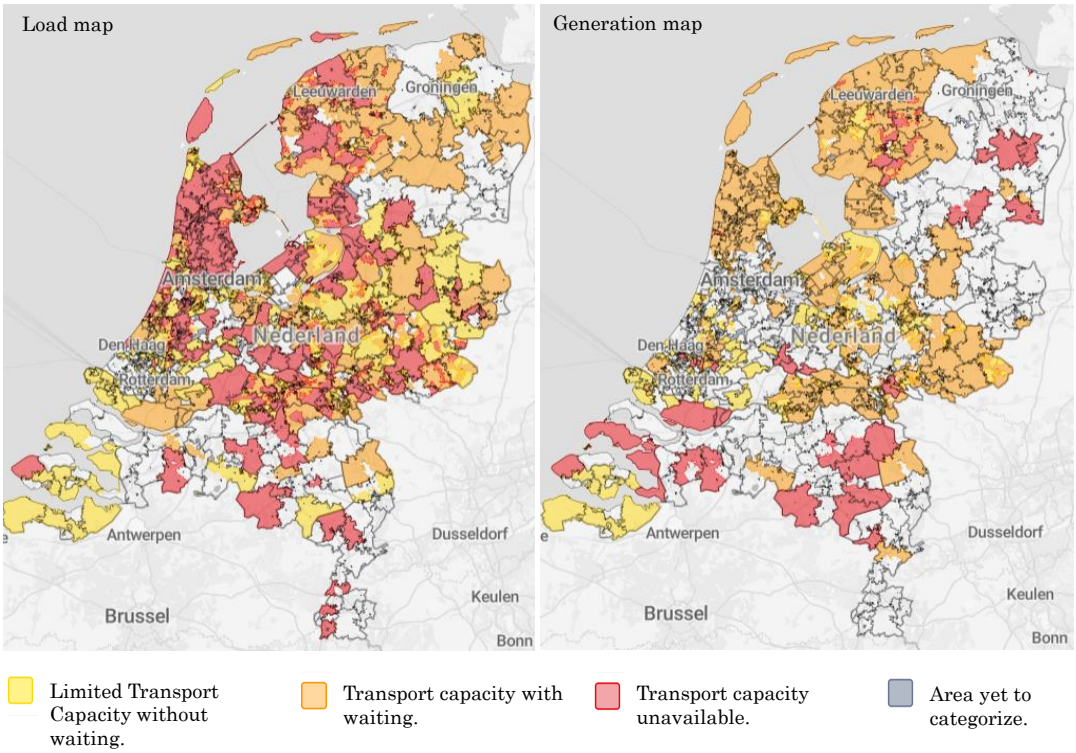


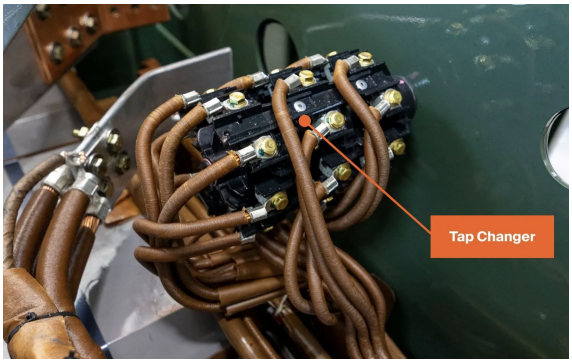
Figure 1-1: Regional congestion map for DG in the Netherlands (adapted from [1]). The load map indicates the grid capacity available for new registrations for power consumption at specific locations. The generation map shows transport capacity available for new registrations for distributed generation. Transparent areas denote locations with available transport capacity without any waiting in registration.

Another key aspect that influences this work is the type of measurements used for SE. Most of the MV networks today rely on data from Supervisory Control and Data Acquisition (SCADA) systems, which typically provide measurements averaged over 5-minute intervals [6]. As a result, transient behaviours are not captured because deploying advanced, high-frequency sensors like Phasor Measurement Unit (PMU), capable of tracking such dynamics, is prohibitively expensive. Consequently, the SE performed in industrial settings is focused on steady-state conditions. It computes static root mean square (RMS) values for either voltage, current, power injections or power flows, disregarding any transient dynamics or time-varying characteristics. This SE approach aligns with the limitations of SCADA data and prioritises operational efficiency over capturing rapid fluctuations. For these reasons, this work focuses on developing SE for static conditions, aiming to estimate the steady-state of the system.

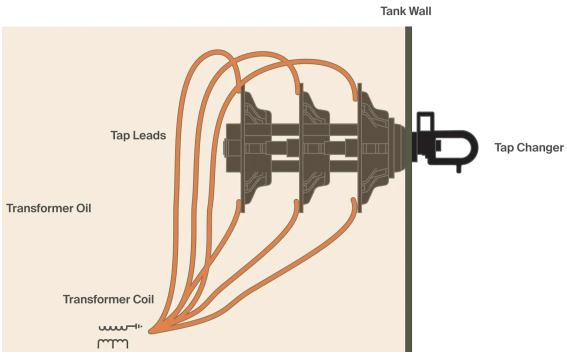
1-3 Research Question

The aspects mentioned above guide the careful selection of methodologies with strong potential for deployment in industrial settings. To systematically identify and evaluate existing work, address its limitations, and propose improved methodologies, it is essential to define a clear scope and research objective. The scope of this work is shown in Figure 1-4. Building

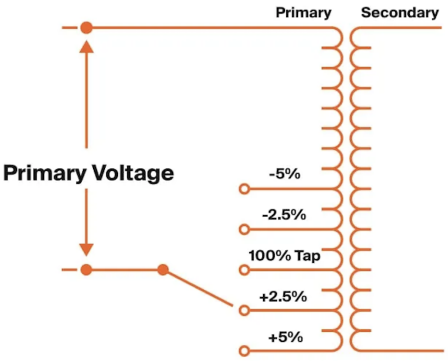
(a) Disconnected tap-changer showing rotary-switch to switch between 5-tap positions from A-E



(b) Tap changer connected with transformer winding using tap leads



(c) Cross-sectional view of tap-changer connected with the transformer winding



(d) Schematic view showing the working principle of tap-changers regulating secondary voltage by changing winding ratio

Figure 1-2: Illustration of tap-changer structure and operation (Courtesy: [2])

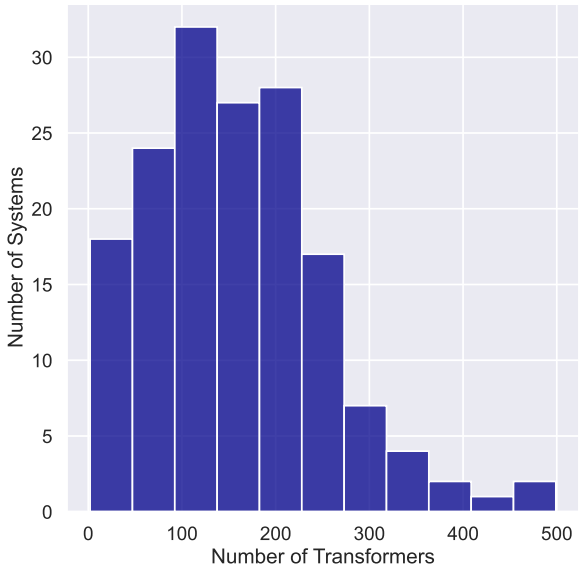


Figure 1-3: Histogram showing the distribution of the number of transformers in 160 MV/LV networks.

on top of this, the research objective is framed as the central research question that this work aims to answer.

How can graph-based methods effectively leverage topological information to perform generalizable, computationally efficient and joint state and transformer-tap position estimation in electric networks with limited observability?

By emphasising graph-based methods, the question targets solutions that take into account the topology of the power grid with sparse measurements. The properties of these methods, like parameter-sharing, shift-invariance, and permutation equivariance, contribute to the development of models that are both generalizable, scalable and computationally efficient.

1-4 Main Contribution

The main contributions of this work are summarised as follows,

1. **Industrial-Grade Data-Processing Pipeline**
   A scalable data-processing pipeline is designed to generate and sample synthetic data from practical DG topologies, tailored for static SE.
2. **Topology-Aware State and Parameter Estimation**
   A novel machine learning model, TapSEGNN, is developed using a hybrid node-centric graph neural network and edge-centric simplicial complex neural network architecture. It incorporates higher-order topological features, such as edge-triangle relations, for the accurate estimation of system states and parameters.
3. **Imputation of Missing Measurements with Adversarial Learning**
   A topology-aware generative adversarial network is proposed to impute missing critical measurements by implicitly learning the distribution of synthetic power flow results.

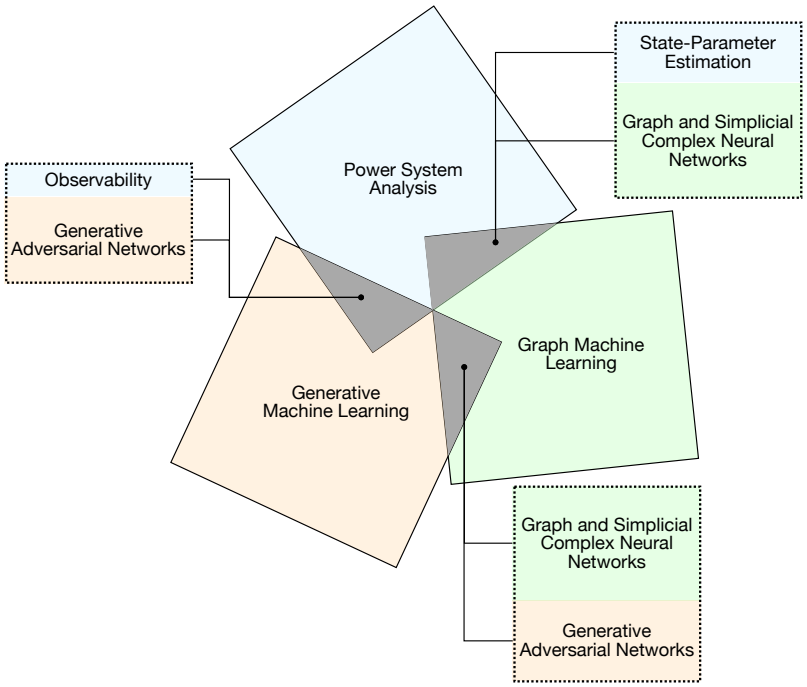


Figure 1-4: Scope of this work as an overlap of key research domains: power system analysis, generative machine learning and graph machine learning (Figure inspired by a covering problem [3])

The complete codebase for this work is publicly accessible via the GitHub repository at [16], accompanied by a detailed README file.

1-5 Report Overview

This thesis is structured into five chapters and two appendices that provide supplementary details. Following the introduction, Chapter 2 presents the theoretical background for power system SPE. It starts with the mathematical modelling of power system components for the PF problem, followed by a review of both conventional and emerging SPE methods. Due to the limitations posed by topological observability in the DGs, the chapter introduces topology-aware neural network architectures, including GNN and simplicial complex neural network (SCNN), which form the foundation of the proposed learning frameworks.

Chapter 3 outlines the objective of the methodology and details the core architecture built using GNN and SCNN. This architecture is further extended to support GAN and applied to the joint SPE task. Chapter 4 describes the scenario generation process and presents the main results, focusing on the performance of the GAN and SPE models. It also evaluates their scalability and generalisability aspects, as well as their comparison with baseline methods. Finally, Chapter 5 summarises the key findings, answers the central research question, and discusses the limitations and future directions of this work.

Chapter 2

Theoretical Foundations

In electrical power networks of sizes as large as those given in Table 1-1, it is essential for the operator to dispatch power optimally over large geographic distances, taking into account system losses. To accomplish this, alternating-current power flow (PF) calculations (hereafter referred to simply as PF) are necessary to build accurate network models that help maintain nominal power loading across the network. Therefore, this chapter establishes the foundational concepts to represent an accurate network model. It begins with the mathematical modelling of power system components and formulates the PF problem, which underpins traditional and modern state and parameter estimation (SPE) methods. The chapter then reviews conventional and emerging SPE techniques, particularly in the context of distribution grid (DG) with limited observability.

Recognising the limitations of the classical approaches, this chapter introduces topological observability as a critical concept and explores how power systems can be represented using graph structures and simplicial complexes. These representations enable the use of topology-aware neural networks. In particular, graph neural network (GNN) and simplicial complex neural network (SCNN) architectures embed physical topology directly into the learning architecture. By leveraging domain structure, these models enhance interpretability, scalability, and efficiency in distribution grid state and parameter estimation (DGSPE).

2-1 Mathematical Modelling of Power System

2-1-1 Phasor Notation

Phasor notation is required to denote the sinusoidal steady-state conditions in the power system for various quantities like voltage, current, power, and impedance. Consider root mean square (RMS) phasors for voltage and current with angular frequency ω as

$$\begin{aligned} \underline{V} &= |\underline{V}|e^{j(\omega t+\theta_v)}, \\ \underline{I} &= |\underline{I}|e^{j(\omega t+\theta_i)}, \end{aligned} \tag{2-1}$$

where $|\underline{V}|$ and $|\underline{I}|$ are the peak magnitude of voltage and current phasors, respectively, with the corresponding phase-shift angles θ_v and θ_i. Using these RMS phasors, time-average complex power injection $\underline{S}$ at a bus can be given as

$$\underline{S} = \underline{V}\ \underline{I}^* := P + jQ, \tag{2-2}$$

where P and Q denote the active and reactive power injection components [17]. Power delivered to the bus is considered positive, and vice versa. Similarly, the time-average complex power flow from bus i to bus j, $\underline{S}_{ij}$, can be shown as

$$\underline{S}_{ij} = \underline{V}_i \underline{I}^*_{ij} := P_{ij} + jQ_{ij}, \tag{2-3}$$

where P_{ij} and Q_{ij} are the active and reactive PF components. In further discussion, complex numbers will be denoted with an underline. The expressions in (2-2) and (2-3) will be discussed further in Section 2-1-6.

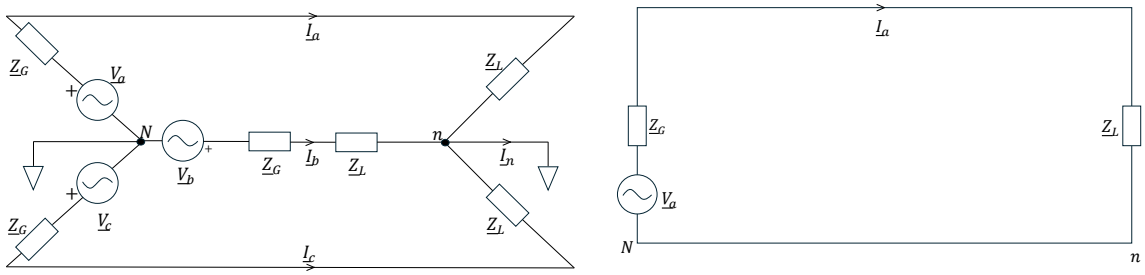
2-1-2 One-line Diagram

The practical networks consist of three-phase PF. The analysis of these networks becomes very complicated and cumbersome if they are depicted with three lines for each component, as shown in Figure 2-1a. For this reason, the power system community commonly uses per-phase equivalent models and one-line diagrams for power flow analysis (PFA) [18]. However, these simplified representations are valid only if the system is balanced, which necessitates meeting the following conditions in the power system:

- loads are equally distributed among all three phases of the system.
- no mutual inductance is present between the three-phase lines.
- neutral for all sources and loads are at the same potential.
- all network variables operate in the positive sequence, which is associated with normal operating conditions in the power system [17].

As shown in Figure 2-1, a single-phase equivalent can fully represent the balanced power system network. A single-phase diagram can be simplified further for PFA, which is called a one-line diagram. These diagrams only incorporate the components necessary for the problem under study. In PFA, for example, one-line diagrams do not show the circuit breakers but are required for network protection analysis [18]. Moreover, the standards for such representations can vary across different industries.

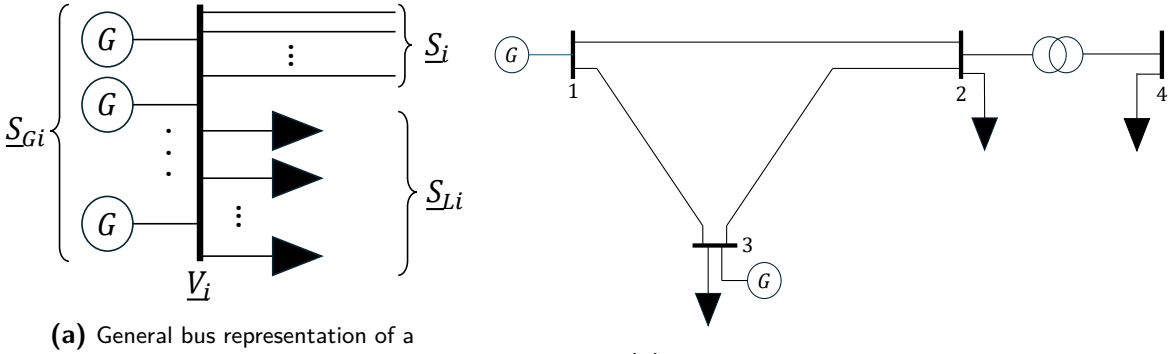
In the context of PFA and state estimation (SE) carried out in this study, a one-line diagram depicts the following components: 1) generators, 2) loads, 3) transformers, 4) power lines, 5) power stations. The power stations are modelled as buses or nodes connected to generators, loads, transmission lines or transformers–if any–as shown in Figure 2-2a. The transformers and transmission lines connect these buses as branches. A simple one-line diagram for a 4-bus system is given in Figure 2-2b.



(a) Three-phase representation

(b) Single-phase equivalent of the balanced three-phase network

Figure 2-1: Representations of a balanced power system network with three balanced sources $\underline{V}_a$, $\underline{V}_b$, and $\underline{V}_c$ with source impedance $\underline{Z}_G$ supplying to wye-connected balanced loads. For $\underline{I}_n = 0$ Figure 2-1a can be represented as Figure 2-1b.



(a) General bus representation of a power station

(b) One-line diagram of a 4-bus system

Figure 2-2: One-line diagram with detailed bus representation. In (a), the ith bus is injected with complex power $\underline{S}_{Gi}$ from generators, supplying complex power $\underline{S}_i$ to transmission lines and $\underline{S}_{Li}$ to loads. In (b), there are two generators located at buses 1 and 3; 3 loads at buses 2, 3, and 4; and a transformer connecting buses 2 and 4.

2-1-3 Per-Unit (pu) System

The per-unit (pu) system is a normalisation technique for power system analysis. It is used to express the voltage, current, and impedance across the network with respect to predefined base quantities, significantly simplifying calculations and system studies [17]. Moreover, this technique allows for narrowing down the wide range of values of certain quantities, offering clear relative insights and comparisons across the network. Using Figure 2-1b as an example, let the voltage distribution across the impedances $\underline{Z}_G$ and $\underline{Z}_L$ be noted as $|\underline{V}_G|$ and $|\underline{V}_L|$, respectively. Also, consider the RMS amplitudes of base voltage and base current be $|\underline{V}_B|$ and $|\underline{I}_B|$, so that per-unit quantities across the circuit can be calculated as

$$\underline{v}_{G/L} = \frac{\underline{V}_{G/L}}{\underline{V}_B}, \tag{2-4}$$

$$\underline{i}_a = \frac{\underline{I}_a}{\underline{I}_B}. \tag{2-5}$$

As a result, the per-unit impedance and RMS power can be given as

$$\underline{Z}_B = \frac{\underline{V}_B}{\underline{I}_B} \tag{2-6}$$

$$\implies \underline{z}_{G/L} = \frac{\underline{v}_{G/L}}{\underline{i}_a} = \frac{\underline{Z}_{G/L}}{\underline{Z}_B}$$

$$\underline{S}_B = \underline{V}_B \underline{I}_B. \tag{2-7}$$

It is important to note that, for networks with transformers, the network can be segmented into different sections by the transformers. Each section has its own base voltage. This way, the transformer is already processed in the base quantities, thus simplifying an ideal transformer to an ideal line. To simplify notation and prevent duplication of essential concepts in the forthcoming sections, the scalar per-unit quantities for voltage and current are represented by $|\underline{V}|$ and $|\underline{I}|$, respectively.

2-1-4 Two-port Network Representation

In the context of PFA, the per-phase equivalents of the transmission lines and transformers are represented using a two-port network model [18], as illustrated in Figure 2-3. This approach simplifies the component into a lumped model characterised by generalised circuit constants. The key assumptions underlying this representation are as follows:

- Passive: The component (transmission lines or transformers) does not generate energy. For instance, in the case of transmission lines, it is assumed that no generators are connected along the line.
- Linear: The component parameters, such as resistance, inductance, and capacitance, remain constant regardless of the amount of current flowing through them.
- Bilateral: These parameters do not depend on the direction of the current flow.

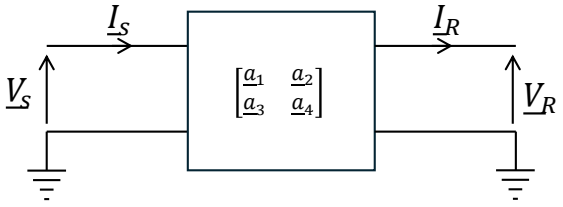


Figure 2-3: Two-port representation with $\underline{a}_1, \underline{a}_2, \underline{a}_3$ and $\underline{a}_4$ as the generalised circuit constants which are scalar complex numbers representing a circuit admittance. Subscripts S and R represent sending and receiving terminals.

In matrix form, the two-port representation can be given as

$$\begin{bmatrix} \underline{V}_S \\ \underline{I}_S \end{bmatrix} = \begin{bmatrix} \underline{a}_1 & \underline{a}_2 \\ \underline{a}_3 & \underline{a}_4 \end{bmatrix} \begin{bmatrix} \underline{V}_R \\ \underline{I}_R \end{bmatrix}. \tag{2-8}$$

It should be noted that the constants $\underline{a}_1, \underline{a}_2, \underline{a}_3$ and $\underline{a}_4$ are complex pu values and satisfy the following relationship:

$$\underline{a}_1 \ \underline{a}_4 - \underline{a}_2 \ \underline{a}_3 = 1. \tag{2-9}$$

For PF studies, π-representation is used for transmission lines and transformers. A general tap-changing and phase-shifting transformer model used for PFA is shown in Figure 2-4 using a nominal-π representation. The tap positions in the transformers are used to control the winding ratio. This allows control of the voltage at the tapped side of the transformer. Phase-shifting can control the active power injection into the transmission line, preventing issues like uncontrolled PFs and uneven transmission line loading [19]. The generalisation constant matrix for transformers can be given by

$$\begin{bmatrix} \underline{I}_f \\ \underline{I}_t \end{bmatrix} = \begin{bmatrix} \left(\underline{y}_s + j\frac{b_c}{2}\right)\frac{1}{\underline{N}\,\underline{N}^*} & -\underline{y}_s\frac{1}{\underline{N}^*} \\ -\underline{y}_s\frac{1}{\underline{N}} & \underline{y}_s + j\frac{b_c}{2} \end{bmatrix} \begin{bmatrix} \underline{V}_f \\ \underline{V}_t \end{bmatrix}, \tag{2-10}$$

where $\underline{N} = \tau e^{j\delta}$ is the complex-tap τ ratio of the transformer. For this work, the phase-shift angle $\delta = 0$ to reflect a common assumption in DGs where transformers are not used for phase-shifting but only voltage magnitude regulation.

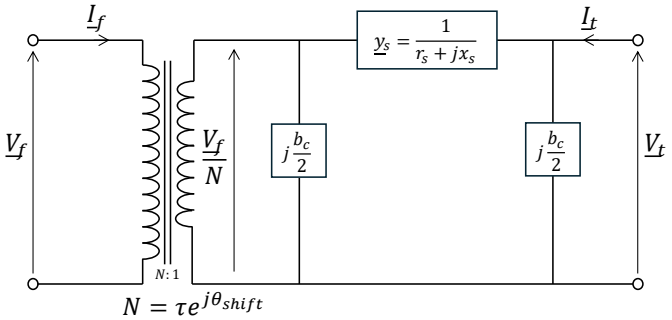


Figure 2-4: Impedance diagram of a general phase-shifting tap-changing transformer. The fixed terminal of the transformer has voltage $\underline{V}_f$ with entering current $\underline{I}_f$. The series admittance $\underline{y}_s$ (reciprocal of impedance) is at the tapped side of the transformer with magnetising susceptance $j\frac{b_c}{2}$ accounting for all leakage and winding losses. The tap-position is given by τ and phase-shift by θ_{shift}.

Transmission line models can be classified based on their lengths, as outlined below,

- Short lines: For transmission lines less than 100 km long, the shunt admittance ($\underline{y}_{sh}$) can be neglected because the voltages are not very high; as a result, the capacitive effects are minimal.
- Medium lines: For lines with lengths between 100 km and 250 km, nominal-π representation is used. The term "nominal" indicates that the line parameters can be considered lumped, providing an accurate approximation for most practical purposes. This simplification is widely adopted in PFA and is discussed here.

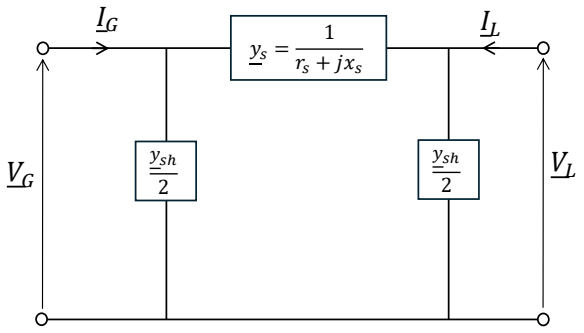


Figure 2-5: Nominal-π of a transmission line. The subscripts G and L depict the generator and load side of the line, respectively. Series and shunt admittance given by $\underline{y}_s$ and $\underline{y}_{sh}$ respectively.

- Long lines: For lines whose lengths are more than 250 km, an equivalent-π representation is employed. Unlike the nominal-π model, this approach treats the line parameters as uniformly distributed along their length. First-order differential equations give equivalent descriptions for the generalised circuit constants of these lines.

The nominal-π representation model for transmission line is illustrated in Figure 2-5. The generalisation circuit constants for it can be given by

$$\begin{bmatrix} \underline{I}_G \\ \underline{I}_L \end{bmatrix} = \begin{bmatrix} \underline{y}_s + \frac{\underline{y}_{sh}}{2} & -\underline{y}_s \\ -\underline{y}_s & \underline{y}_s + \frac{\underline{y}_{sh}}{2} \end{bmatrix} \begin{bmatrix} \underline{V}_G \\ \underline{V}_L \end{bmatrix}. \tag{2-11}$$

2-1-5 Construction of Bus Admittance Matrix

In the context of network representation shown in Figure 2-2, the bus admittance matrix ($\mathbf{Y}_{\text{bus}}$) is used to describe the connectivity of transmission lines and transformers with the buses in the network. Each entry in this matrix is a complex value representing the generalised circuit constants derived from the two-port network model discussed in Section 2-1-4. For instance, these two equations (2-10) and (2-11) represent a 2×2 bus admittance matrix that models the connection between two buses. A generalized construction of the $\mathbf{Y}_{\text{bus}}$ can be given by Figure 2-6.

Using Kirchhoff's circuit laws,

$$\begin{aligned} \underline{I}_i &= \sum_{\substack{k=1 \\ k \neq i}}^{n} \left(\underline{I}_{s,ik} + \underline{I}_{sh,ik} \right) \\ &= \sum_{\substack{k=1 \\ k \neq i}}^{n} \left(\underline{y}_{s,ik} (\underline{V}_i - \underline{V}_k) + \underline{y}_{sh,ik} \underline{V}_i \right) \\ &= \underline{V}_i \left(\sum_{\substack{k=1 \\ k \neq i}}^{n} \left(\underline{y}_{s,ik} + \underline{y}_{sh,ik} \right) \right) - \sum_{\substack{k=1 \\ k \neq i}}^{n} \underline{y}_{s,ik} \underline{V}_k, \end{aligned} \tag{2-12}$$

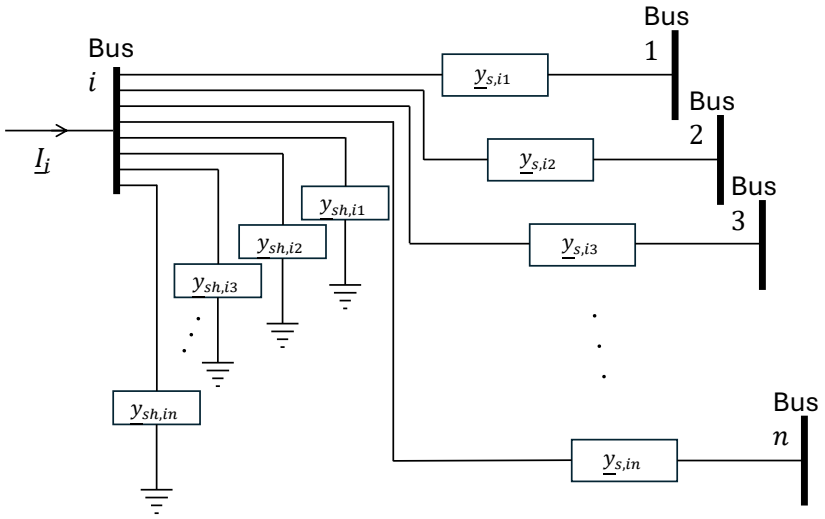


Figure 2-6: Bus i connecting to n other buses

where, the injection current at bus i is divided among the branches flowing through the series impedance $\underline{y}_{s,ik}$ and shunt impedance $\underline{y}_{sh,ik}$ as $\underline{I}_{s,ik}$ and $\underline{I}_{sh,ik}$, respectively. First term represents the diagonal elements describing the relation between $\underline{I}_i$ and $\underline{V}_i$. The second term represents the off-diagonal elements relating $\underline{I}_i$ with the voltage magnitude of all other buses $\underline{V}_k, \forall k \neq i$, thus

$$\underline{Y}_{ii} = \left(\sum_{\substack{k=1 \\ k \neq i}}^{n} \left(\underline{y}_{s,ik} + \underline{y}_{sh,ik)} \right) \right),$$
$$\underline{Y}_{ik} = -\underline{y}_{s,ik}, \quad \forall k \neq i,$$

so that,

$$\underline{I}_i = \sum_{k=1}^{n} \underline{Y}_{ik} \underline{V}_k. \tag{2-13}$$

Now, (2-13) can be given for all such buses, writing in Ohm's law as

$$\begin{bmatrix} \underline{Y}_{11} & \underline{Y}_{12} & \cdots & \underline{Y}_{1n} \\ \underline{Y}_{21} & \underline{Y}_{22} & \cdots & \underline{Y}_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \underline{Y}_{n1} & \underline{Y}_{n2} & \cdots & \underline{Y}_{nn} \end{bmatrix} \begin{bmatrix} \underline{V}_1 \\ \underline{V}_2 \\ \vdots \\ \underline{V}_n \end{bmatrix} = \begin{bmatrix} \underline{I}_1 \\ \underline{I}_2 \\ \vdots \\ \underline{I}_n \end{bmatrix}$$
$$\text{or} \quad \mathbf{Y}_{\text{bus}} \mathbf{V}_{\text{bus}} = \mathbf{I}_{\text{bus}}, \tag{2-14}$$

where $\mathbf{Y}_{\text{bus}} \in \mathbb{C}^{n \times n}$, $\mathbf{I}_{\text{bus}}$, $\mathbf{V}_{\text{bus}} \in \mathbb{C}^{n \times 1}$ for network with n buses in the network.

2-1-6 Power Flow Problem

PF problem involves determining the unknown variables across the buses of the single-phase equivalent power system network based on known variables, using a set of nonlinear equations that describe the relationships between these variables. Once all the values are determined, the PF problem is considered fully solved: the RMS values of the quantities in the system are completely defined. It does not account for any transient effects in these quantities and, therefore, is considered static. The PF equations govern the relationships between the four variables considered at each bus: voltage magnitude ($|\underline{V}|$), voltage phase-shift angle (θ which is equal to θ_v in Section 2-1-1), active power injection (P) and reactive power injection (Q), derived from Section 2-1-1 as

$$\underline{V}_i = \frac{1}{\underline{Y}_{ii}} \left(\frac{P_i - jQ_i}{\underline{V}_i^*} - \sum_{\substack{k=1 \\ k \neq i}}^{n} \underline{Y}_{ik} \underline{V}_k \right), \tag{2-15}$$

$$\theta_i = \arg(\underline{V}_i), \tag{2-16}$$

$$P_i = |\underline{V}_i| \sum_{k=1}^{n} |\underline{V}_k| \left(g_{ik} \cos\theta_{ik} + b_{ik} \sin\theta_{ik} \right), \tag{2-17}$$

and

$$Q_i = |\underline{V}_i| \sum_{k=1}^{n} |\underline{V}_k| \left(g_{ik} \sin\theta_{ik} - b_{ik} \cos\theta_{ik} \right), \tag{2-18}$$

where, $g_{ik} = \mathrm{Re}(\underline{Y}_{ik})$ and $b_{ik} = \mathrm{Im}(\underline{Y}_{ik})$ from the $\mathbf{Y}_{\text{bus}}$ matrix, and $\theta_{ik} = \theta_i - \theta_k$. Moreover, the complex power injection can be denoted as $\underline{S}_i = P_i + jQ_i$.

Essentially, for a network with n buses, there will be $4n$ bus variables for the PF problem. At each of these buses, if two variables are specified, the remaining $2n$ variables can be determined by solving the $2n$ PF equations. Assigning the known variables at every bus depends on the devices connected to that bus. In physical networks, [17], there are three specific combinations giving rise to three types of buses:

1. PQ Bus: At these buses, the known quantities are real power $-P_i$ and reactive power $-Q_i$ ($-ve$ sign implies power is drawn from the bus). For this bus, the objective of the PFA is to determine $|\underline{V}_i|$ and θ_i.

2. PV Bus: These buses have input real power P_i and voltage $|\underline{V}_i|$ specified[1]. For these buses, PFA determines θ_i and Q_i.

[1] Specified as RMS values

3. Slack or Reference Bus: This bus specifies the phase angle (chosen as $\theta_i = 0°$) as a reference for all other buses and the voltage (specified one p.u.). The bus connected with the largest generator in the network is considered the slack bus, because it is required that the variations in both active and reactive power due to I^2R losses are the minimal percentage of the generator's overall capacity. Buses connected with small generators could lead to instability or overloading. PFA uses this bus as the reference for determining the angles in all other buses.

The definition of the state vector in this problem depends on the solution method employed. Extensive literature is available on solving this problem using numerical techniques, data-driven black-box approaches, and model-based methods. To provide a foundational understanding of the PF problem, the most popular iterative solution methods: Gauss-Seidel (GS) and Newton-Raphson (NR) are discussed in Appendix A. Since, PFA serves as the foundation for measurement model in SE, an SE solver can solve the PF problem, but a PF solver will be unable to address a nontrivial SE problem [14]. The following section will discuss SE methods.

2-2 State and Parameter Estimation Methods

Formally, let $\mathcal{B}$ represent the set of buses in the network with number of buses $|\mathcal{B}| = n$. Bus 1 can be the slack bus and the remaining $n-1$ PQ buses. Moreover, consider the branch set $\mathcal{U} \subseteq \mathcal{B} \times \mathcal{B}$ with $|\mathcal{U}| = d$ denoting the distribution lines and transformers. For n buses and d branches, the primary goal of SE is to determine the voltage magnitude $|\underline{V}|$ and voltage phase angle θ as states for all buses in the network. It is important to recall from Section 2-1-6 that one bus in the network specifies the phase angle $\theta_1 = 0°$ as a reference for other buses. Therefore, SE requires determining only unknown $2n-1$ states with state-vector $\mathbf{u} \in \mathbb{R}^{2n-1}$ defined as,

$$\mathbf{u} = [\theta_2, \theta_3, ..., \theta_n, |\underline{V}|_1, |\underline{V}|_2, ..., |\underline{V}|_n]. \tag{2-19}$$

The determination of the states for the entire network allows for the calculation of other quantities like complex power injections $P_i + jQ_i$ at buses and power flows $P_{ij} + jQ_{ij}$ in the branches. Additionally, using the notation in above sections, the network branch parameters considered for the PF problem here[2] for branch ij will be $[\mathrm{Re}(\underline{y}_{s,ij}), \mathrm{Im}(\underline{y}_{s,ij}), \mathrm{Re}(\underline{y}_{sh,ij}), \mathrm{Im}(\underline{y}_{sh,ij}), \tau_{ij}]$ or $[r_{ij}, x_{ij}, g_{ij}, b_{ij}, \tau_{ij}]$. So, the parameter-vector $\mathbf{p} \in \mathbb{R}^{d\times 5}$ with kth row corresponding to the branch connecting bus i and j, i.e., $[\mathbf{p}]_k = [r_{ij}, x_{ij}, g_{ij}, b_{ij}, \tau_{ij}]$.

Using the above definitions, the measurement model for the SE can be given as

$$\mathbf{z} = \boldsymbol{g}(\mathbf{u}) + \mathbf{e}, \tag{2-20}$$

where, $\mathbf{z} \in \mathbb{R}^m$ is the measurement vector, $\mathbf{u}$ is the state-vector, $\boldsymbol{g} : \mathbb{R}^{2n-1} \to \mathbb{R}^m$ is the nonlinear vector-valued function relating measurements in $\mathbf{z}$ to state $\mathbf{u}$ and $\mathbf{e} \in \mathbb{R}^m$ is the

[2]Phase-shifting angle $\delta = 0$ for all transformers.

measurement-error vector with arbitrary statistical properties (e.g., Gaussian distribution). Various SE methods introduce their own $\boldsymbol{g}$ to effectively[3] minimise the difference between $\mathbf{z}$ and $\boldsymbol{g}(\mathbf{u})$. This formulation can be extended for parameter estimation in three ways: sequential, joint, or disjoint [14] approaches.

- **Sequential Approach**: SE is performed first, followed by parameter estimation (PE), avoiding any changes to the SE algorithms. Therefore, the measurement model for SE stays the same here.
- **Joint Approach**: SE and PE are solved simultaneously, modifying the measurement model as

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{z}_p \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}(\mathbf{u}, \mathbf{p}) \\ \boldsymbol{g}_p(\mathbf{u}, \mathbf{p}) \end{bmatrix} + \begin{bmatrix} \mathbf{e} \\ \mathbf{e}_p \end{bmatrix}, \tag{2-21}$$

  where, $\mathbf{z}_p$ contains the suspected parameter values from the database, $\boldsymbol{g}_p$ contains the expressions (mostly linear) having *a priori* mathematical model about the parameter vector [20] and $\mathbf{e}_p$ denotes the error-vector corresponding to $\mathbf{z}_p$.
- **Disjoint Approach**: PE is conducted independently of SE, without relying on the concepts of SE [21].

For transformer tap-position estimation, sequential and joint approaches based on SE concepts are more suitable than disjoint methods. Disjoint methods for individual equipment measurements, while potentially more accurate, are generally impractical for networks with numerous equipment following the arguments in Section 1-2, specifically, Figure 1-2.

2-2-1 Conventional Methods

For the measurement model given in (2-20), the function $\boldsymbol{g}$ consists of the PF equations mapping states to the measurement. The goal of SE is to solve for the most likely $\mathbf{u}$ such that it minimizes the residual vector ($\mathbf{r} = \mathbf{z} - \boldsymbol{g}(\mathbf{u})$). A method widely used in statistics for this is maximum likelihood estimation (MLE), whose objective is to maximise the likelihood function that represents the joint probability distribution of all the measurements as a function of the state vector. In MLE, assuming Gaussian-distributed errors, the negative log-likelihood function is equivalent to the sum of squared residuals, corresponding to weighted least squares (WLS) [9].

Newton and Quasi-Newton methods are conventional numerical techniques for solving this nonlinear WLS problem. These methods rely on the assumptions that the error vector is independent (covariance matrix $\mathbb{E}(\mathbf{e}\mathbf{e}^\top)$ is diagonal) and identically distributed (sampled from a zero-mean Gaussian distribution $\mathbb{E}(\mathbf{e}) = \mathbf{0}$). Under these assumptions, the objective function boils down to a nonlinear WLS problem given as

$$\min_{\mathbf{u}} \quad W(\mathbf{u}) = \mathbf{e}^\top \mathbf{R}^{-1} \mathbf{e}, \tag{2-22}$$

[3] depends on the type of optimisation problem and underlying assumptions

where the covariance matrix $\mathbf{R} = E(\mathbf{e}\mathbf{e}^\top) = \text{diag}(\sigma_1^2, \sigma_2^2, ..., \sigma_m^2) \in \mathbb{R}^{m\times m}$ with σ_i as the standard deviation associated to measurement i. Since $\boldsymbol{g}(\cdot)$ is a multivariate nonlinear vector-valued function, this problem can be classified as a nonlinear, non-convex, unconstrained minimisation problem. From a systems and control perspective, gradient-based and gradient-free methods solve this problem. The most widely adopted methods are gradient-based. For (2-22), the first derivative can be given by [22]

$$\begin{aligned} \nabla^\top W &= \mathbf{0} \\ \implies \mathbf{J}^\top \mathbf{R}^{-1}\mathbf{e} &= \mathbf{0}, \end{aligned} \tag{2-23}$$

with Jacobian matrix $\mathbf{J} = \left[\frac{\partial \boldsymbol{g}(\mathbf{u})}{\partial \mathbf{u}}\right]$. Now, this is equivalent to solving $\mathbf{x}^1(\mathbf{u}) = 0$ with Newton's method in Section A-0-2 giving rise to the following iteration:

$$\begin{aligned} \nabla \left(\mathbf{J}^\top \mathbf{R}^{-1}\mathbf{e}\right)_{\mathbf{u}=\mathbf{u}^{(k)}} \Delta\mathbf{u}^{(k)} &= -\left(\mathbf{J}^\top \mathbf{R}^{-1}\mathbf{e}\right)_{\mathbf{u}=\mathbf{u}^{(k)}} \\ \implies \left(-\mathbf{J}^\top \mathbf{R}^{-1}\mathbf{J} + \mathcal{A}\right)_{\mathbf{u}=\mathbf{u}^{(k)}} \Delta\mathbf{u}^{(k)} &= -\left(\mathbf{J}^\top \mathbf{R}^{-1}\mathbf{e}\right)_{\mathbf{u}=\mathbf{u}^{(k)}}, \end{aligned} \tag{2-24}$$

where $\mathcal{A} \in \mathbb{R}^{m\times m}$ is the Hessian matrix [22], whose elements can be given by

$$[\mathcal{A}]_{pq} = \sum_{l=1}^{m} \frac{\partial^2 \boldsymbol{g}(\mathbf{u})}{\partial[\mathbf{u}]_p \partial[\mathbf{u}]_q} [\mathbf{R}]_{ll}^{-1} [\mathbf{e}]_l. \tag{2-25}$$

The variant of Newton's method which ignores the Hessian matrix–not accounting for second order derivatives–results in the Gauss-Newton (GN) method, whose iteration will be,

$$\begin{aligned} \mathbf{G}^{(k)} \Delta\mathbf{u}^{(k)} &= \left(\mathbf{J}^\top \mathbf{R}^{-1}\mathbf{e}\right)_{\mathbf{u}=\mathbf{u}^{(k)}} \\ \mathbf{u}^{(k+1)} &= \mathbf{u}^{(k)} + \Delta\mathbf{u}^{(k)}. \end{aligned} \tag{2-26}$$

with $\mathbf{G} = \mathbf{J}^\top \mathbf{R}^{-1}\mathbf{J}$ as the gain matrix.

The convergence of this algorithm highly depends on the amount of omission due to the Hessian matrix $\mathcal{A}$. For local (when $||\mathbf{u}^{(0)} - \mathbf{u}^*||_2$ is small) convergence,

- if $||\mathcal{A}(\mathbf{u}^{(k)})|| << ||\mathbf{G}^{(k)}||$ then algorithm is quadratically convergent.
- if $||\mathcal{A}(\mathbf{u}^{(k)})|| < ||\mathbf{G}^{(k)}||$ then algorithm is linearly convergent.
- Otherwise, it may not be convergent at all.

In power systems, the impact of $\mathcal{A}$ is negligible only when measurement errors are simple, as noted in [23]. However, topological or parameter errors can significantly affect the system, making the convergence of the GN or Newton's method unreliable. To address this, global convergence criteria can be used by modifying the GN method employing techniques like line search or trust-region approaches as discussed in [24]. The main idea of these methods is to modify the step length $\Delta\mathbf{u}$ to make it feasible for the optimisation problem, which is discussed in detail in [25].

Upon comprehensively addressing the GN method, the PE techniques built upon it can be discussed.

Method of State-Vector Augmentation

This method adopts a joint approach to SPE. When the GN method accounts for parameters as states in its formulation, it is called the method of state-vector augmentation. Using (2-21), the WLS objective function (2-22) for this method becomes

$$\min_{\mathbf{u},\mathbf{p}} W(\mathbf{u},\mathbf{p}) = \begin{bmatrix}\mathbf{e} & \mathbf{e}_p\end{bmatrix} \begin{bmatrix}\mathbf{R} & 0 \\ 0 & \mathbf{R}_p\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{e} \\ \mathbf{e}_p\end{bmatrix}, \tag{2-27}$$

with $\mathbf{R}_p = E(\mathbf{e}_p\mathbf{e}_p^\top)$ the error covariance matrix for parameter vector.

Method of Residual Analysis

As opposed to state-vector augmentation, residual analysis is a sequential method that first performs SE, identifies high residuals as bad data, and focuses on the network parameters associated with these bad data for PE. However, this method relies on linear approximations of PF equations, limiting validity and struggles to distinguish between errors due to bad measurements versus incorrect parameters. Additionally, estimating transformer tap positions critically depends on reactive power injection and terminal voltage measurements, often unavailable in DGs.

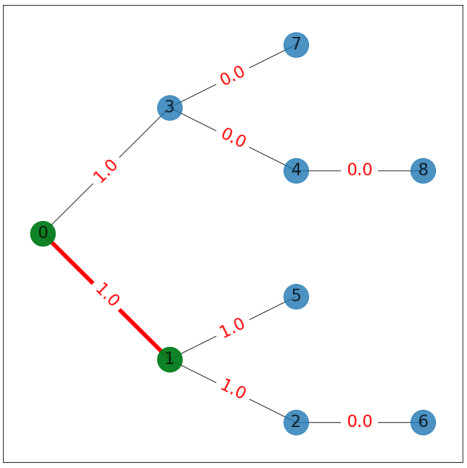
The state-vector augmentation method relies on global information from all network buses, while the residual analysis method uses highly local data from the terminal buses of the transformer. Figure 2-7b shows how tap-position errors affect SE accuracy across varying parameter error levels and neighbourhood sizes (see Figure 2-7a). The results indicate that such errors influence a region broader than the immediate neighbourhood but not the entire network, underscoring the need for a balanced approach. Specifically, a method that incorporates information beyond the terminal nodes, yet avoids full-network dependence, can offer more effective estimation. Section 2-2-2 explores such methods, which leverage neighbourhood topology for parameter estimation.

Since both the methods for PE depend on the GN, the limitations in GN impact the performance of PE methods. The main limitation of the GN method is that it assumes Gaussian noise and a full-rank Jacobian resulting from a fully observable network, which is not the case in practical DGs. Moreover, even for fully observable networks, convergence issues in these methods can stem from high weights on specific measurements, significant disparities in line lengths, or an excess of power injection data.

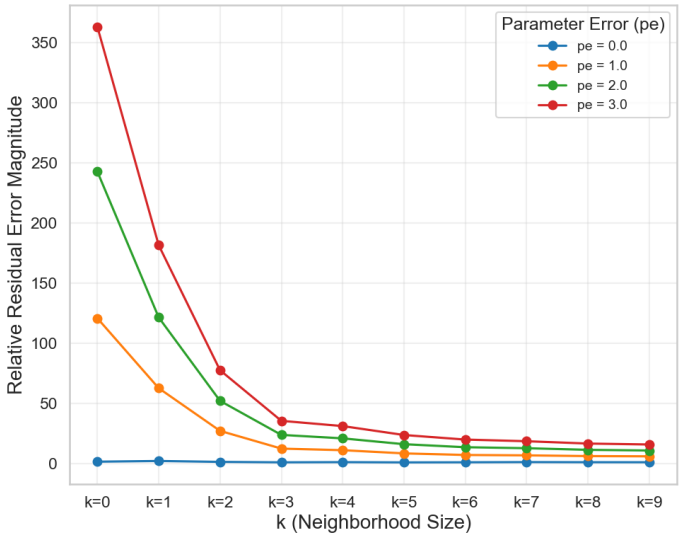
To address such limitations, neural network methods offer a promising solution by (1) bypassing the need for linearization, which forms the basis of GN method and residual analysis, (2) being trainable to identify patterns and correlations in the data, aiding in distinguishing measurement errors from parameter inaccuracies, and (3) reducing reliance on strict assumptions, thereby making the model more adaptable to real-world scenarios. These methods are discussed in the next section as emerging methods.

2-2-2 Emerging Methods

The logical foundation of the emerging methods is to employ a neural network (NN) as a universal functional approximator [26]. They learn the nonlinear mapping between input



(a) One-hop neighbourhood of transformer (edge) connecting bus 0 and 1 highlighted in red. Edges considered as neighbourhood are marked by 1.0, otherwise 0.0. This makes one-hop neighbours of the transformer as buses (nodes) 2, 3 and 5.



(b) Impact of transformer tap-position error on residual magnitudes across different neighbourhood sizes in a 42-bus network, normalised relative to the baseline case (pe = 0.0).

Figure 2-7: Visualisation of transformer neighbourhood (left) and tap position influence (right).

measurements and system states or parameters without requiring explicit physical models. These methods can be classified into physics-agnostic neural networks (PANNs) and physics-informed neural networks (PINNs).

Physics-Agnostic Neural Networks

The PANNs include fully connected neural networks (FCNNs), convolutional neural networks (CNNs), and autoencoders, which are collectively referred to as deep neural networks due to their multiple hidden layers and strong approximation capabilities [26]. Early work in this area, such as [27] applied a single perceptron NN for SE. But, the number of inputs required here must be at least $2n + 1$ for a network with n buses to ensure the output is meaningful. The capabilities of FCNNs are enhanced in recent studies, like [4], by applying hierarchical agglomerative clustering to merge statistically correlated neurons, forming a more compact and interpretable architecture, illustrated in Figure 2-8.

The independence of physics-agnostic methods from network topology and parameters, while providing flexibility, comes at the cost of increased complexity. This can lead to overfitting, where the model learns to capture noise or irrelevant patterns in training the data, reducing its ability to generalise. Additionally, the lack of domain knowledge can result in the model producing infeasible results that are inconsistent with the underlying physical principles governing the system. Due to these reasons, the model learnt may not be interpretable, adding a risk of bias. Therefore, physics-informed neural networks are discussed in the next section.

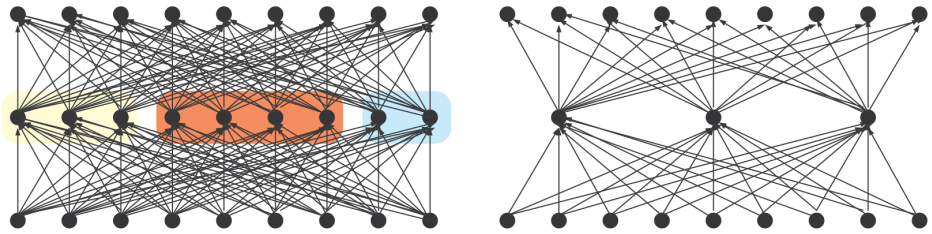


Figure 2-8: Neuron reduction in a layer via clustering: Left panel illustrates three distinct clusters in the hidden layer, while right panel depicts the clustered layer with a single representative neuron for each cluster [4].

Physic-informed Neural Networks

The concept of PINNs is well-suited and explored for application to DGSPE. There are a total of four general approaches to integrate the physics into a deep NN model [28, 29]:

1. **Physics-informed Initialisation**: Instead of relying on traditional initialisers such as zero, random, or Xavier initialisation [30], the nonlinear and non-convex nature of the PF problem can be addressed by leveraging sequential pre-training and fine-tuning. Pre-training involves training the model on a synthetic dataset to provide a robust starting point, followed by fine-tuning on real-world data to enhance convergence.

2. **Physics-informed Loss-Function**: Rather than exclusively using a vanilla loss function, physics-informed soft constraints can be added into the loss function alongside regularizers to embed prior knowledge of the system or penalise violations in model output. For example, in [31], the loss function is given as,

$$L^{\text{PINN}} = ||\mathbf{V}_{\text{bus}} - \hat{\mathbf{V}}_{\text{bus}}||^2 + \lambda_{\text{PINN}}||\underline{\mathbf{S}} - \hat{\mathbf{V}}_{\text{bus}}\mathbf{Y}^*_{\text{bus}}\hat{\mathbf{V}}_{\text{bus}}|| \tag{2-28}$$

   where $\hat{\mathbf{V}}_{\text{bus}}$ is the predicted voltage magnitude vector for all buses and other notations reused from Section 2-1-5. The first term is the vanilla loss between predicted and output voltage phasor, and the second term penalises the non-feasible PF equation solutions with λ_{PINN} as an arbitrary hyperparameter.

3. **Physics-informed design of NN Architecture**: Embedding physical principles directly into the NN architecture is another approach. The most utilised concept in this context involves leveraging the network topology in the PF problem to build GNN, which are extensively discussed in the following sections.

4. **Physics-informed hard constraint**: Since a machine learning model is essentially an optimisation problem, hard constraints can be incorporated into the model. However, this approach is generally ineffective due to the large number of parameters in typical DNN models [29]. While methods like the Krylov subspace approach can handle these constraints, they often lead to worse performance than a simple soft-constrained loss function.

The above approaches can complement each other in learning complex mappings, particularly for DGSPE. Among them, the most impactful is the physics-informed design of the NN architecture. This approach enhances computational efficiency and generalisation across varying network conditions by directly embedding a non-Euclidean data structure as the network topology into the model. Unlike FCNNs, GNNs enable localised message passing and sparse matrix operations, significantly reducing computational overhead and improving scalability. This architectural advantage makes them well-suited for real-world DG applications, as explored in Section 2-3 and Section 2-4.

2-3 Topological Representations for Power Systems

Graph representations enable the application of deep learning to non-Euclidean data[4], extending CNNs beyond traditional Euclidean domains, such as images. Graph representations, especially for domains like power systems, display remarkable capabilities to model and understand complex relationships between different components in the power grid [32]. This allows solving the SE and PE problems as node or edge-level prediction tasks depending on the representation. For the task of SPE, there are various ways of modelling the power grid as a graph. For example, the one-line diagram shown in Figure 2-2(b) can be transformed into various graph-based formats, as illustrated in Figure 2-9, to suit different modelling and learning needs.

- **Standard Bus-Branch Graph:** Represents buses as nodes and branches (lines/transformers) as edges, commonly used in [33, 34, 35, 36, 37]; illustrated in Figure 2-9(a).
- **Branch-as-Node Representation:** Each branch is modelled as a node with embedded terminal bus information, as proposed in [38].
- **Three-Phase Extensions:** Product graphs are used to couple single-phase nodes across multiple phases, allowing three-phase SE as demonstrated in [39, 40]; see Figure 2-9(b).
- **Heterogeneous Hypergraphs:** All grid components are modelled as nodes or edges with diverse feature sets, forming hyper-nodes [41] or hyper-edges [42], as shown in Figure 2-9(c-d).

This work extends the standard bus-branch graph to a hybrid representation that integrates undirected and directed graphs as discussed in detail in Section 3-2 in the context of DGs. Here, the abstract representations will be discussed. The undirected graph representation is detailed in the next section, followed by that for the directed graph. The latter is complemented by simplicial complexes (SC), which enhances the directed graph representation by capturing high-order interactions in the network topology.

[4]Data not defined by a Euclidean distance metric

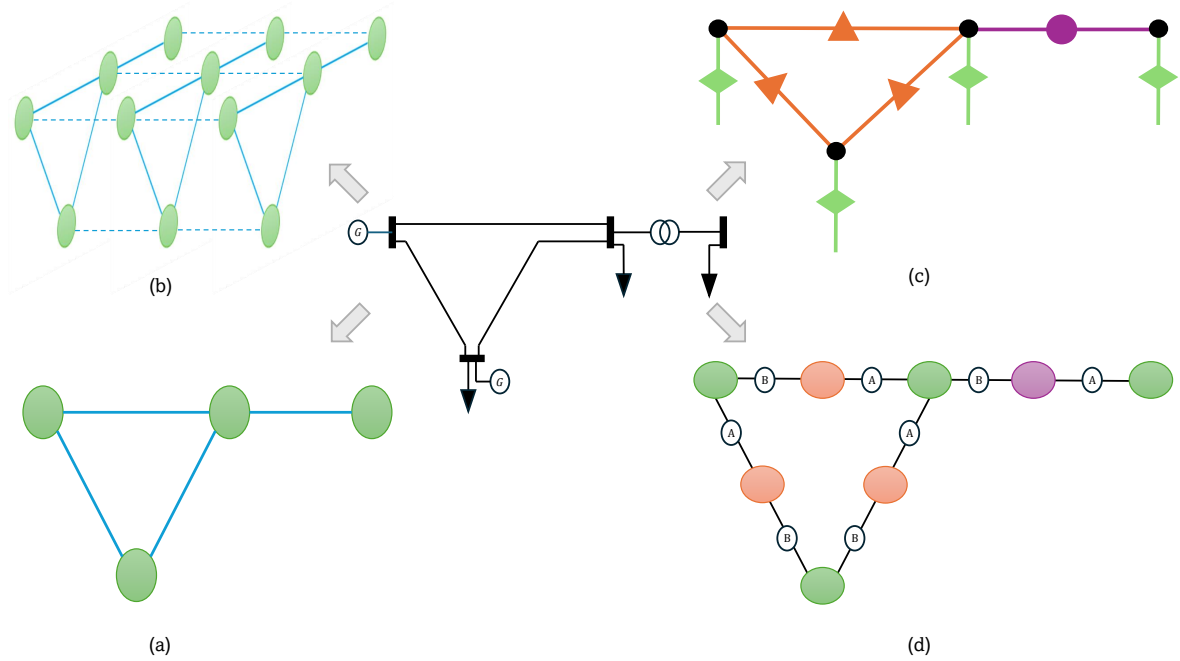


Figure 2-9: Graph representations of Figure 2-2(b): (a) Simple graph with node and edge; (b) Cartesian product graph with no inter-phase dependencies among neighbouring nodes; (c) Heterogeneous hyper-graph with hyper-edges; (d) Heterogeneous hyper-graph with hyper-nodes where A and B represent the direction associated with the power flow across the component. Non-black colours represent feature spaces, with identical colours indicating the uniform feature space.

2-3-1 Terminology for Graph Representation

A common graph terminology, is given by an undirected unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denote the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ the set of edges. For this graph, the neighbouring set for a node i is denoted by $\mathcal{N} = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$. The spatial dependency of the nodes is represented by a graph shift operator (GSO) denoted by $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. The generic requirement for GSO is that $[\mathbf{S}]_{ij} = 0 \quad \forall (i,j) \neq \mathcal{E}$, and $i \neq j$. The graph adjacency matrix, $\mathbf{A}$ and Laplacian matrix, $\mathbf{L}$ are the special cases of $\mathbf{S}$. Both of them are sparse, symmetric, and positive semi-definite matrices. For a weighted $\mathbf{A}$, $[\mathbf{A}]_{ij} > 0 \quad \forall (i,j) \in \mathcal{E}$. Furthermore, a diagonal matrix $\mathbf{D}$ for a graph denotes ith diagonal element as node-degree, i.e., $|\mathcal{N}(i)|$ for node i. Using $\mathbf{A}$ and $\mathbf{D}$, the graph Laplacian is defined as, $\mathbf{L} = \mathbf{D} - \mathbf{A}$. For numerical stability, which is relevant for GNNs, the normalised version of these matrices is used as GSO,

$$\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \qquad \tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}, \tag{2-29}$$

where $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{L}}$ denote the normalized adjacency and Laplacian.

Node features are represented by the matrix $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_F] \in \mathbb{R}^{|\mathcal{V}|\times F}$ with F features per node, and edge features by $\mathbf{X}^1 = [\mathbf{x}_1^1, ..., \mathbf{x}_G^1] \in \mathbb{R}^{|\mathcal{E}|\times G}$ with G features per edge. An equivalent node- and edge-wise representation can be given as

$$\mathbf{X} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_{|\mathcal{V}|}^\top \end{bmatrix}, \quad \boldsymbol{x}_i \in \mathbb{R}^F \quad \forall i \in \mathcal{V} \tag{2-30}$$

and

$$\mathbf{X}^1 = \begin{bmatrix} (\boldsymbol{x}_1^1)^\top \\ (\boldsymbol{x}_2^1)^\top \\ \vdots \\ (\boldsymbol{x}_{|\mathcal{E}|}^1)^\top \end{bmatrix}, \quad \boldsymbol{x}_i^1 \in \mathbb{R}^G \quad \forall i \in \{0, 1, \ldots, |\mathcal{E}|\}, \tag{2-31}$$

respectively. To distinguish filter banks in the coming discussion, for $F = 1$ and $G = 1$, the feature vectors are denoted as $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$ and $\mathbf{x}^1 \in \mathbb{R}^{|\mathcal{E}|}$ for node and edge, respectively.

Remark 2.1 The notation for node and edge feature vectors and matrices holds for both the undirected and directed graph representation.

A notable limitation with the graph representations is their inability to capture richer information about network edges–specifically, the spectral dependencies that arise from modelling higher-order structures like edges and triangles within the topology. Approaches leveraging SC have been proposed here to address this limitation [43, 44].

2-3-2 Terminology of Simplicial Complexes Representation

Conceptually, SCs generalise graphs. The above graph representation can be replaced with the set of directed edges $\mathcal{E}_d$ so that directed graph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E}_d)$. For the set of nodes $\mathcal{V}$, $k-$simplex $\mathcal{S}^k$, is a subset of $\mathcal{V}$ consisting of $k+1$ unique elements. Therefore, $\mathcal{V} = \mathcal{S}^0$ represents the node-set. Similarly, edge-set is $\mathcal{E}_d = \mathcal{S}^1$ and triangle-set can be denoted as

$$\mathcal{S}^2 = \{\{v_i, v_j, v_k\} | v_i, v_j, v_k \in \mathcal{V}, v_i \neq v_j \neq v_k\}.$$

Moreover, a simplicial complex $\mathcal{X}^K$ of order K, is a collection of simplices such that for any $k-$simplex $\mathcal{S}^k$, it includes any subset $\mathcal{S}^{k-1} \subset \mathcal{S}^k$, as illustrated in Figure 2-10. Like the adjacency matrix, which encodes the spatial dependency of the nodes in the graphs, Hodge Laplacians and incidence matrices encode that for SCs [43]. The incidence matrix, $\mathbf{B}_k \in \mathbb{R}^{|\mathcal{S}^{k-1}| \times |\mathcal{S}^k|}$, has $(k-1)-$simplices as rows and $k-$simplices as columns; representing the adjacency between these elements. For example, node-to-edge incidence will be represented by $\mathbf{B}_1$ and edge-to-triangle incidence will be represented by $\mathbf{B}_2$.

Furthermore, to understand the boundary condition in SCs, the boundary operator can be defined using graph-theoretic Hodge theory [43].

Definition 1. *(Boundary Operator) For a given $k-$simplex, $\mathcal{S}^k$, the boundary homomorphism (or the boundary operator) is $\partial_k : C_k \to C_{k-1}$ is,*

$$\partial_k \mathcal{S}^k = \sum_i (-1)^i [v_0, v_1, ..., \hat{v}_i, ..., v_n], \tag{2-32}$$

where $\hat{v}_i$ indicates that point v_i is deleted from the sequence and C_k is the space of $k-$chains [43].

An example can make it more concrete,

- For $\mathcal{S}^1 = [v_1, v_2]$,

$$\begin{aligned}\partial_1[v_1, v_2] &= (-1)^0[v_2] + (-1)^1[v_1] \\ &= v_2 - v_1.\end{aligned}$$

- For $\mathcal{S}^2 = [v_1, v_2, v_3]$,

$$\begin{aligned}\partial_2[v_1, v_2, v_3] &= (-1)^0[v_2, v_3] + (-1)^1[v_1, v_3] + (-1)^2[v_1.v_2] \\ &= [v_2, v_3] - [v_1, v_3] + [v_1, v_2].\end{aligned}$$

- Also note that, taking $1-$simplex boundary of $2-$simplex boundary,

$$\partial_1\partial_2[v_1, v_2, v_3] = [v_3] - [v_2] - [v_3] + [v_1] + [v_2] - [v_1] = 0. \tag{2-33}$$

When extending the concept of boundary operator to a finite set of $\mathcal{S}^k$, the algebraic representation is given by the incidence matrix. Thus, (2-33) extends as

$$\mathbf{B}_{k-1}\mathbf{B}_k = \mathbf{0}. \tag{2-34}$$

Using these adjacency relations for $\mathcal{X}^2$, the Hodge Laplacian structure of the graph as a SC can be fully given by,

$$\begin{aligned}\mathbf{L}_0 &= \mathbf{B}_1\mathbf{B}_1^\top, \\ \mathbf{L}_1 &= \mathbf{L}_{1,l} + \mathbf{L}_{1,u} := \mathbf{B}_1^\top\mathbf{B}_1 + \mathbf{B}_2\mathbf{B}_2^\top \\ \mathbf{L}_2 &= \mathbf{B}_2^\top\mathbf{B}_2.\end{aligned} \tag{2-35}$$

where, the matrix $\mathbf{L}_0 \in \mathbb{R}^{|\mathcal{S}^0|\times|\mathcal{S}^0|} = \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, known as the graph Laplacian, $\mathbf{L}_1 \in \mathbb{R}^{|\mathcal{S}^1|\times|\mathcal{S}^1|} = \mathbb{R}^{|\mathcal{E}_\Gamma|\times|\mathcal{E}_\Gamma|}$ captures relationships between edges in two ways: through the lower Laplacian $\mathbf{L}_{1,l}$, which accounts for shared vertices, and the upper Laplacian $\mathbf{L}_{1,u}$, which accounts for shared triangles. Similarly, $\mathbf{L}_2 \in \mathbb{R}^{|\mathcal{S}^2|\times|\mathcal{S}^2|}$ represents the connectivity or proximity between triangles based on the edges they share. These Laplacians will further allow the formulation of simplicial convolutional neural networks in Section 2-4-3.

In this context, the Hodge-Laplacian $\mathbf{L}_1$ can be further studied to gain more insights using an example from a subnet of a practical network in the Southern Netherlands. Topologically, for the subnet with 12 nodes, 14 edges and 3 triangles, consider $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|} = \mathbb{R}^{12}$, $\mathbf{x}^1 \in \mathbb{R}^{|\mathcal{E}|} = \mathbb{R}^{14}$ and $\boldsymbol{\tau} \in \mathbb{R}^{|\Theta|} = \mathbb{R}^3$ as the 1-dimensional signal on node, edge and triangles. The incidence matrices $\mathbf{B}_1, \mathbf{B}_2$ and their transpose $\mathbf{B}_1^\top, \mathbf{B}_2^\top$ can be interpreted by divergence, gradient, curl-adjoint, and curl operator as discussed next.

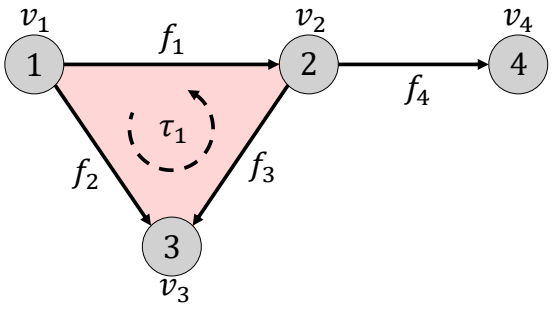


Figure 2-10: Simplicial complex representation of Figure 2-9(a) with arbitrary orientations (not strictly required but consistent with the classical definition). Signal for 0-simplices: $[v_1, v_2, v_3, v_4]$; signal for 1-simplices: $[f_1, f_2, f_3, f_4]$; and signal for 2-simplices: $[\tau_1]$. For example, with $\mathcal{S}^0 : \{2\}, \{4\}$, $\mathcal{S}^1 : \{1,2\}, \{1,3\}$, $\mathcal{S}^2 : \{1,3,2\}$ the corresponding SC will be $\mathcal{X}^2 : \{\{1\}, \{3\}, \{2\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1,3,2\}\}$.

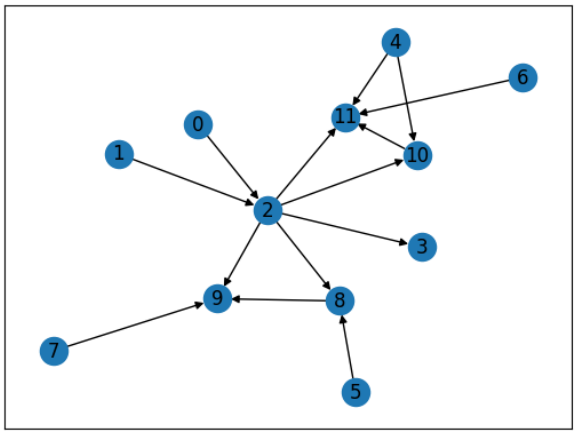


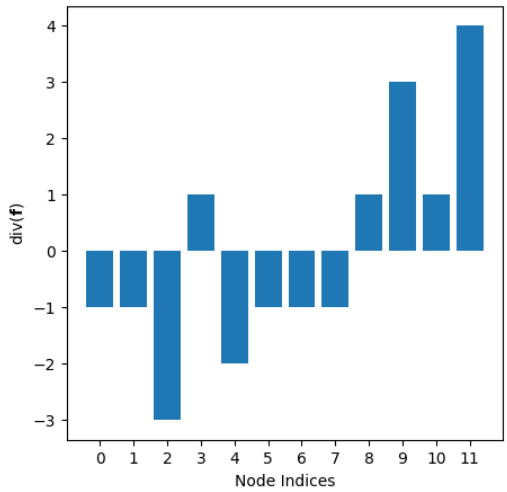
Figure 2-11: Subnet from a real DG with three 2-simplices (triangles) in the topology.

1. Divergence Operator $\mathbf{B}_1$: As the name suggests, the divergence operator, $\text{div}(\mathbf{x}^1) = \mathbf{B}_1\mathbf{x}^1$ when applied to the edge-flow, maps the flow *diverging* through each node of the graph. Along this line, for directed edges (i,j) and (j,k) connecting nodes i,j,k with edge-signal $[\mathbf{x}^1]_l$ and $[\mathbf{x}^1]_m$ respectively, the lth element of $\mathbf{B}_1\mathbf{x}^1$ will be $[\mathbf{x}^1]_l - [\mathbf{x}^1]_m$ (inflow minus outflow) [45]. Thus, in Figure 2-12a, the net flow can be illustrated for each node.

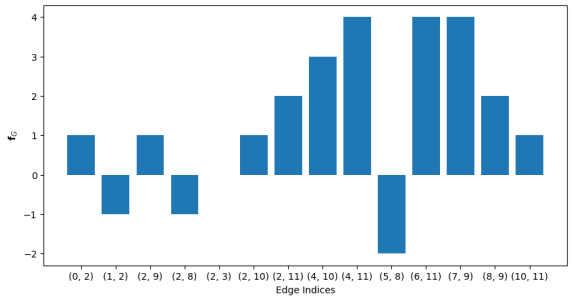
2. Gradient Operator $\mathbf{B}_1^\top$: The gradient operator for a directed graph is given by $\mathbf{x}^1_{\text{G}} \in \text{im}(\mathbf{B}_1^\top)$ where $\text{im}(\cdot)$ represents the image or column space of the matrix. Applying the gradient operator on the node-signal $\mathbf{x}$ is a simple matrix transformation, $\mathbf{B}_1^\top\mathbf{x}$. The logical foundation of this operator is taking the difference of node-signals across the directed edge. This interpretation is shown in Figure 2-12b for a random signal over the nodes.

3. Curl adjoint $\mathbf{B}_2$: The curl adjoint represents the image of $\mathbf{B}_2$, i.e., $\mathbf{x}^1_{\text{C}} \in \text{im}(\mathbf{B}_2)$. For instance, applying the curl over a triangle signal $\boldsymbol{\tau} = \mathbf{1}$, $\mathbf{x}^1_{\text{C}} = \mathbf{B}_2\boldsymbol{\tau}$ can be illustrated in Figure 2-12c.

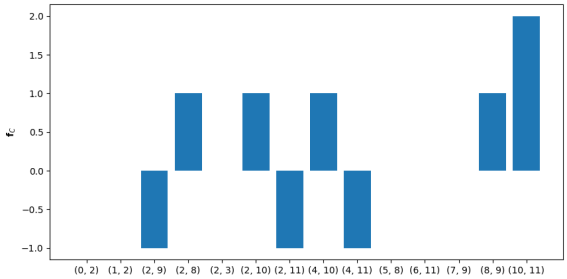
4. Curl Operator $\mathbf{B}_2^\top$: Applying curl operator to edge-signal $\mathbf{x}^1$, the net flow circulating along each triangle is computed. Curl operator is represented as, $\text{curl}(\mathbf{x}^1) = \mathbf{B}_2^\top \mathbf{x}^1$, so the ith element of $\text{curl}(\mathbf{x}^1)$ is the net flow *curling* across the ith triangle. This operator can be interpreted from Figure 2-12d.



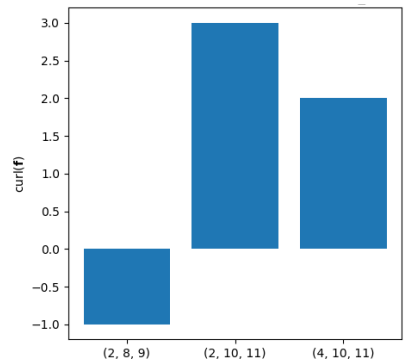
(a) Divergence for edge-signal $\mathbf{x}^1 = \mathbf{1}$. For instance, at node 3, $\text{div}(\mathbf{x}^1) = 1$, representing net inflow of value 1. This can be verified by Figure 2-11 with node 3 only with incident edge $(2, 3)$.



(b) Gradient $\mathbf{x}_{\mathsf{G}}^1$ for random node-signal $\mathbf{x} = [-1, 1, 0, 0, -2, 1, -2, -3, -1, 1, 1, 2]$. As an example, for $x_4 = -2, x_{10} = 1, x_{11} = 2$. The flow induced across the edges $(4, 10)$ and $(4, 11)$ will be 3 and 4 respectively.



(c) Curl adjoint $\mathbf{x}_{\mathsf{C}}^1$ for $\boldsymbol{\tau} = \mathbf{1}$. As seen in Figure 2-11, for the triangle formed by edges $(2, 8), (8, 9)$ and $(2, 9)$, the edge $(2, 9)$ is of uncyclic nature, i.e., for nodes (a, b, c) with $c > b > a$, cyclic edges are $(a, b), (b, c)$ and (c, a). Thus, $\mathbf{x}_{\mathsf{C}}^1 = -1$ for $(2, 9)$ for $\boldsymbol{\tau} = \mathbf{1}$. Moreover, edge $(10, 11)$ is a common edge for two triangles and is cyclic for both, therefore $\mathbf{x}_{\mathsf{C}}^1 = 2$.



(d) Curl operator $\text{curl}(\mathbf{x}^1)$ for random $\mathbf{x}^1 = [0, 2, 1, 0, -2, -1, -2, 1, 1, 1, 1, 1, 0, 2]$. For triangle $(2, 8, 9)$, edge-signal at $(2, 8), (8, 9)$ and $(2, 9)$ is $0, 0$ and 1 respectively. And since, $(2, 9)$ is uncyclic, $\text{curl}(\mathbf{x}^1) = 1 \cdot 0 + 1 \cdot 0 + (-1) \cdot 1 = -1$.

Figure 2-12: Interpretation of subspaces of incidence matrices $\mathbf{B}_1, \mathbf{B}_2$ using (a) Divergence operator showing net flow at nodes, (b) Gradient operator computing edge flows from node potentials, (c) Curl adjoint operator for triangle flows, and (d) Curl operator measuring circulation around triangles.

These interpretations also summarise the subspaces of the Hodge-Laplacian $\mathbf{L}_1$ using direct-sum decomposition given as

$$\mathbb{R}^{|\mathcal{E}|} = \text{im}(\mathbf{B}_1^\top) \oplus \text{im}(\mathbf{B}_2) \oplus \ker(\mathbf{L}_1). \tag{2-36}$$

Note that, columnspace$(\mathbf{B}_1^\top)$ = rowspace$(\mathbf{B}_1)$ = im$(\mathbf{B}_1)$. Out of these three subspaces, the first two, im$(\mathbf{B}_1^\top) \oplus$ im$(\mathbf{B}_2)$ are accounted for by above operators. The third subspace is the nullspace, which is accounted for by the harmonic operator.

5. Harmonic Operator: The $\ker(\mathbf{L}_1)$ is the harmonic space, which is both divergence- and curl-free. Reconsidering our example, for the given topology, the $\ker(\mathbf{L}_1) = \emptyset$.

These interpretations justify the richness of information that the SC can extract, complementing the simple graph representation.

In summary, five representations of power systems are discussed. The first four, excluding SCs, are common in the literature for distribution grid state estimation (DGSE). However, the fifth one, the SCs, demonstrates the potential to extract additional topological information applicable to a wide range of networks [46]. These representations can be applied to design topology-aware filters and neural networks. The former is a foundational topic in graph signal processing (GSP) discussed in Appendix B. Building on that, the latter is discussed in the next section.

2-4 Topology-Aware Neural Networks

GNNs are built on graph filters by introducing *learnable* filter coefficients and model complex dependencies by adding components like pointwise nonlinearities. Similar to how GSP extends digital signal processing to graph-structured data, GNNs extend traditional neural networks to operate on non-Euclidean domains. This section summarises the main GNN architectures utilised in this work, introducing graph convolutional neural network (GCNN), graph attention neural network (GAT), and SCNN.

2-4-1 Graph Convolutional Neural Networks

For a GCF defined in Definition 7, a graph perceptron nesting the GCF with a pointwise nonlinear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ can be denoted as

$$\mathbf{x}' = \sigma(\text{GCF}(\mathbf{x})), \tag{2-37}$$

where $\mathbf{x}'$ denotes output graph signal embedding of the perceptron. Some examples of this activation function for $x \in \mathbb{R}$ include $\text{ReLU}(x) = \max(0, x)$ or hyperbolic tangent $\tanh(x)$, depending on the task. Like the traditional multi-layer perceptron, the multi-layer graph perceptron is illustrated in Figure 2-13(a). Further, extending for graph signals with multiple features, using graph perceptron as denoted in (B-6), a multi-layer graph perceptron is shown in Figure 2-13(b). For a graph signal with multiple features $\mathbf{X}$, the number of features at each layer is determined by the GCNN parameter matrix $\mathbf{H}_{l,k} \in \mathbb{R}^{F_{l-1} \times F_l}$ at lth layer for kth filter order. So that parameter matrix for the first layer can be denoted as $\mathbf{H}_{1,k} \in \mathbb{R}^{F \times F_1}$.

This matrix maps the F_{l-1} input features from the previous layer to F_l output features in the current layer. As a result, the output embedding of the layer is represented as $\mathbf{X}_l \in \mathbb{R}^{|\mathcal{V}| \times F_l}$. Incorporating all the parameter matrices, the parameter set for GCNN can be denoted as

$$\mathcal{H}^{\text{GCNN}} = \bigcup_{l=0}^{L} \bigcup_{k=0}^{K} \mathbf{H}_{lk} \cup \mathcal{W}^{\text{GCNN}}, \tag{2-38}$$

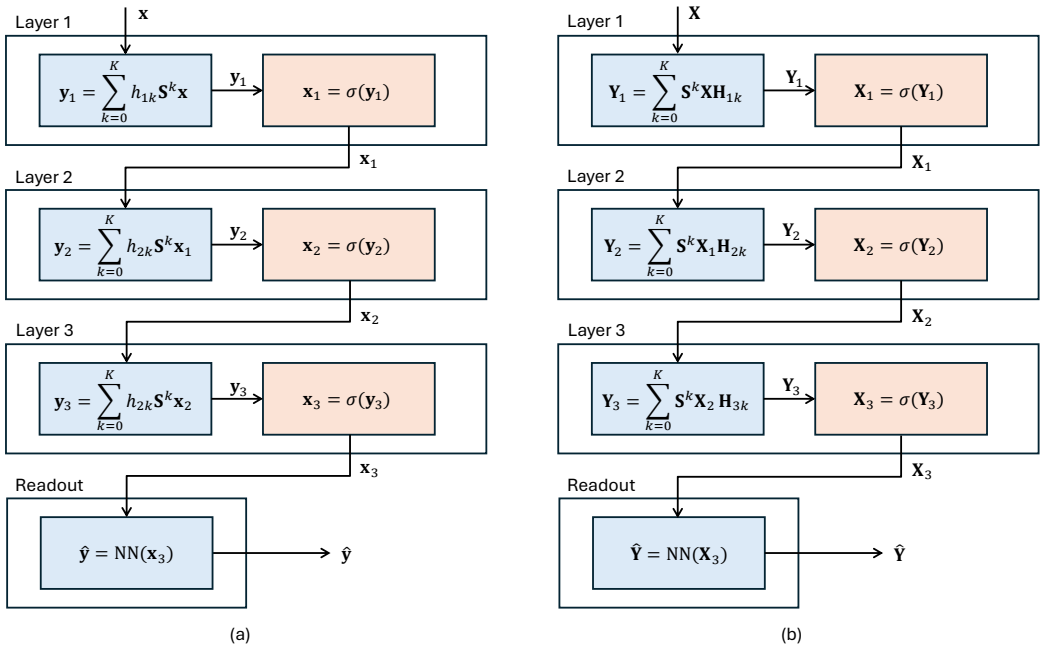


Figure 2-13: Graph Convolutional Neural Network. A single feature GCNN is shown in (a), and a GCNN with filter banks in (b), with three layers and a readout layer. The GCFs are shown in the blue boxes, and activation functions are shown in the orange boxes.

with $\mathcal{W}^{\text{GCNN}}$ denoting the parameter set of the readout layer, which depends on the specific task of the GCNN, such as node prediction, classification, etc. Since this work proposes to use the GCNN, a compact representation can be given as

$$\hat{\mathbf{Y}} = \text{GCNN}(\mathbf{S}, \mathbf{X}, \mathcal{H}^{\text{GCNN}}), \tag{2-39}$$

where $\hat{\mathbf{Y}}$ is the output node embeddings, as illustrated in Figure 2-13(b). The complexity of L layer GCNN is $\mathcal{O}(LF^2K(|\mathcal{E}| + |\mathcal{V}|))$, linear in the number of nodes and edges. The implementation of this architecture for the proposed work will be discussed in the next chapter.

2-4-2 Graph Attention Networks

Inspired by the popular self-attention mechanism used in natural language processing (NLP) [47], which enables learning the most relevant parts of sentences to make decisions, a similar concept can be applied to graph signals. The *relevance* between two nodes can be determined

by learning parametric edge weights from the node and connecting edge features[5] [48]. Using the representation for node-feature matrix from (2-30), a shared linear transformation is applied to every node parameterised by a weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$. Similarly, edge features $\mathbf{X}^1 = [\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, ..., \boldsymbol{x}_{|\mathcal{E}|}^1]$ are transformed using $\mathbf{W}_e \in \mathbb{R}^{F' \times G}$. This way, the relevance of node j's features to node i connected by edge k can be measured by the attention coefficient

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}_s^\top \mathbf{W}\boldsymbol{x}_i + \mathbf{a}_t^\top \mathbf{W}\boldsymbol{x}_j + \mathbf{a}_e^\top \mathbf{W}_e \boldsymbol{x}_k^1\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}_s^\top \mathbf{W}\boldsymbol{x}_i + \mathbf{a}_t^\top \mathbf{W}\boldsymbol{x}_j + \mathbf{a}_e^\top \mathbf{W}_e \boldsymbol{x}_k^1\right)\right)} \tag{2-40}$$

where $\mathbf{a}_s \in \mathbb{R}^{1\times F'}$, $\mathbf{a}_t \in \mathbb{R}^{1\times F'}$ and $\mathbf{a}_e \in \mathbb{R}^{1\times F'}$ shared across all nodes. The softmax function: $\frac{\exp(\cdot)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}}(\cdot)}$ here allows discriminating important nodes across the neighborhood of node i. Since the attention coefficients depend on the node degree, $\alpha_{i,j} \neq \alpha_{j,i}$. Learning all the attention coefficients in the graph is equivalent to learning a *attention-induced* GSO: $\mathbf{A}_\alpha$ $[\mathbf{A}_\alpha]_{i,j} = \alpha_{i,j}$. Thereby, the propagation rule, similar to GCNN for layer l will be

$$\mathbf{X}_l = \sigma(\mathbf{A}_\alpha \mathbf{X}_{l-1} \mathbf{W}^\top), \tag{2-41}$$

where $\mathbf{X}_l \in \mathbb{R}^{|\mathcal{V}|\times F'}$. Following the work of [47], the learning process can be stabilised by a multi-head attention mechanism [48]. In this mechanism, M independent attention mechanisms are executed, and the resulting feature matrices are averaged. This multi-head attention layer can be given as an extension of (2-41) as

$$\mathbf{X}_l = \sigma\left[\frac{1}{M}\sum_{m=1}^{M} \mathbf{A}_\alpha^{(m)} \mathbf{X}_{l-1} [\mathbf{W}^{(m)}]^\top\right]. \tag{2-42}$$

The total set of trainable parameters for GATs here will be

$$\mathcal{H}^{\text{GAT}} = \bigcup_{m=1}^{M} \mathbf{A}_\alpha^{(m)} \cup \bigcup_{m=1}^{M} \mathbf{W}^{(m)} \cup \mathcal{W}^{\text{GAT}}, \tag{2-43}$$

where $\mathcal{W}^{\text{GAT}}$ is a task-dependent set of parameters associated with the readout layer, similar to (2-38). To utilise this architecture in the proposed method, consider the compact representation of (2-42) as

$$\hat{\mathbf{Y}} = \text{GAT}\left(\mathbf{S}, \mathbf{X}, \mathbf{X}^1, \mathcal{H}^{\text{GAT}}\right), \tag{2-44}$$

where $\mathbf{S}$ specifies the neighborhood set $\mathcal{N}(i)$ used in (2-40). Some key properties of GAT are given below,

[5] Original work uses only node features, while standard implementations allow edge features.

1. Discriminatory power: The attention coefficients assign different importances to nodes of the same neighbourhood, improving the selectivity power of the model. Further, the learned α_{ij} can also benefit the interpretability of the model.
2. Applicable for varying degrees of nodes: By specifying arbitrary weights to the neighbours.
3. Inductive implementation: As the set of parameters is independent of the graph dimensions, this mechanism can generalise to unseen graphs.
4. Computational Complexity: The complexity of L layer GAT is $\mathcal{O}(LMF'(|\mathcal{V}|F + |\mathcal{E}|))$, which is linear in the number of nodes and edges, thus on par with the complexity of GCNNs. Since individual head computations are independent, they can be parallelised for efficient operation.

While GCNN and GAT architectures leverage spatial information with a primary focus on nodes and their features, they often overlook the rich spatial information embedded in edges and their multiple features, as motivated in Section 2-3-2. To address this limitation, SCNNs–naturally suited for edge-centric tasks–offer a promising alternative, as discussed in the next section.

2-4-3 Simplicial Complex Neural Networks

Building upon the simplicial convolutional filter (SCF) defined in (B-7) for 1−simplicial signals, the SCNN layer with SC perceptron nesting the SCF with a pointwise nonlinear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ can be denoted for layer l as

$$\mathbf{X}_l^1 = \sigma(\text{SCF}(\mathbf{X}_{l-1}^1)). \tag{2-45}$$

This can be followed by a readout layer having a similar design as discussed in Section 2-4-1 and denoting the SCNN output as $\hat{\mathbf{Y}}^1$. The parameter set associated with SCNN can be given as

$$\mathcal{H}^{\text{SCNN}} = \bigcup_{q_1=0}^{Q_1} \mathbf{H}_{q_1}^l \cup \bigcup_{q_2=0}^{Q_2} \mathbf{H}_{q_2}^u \cup \mathcal{W}^{\text{SCNN}}, \tag{2-46}$$

where $\mathcal{W}^{\text{SCNN}}$ is associated with the task-dependent readout layer. A compact representation for SCNN can be given as

$$\hat{\mathbf{X}}^1 = \text{SCNN}\left(\mathbf{L}_{1,l}, \mathbf{L}_{1,u}, \mathbf{X}^1, \mathcal{H}^{\text{SCNN}}\right). \tag{2-47}$$

The computational complexity of SCNNs with L layers is given by $\mathcal{O}(L|\mathcal{E}|(Q_1 + Q_2)DG^2)$ [49], excluding the readout layer, linear in the number of edges. Similar to Figure 2-13 for GCNN, the architecture of multi-layer SCNN is given in Figure 2-14.

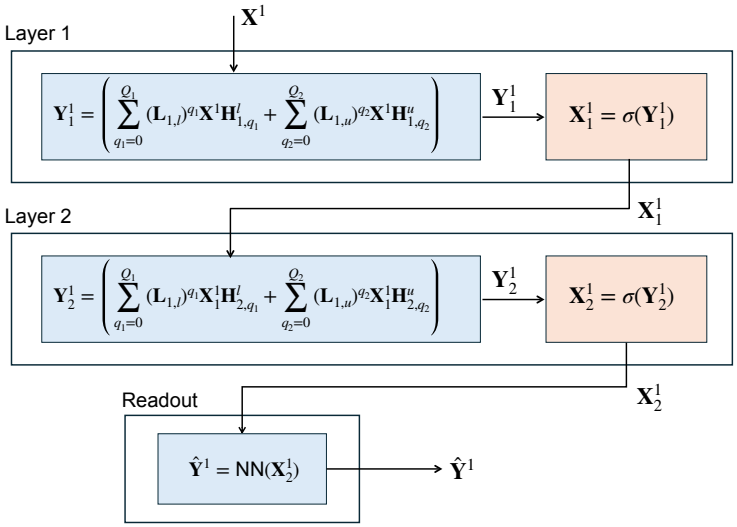


Figure 2-14: Simplicial complex neural network architecture with two layers. The simplicial complex filters are shown in blue boxes, with a readout layer and activation functions in orange boxes.

2-4-4 Recent Works with Topology-Aware Neural Networks

Several works utilise GCNN and GAT as their primary architecture for DGSPE. Table 2-1 distinguishes these works based on their scope and model architecture, if specified.

2-5 Topological Observability in Distribution Grids

From a systems and control perspective, in the context of static SE, the network is fully observable if there exists a unique mapping between the state vector $\mathbf{u}$ and the measurement vector $\mathbf{z}$ under ideal conditions, where the noise vector $\mathbf{e} = 0$. This is equivalent to solving a determined or overdetermined system of nonlinear power flow equations. For DGs, due to a limited number of measurement devices and their suboptimal placement, DGs tend to be completely unobservable [6]. This means that certain buses in the network lack measurements for at least two key quantities among $|\underline{V}|, \theta, P$, and Q. For example, Figure 2-15 shows an unobservable real network in the Southern Netherlands. It is observed that medium voltage (MV) stations: 4, 14, 18, and all low voltage (LV) terminals of MV/LV transformers do not have any measurements. From a mathematical standpoint, unobservable DGs yield an underdetermined nonlinear system with infinitely many solutions.

To complement real measurements, virtual measurements are sometimes inferred using Kirchhoff's and Ohm's laws. However, these laws assume that the mathematical model is accurate, which often fails due to factors like inaccurate system parameters, such as transformer tap positions. A more widely adopted strategy is to derive or generate pseudo-measurements, which involve estimating measurements at unobserved buses based on prior knowledge or historical data. The methods adopting this strategy are briefly classified below:

1. **Aggregated consumer load data**: For LV networks, smart meter data, which typi-

Table 2-1: Recent studies on SE for power systems. OH stands for Observability Handling.

Model name/ref.	**OH**	**PE**	**SCNN in Model Architecture**	**Other Limitations**
Power-GNN [37]	✓	✓	×	Applies Kron-reduction to eliminate unobserved nodes affecting feasibility of SE
MT-GCN [38]	×	✓	×	Assumes SE is accurate before performing PE which is unrealistic in practice
DSS [42]	✓	×	×	Assumes network is observable with pseudo-measurements
Ele-GNN [34]	✓	×	×	Diffusion matrices used for label propagation lack guaranteed spectral properties, affecting observability
GAEN [46]	×	×	✓	Assumes a fully-observable network not suitable for DGs

cally records cumulative energy consumption (in kWh), can be utilised to estimate power injection distributions. However, since SE relies on (5-minute average) Supervisory Control and Data Acquisition (SCADA) measurements with higher sampling rates than (monthly) smart meter data, parametric methods have been developed to infer stochastic power injection distributions for fast-timescale SE using slow-timescale smart meter data [4]. These methods, however, rely on the key assumption that aggregated consumer load is fully known, which is often violated due to privacy-related limitations on access to individual consumption data. Additionally, non-uniform data reporting and lack of synchronisation across measurement sources introduce further uncertainties to these methods [50, 51].

2. **Matrix-Completion Methods**: This method formulates all available data into a matrix and minimises its nuclear norm to estimate missing entries or unknown states. The biggest drawback of this method is that the formulation for minimising the nuclear norm restricts the use of linearised power flow equations, which limit their validity [52, 53].

3. **Label Propagation in Graph Signal Processing**: This method employs graph representation of the DG to enable GSP techniques. As proposed in the Ele-GNN architecture [34], observability is addressed by propagating known measurements–encoded as node features–from the corresponding buses to their connected neighbours through the graph edges. This propagation is governed by a diffusion matrix, where each element is defined using the so-called *electrical distance*: $[\ \mathbf{D}^{\text{elegnn}}\ ]_{ij} = \sqrt{r_{ij}^2 + x_{ij}^2}$. The node-feature matrix is iteratively multiplied by the diffusion matrix until convergence. Despite its simplicity, the definition of *electrical distance* used here oversimplifies the nonlinear and complex nature of the underlying power flow equations. This metric may be more *appropriate* for transmission grid (TG), where the reactance-to-resistance ratio (x/r)

is typically high (on the order of 5–20), allowing the approximation $\sqrt{r_{ij}^2 + x_{ij}^2} \approx x_{ij}$ to hold. But this diffusion behaviour undermines the physical relevance for DGs where $x/r < 1$.

Another critical issue with this method is the absence of guarantees regarding the spectral properties of the diffusion matrices. For convergence, the spectral radii must satisfy $\rho(\mathbf{D}_v^{\text{elegnn}}) \leq 1$ and $\rho(\mathbf{D}_e^{\text{elegnn}}) \leq 1$ for nodes and edge-graphs, respectively; however, this condition is not ensured in the study. Consequently, the reliability of this method for valid SE remains questionable.

The above methods, as discussed, do not directly address the observability problem as one of solving an underdetermined system of equations–an inherent challenge in practical DGs. While matrix-completion methods offer a formulation in this direction, they are restricted to using linear power flow equations [53]. Conceptually, the infinite solution space of an underdetermined system of equations can be constrained by implicitly conditioning the equations on the network topology and synthetic data distribution, which aligns with real loading scenarios. This helps to generate pseudo-measurements that are more *realistic* in nature. Reviewing methods in this direction leads to the umbrella term of generative models in machine learning. Since the scope of this work is limited to graph neural networks, generative models are briefly introduced in the following section to address pseudo-measurements.

2-5-1 Pseudo-measurements using Generative Models

Consider a tractable–simple to express–probability distribution $\mathcal{Z}$ defined in $\mathbb{R}^q$, such as a Gaussian distribution. Now consider the probability distribution χ defined in $\mathbb{R}^p$ representing the space of synthetic power flow results for a network with fixed topology. This distribution is typically multi-modal and high-dimensional due to the nonconvex nonlinear nature of the power flow equations. The goal of a generative model G_Φ parameterised by the set Φ is to learn a function

$$G_\Phi : \mathbb{R}^q \to \mathbb{R}^p \tag{2-48}$$

such that $G_\Phi(\boldsymbol{v}) \approx \boldsymbol{u}$, so that for each sample $\boldsymbol{u} \sim \chi$ there exists at least one point $\boldsymbol{v} \sim \mathcal{Z}$. In other words, a generative model G_Φ enables the mapping of samples from a simple distribution to a complicated distribution. This simple or tractable distribution is commonly called the *latent* distribution, a term originating from models like Restricted Boltzmann Machines [54]. In the above formulation, $\boldsymbol{q}$ is a latent variable that is unobserved, not undefined–it cannot be directly accessed or manipulated during training.

There are various techniques to learn a G_Φ, depending on the domain and task of the problem. Since this work generates synthetic power flow results using high-fidelity model-driven simulations, generative models relying on Markov chain Monte Carlo and variational inference techniques are disregarded for this study, bypassing the need for computationally intensive inference procedures and approximations in favour of direct, physics-based data generation. Moreover, inspired by its success across various scientific disciplines [55, 56], the generative adversarial network (GAN) incorporates highly effective methods, such as backpropagation and dropout [57]. It generates samples using forward passes, enabling any differentiable NN architecture in the G_Φ.

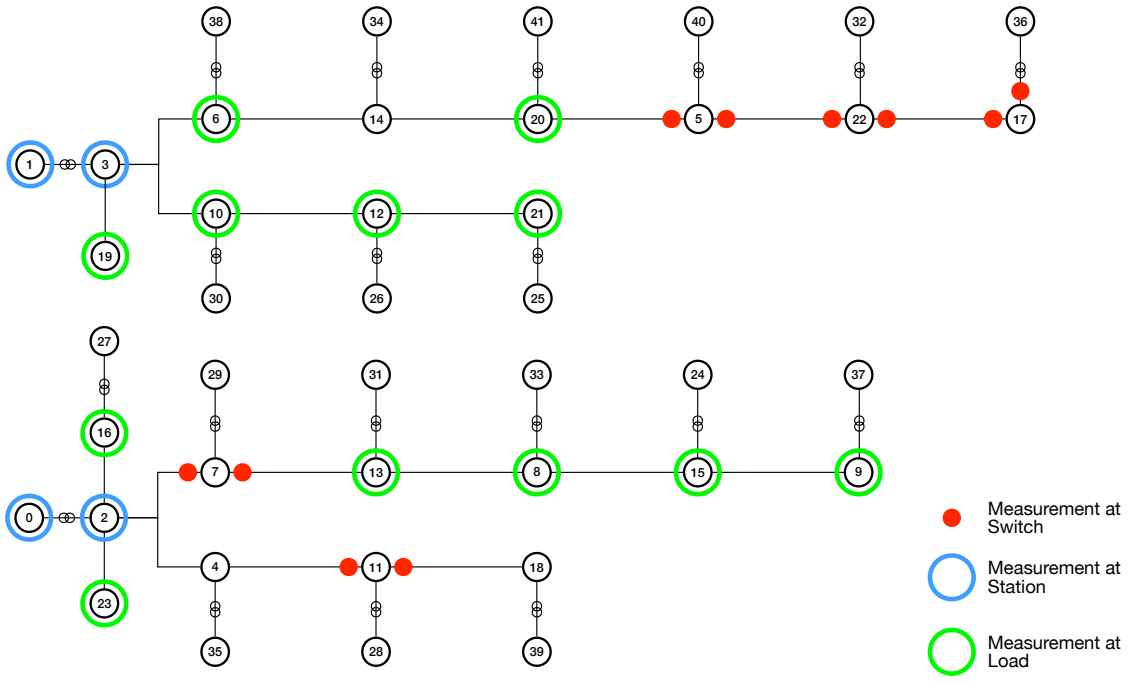


Figure 2-15: Available measurements highlighted in a 42-bus network from the Southern Netherlands. Three types of measurement devices are illustrated: devices in the switches connecting the station, devices within the station, and devices at the load connected to the station.

In this context, to generate pseudo-measurements, Kamal et. al., [58] propose a physics-informed GAN leveraging PF equations in its loss function to infer pseudo-measurements at unobserved buses. More recently, in [59], the Generative Adversarial Imputation Network (GAIN) from [60] is extended by incorporating convolutional neural network (CNN) into the model architecture, to improve the reconstruction accuracy. Despite these advancements, both works share a standard limitation: the generator model ignores explicit knowledge of the grid topology, which is also emphasised in [61]. In other words, neither the fully connected nor the convolutional architectures enforce the physical connectivity or locality inherent to electrical networks. To address this gap, a GNN is proposed to replace the fully connected neural network (FCNN) or CNN backbone.

The detailed design of GNN-based GANs, along with the entire proposed framework for DGSPE, is described in the next chapter.

Chapter 3

Proposed Methodology

This chapter delineates the topology-aware framework as the main contribution of this work for state and transformer tap-position estimation in unobservable distribution grid (DG). A high-level overview of this framework for a 4-bus network with one transformer is given in Figure 3-1. The first component of this framework is generative adversarial network (GAN), where the generator model G_Φ (Φ is the parameter set of the generator model) is trained to impute missing measurements. A discriminator model D_Υ (Υ is the parameter set of the discriminator model) provides adversarial feedback by learning to distinguish between real and imputed (potentially inconsistent) measurement sets, implicitly guiding G_Φ to produce more realistic outputs. This adversarial training is conditioned with the DG topology and synthetic power flow results, addressing limitations as discussed in Section 2-5-1 and ensuring that G_Φ synthesises realistic values.

Once G_Φ reconstructs observability in the system, the observable network is then passed to the next element of the framework: the state and parameter estimation (SPE) model M_Ψ. This model integrates graph neural network (GNN) and simplicial complex neural network (SCNN) architecture, followed by message-passing layers for predicting correct tap positions. Since the G_Φ and M_Ψ models are built upon a high-order, topology-aware neural network architecture, this chapter first explains the underlying graph representation, followed by an overview of the core model architecture. This architecture is then specialised to generate pseudo-measurements using GAN and perform SPE, which will be discussed in the following sections.

3-1 Hybrid Graph Representation

This section discusses the proposed graph representation using an example of a medium voltage (MV) network. Figure 2-15 represents a real network in the Southern Netherlands, hereby referred to as Net 42-A, with two high voltage (HV), 22 MV, and 18 low voltage (LV) buses, indexed as

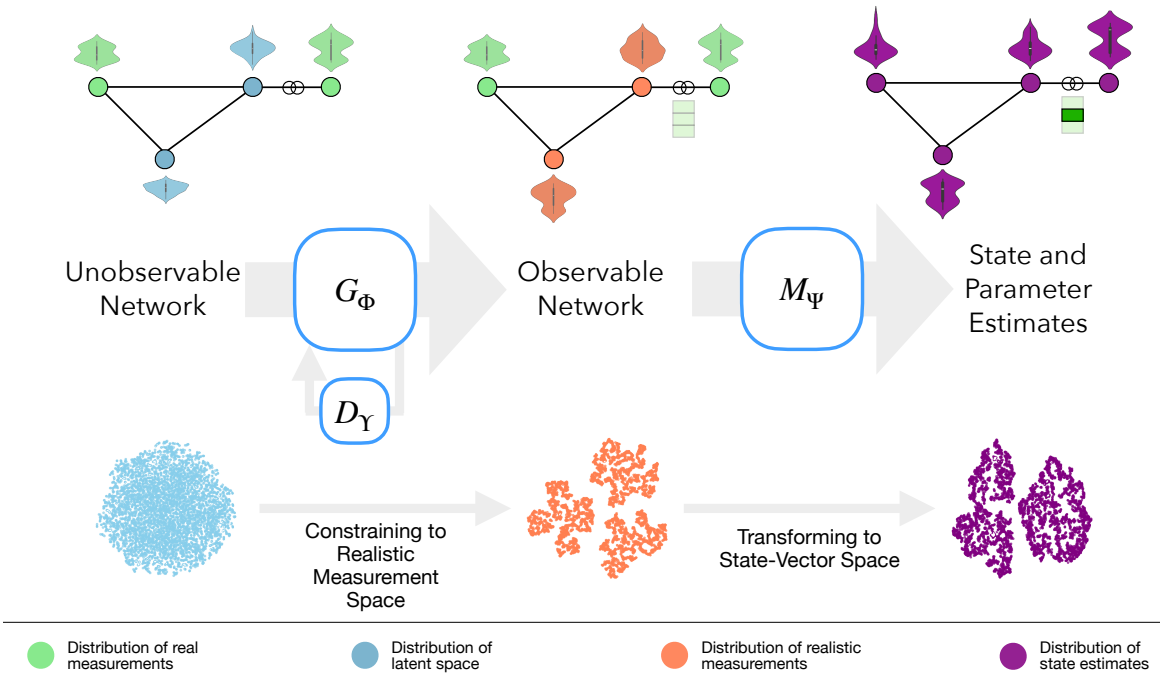


Figure 3-1: Conceptual overview of the proposed framework showing input/output distributions for each component in the framework via violin and scatter plots. Starting with a distribution of latent space at the missing measurements, the G_Φ and D_Υ models in the GANs constrain it to a realistic measurement space. This measurement space is next transformed into the distribution of the state estimates using the M_Ψ model. The edge-feature vector at the transformer edge represents logits for tap-position classification.

- HV Buses: {0,1},
- MV Buses: {2, 3,..., 23}, and
- LV Buses: {24, 25,..., 41}.

Moreover, it has 22 lines, 2 HV/MV transformers, 18 MV/LV transformers, and 20 connected loads, with 12 loads measured. Since HV/MV transformers are typically equipped with automatic tap-changers, they are disregarded for the parameter estimation (PE) task.

In the operational setting, typically, there are two configurations of measurements:

1. Power flow measurement, P_{ij}^{+}, at the switches connecting MV buses i and j.
2. Voltage magnitude $|\underline{V}|_i$ and active power injection P_i measurements at the buses, either as consolidated or simple measurements.

Remark 3.1 To clarify point (2) above, buses 2 and 3 correspond to central MV substations equipped with measurement devices across their respective busbars. Specifically, bus 2 aggregates connection to buses 4, 7, 16, and 22, while bus 3 serves buses 6, 10, and 19. To represent each bus with a single–consolidated measurement–the voltage readings from all

connected buses are averaged to obtain a representative voltage magnitude for the substation. Similarly, the net active power injection is computed as the sum of the individual power injections at the connected buses. These are considered consolidated measurements.

In (1), the power flow measurement is considered as edge-level information. Since these measurements are directional in nature at a given instant, they motivate the use of a directed graph representation. This directionality is essential to capturing the network's actual operational behaviour at a given time. Moreover, the line and transformer parameters can be added to this edge-level information to embed physical context and implicitly learning the bus-admittance matrix $\mathbf{Y}_{\text{bus}}$.

In (2), the voltage magnitude $|\underline{V}|_i$ and active power injection P_i are node features independent of any specific direction, thereby motivating an undirected graph representation. This choice further facilitates the application of graph convolutional neural network (GCNN), which operates effectively on undirected graphs by aggregating node features from symmetric neighbourhoods.

Bringing these two perspectives together, we adopt a hybrid graph representation proposing:

1. An undirected graph $\mathcal{G}$ to process node-level features,
2. A directed graph $\mathcal{G}_d$ to process edge-level features and account for high-order topological dependencies like edge-triangle relationship (see Section 2-3-2).

In this setup, all buses in the network are considered as nodes, i.e., $\mathcal{B} = \mathcal{V}$, and all physical connections–lines and transformers–are represented as directed edges, i.e., $\mathcal{U} = \mathcal{E}_d$. The node feature matrix is defined as $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times 2}$, where each row corresponds to a node feature vector $[\mathbf{X}]_i = [|\underline{V}|_i, P_i]$ for all $i \in \mathcal{V}$. Similarly, the edge feature matrix is defined as $\mathbf{X}^1 \in \mathbb{R}^{|\mathcal{E}|\times 6}$, where for each directed edge $(j,k) \in \mathcal{E}$, the corresponding feature vector is given by

$$[\mathbf{X}^1]_{jk} = [P^+_{jk}, r_{jk}, x_{jk}, g_{jk}, b_{jk}, \tau_{jk}].$$

Here, P^+_{jk} denotes the positive active power flow along the direction of edge (j,k), consistent with the directed graph formulation. Notably, due to the permutation equivariance of simplicial complexes (SC) [45], edges for which P^+ measurements are not available can be assigned an arbitrary direction without compromising the integrity of the learning process. The full representation of the hybrid graph representation is given in Figure 3-2.

To incorporate the underlying topology for learning:

- Node-focused graph convolution uses the normalised adjacency matrix $\tilde{\mathbf{A}}$ as defined in (2-29), enabling feature propagation in the undirected graph $\mathcal{G}$.
- Edge-focused graph convolution exploits higher-order topology using the lower and upper Hodge Laplacians, $\mathbf{L}_{1,l}$ and $\mathbf{L}_{1,u}$, associated with $\mathcal{G}_d$ as defined in (2-35).

The core model architecture utilises these convolution operations on node and edge features as detailed in the next section.

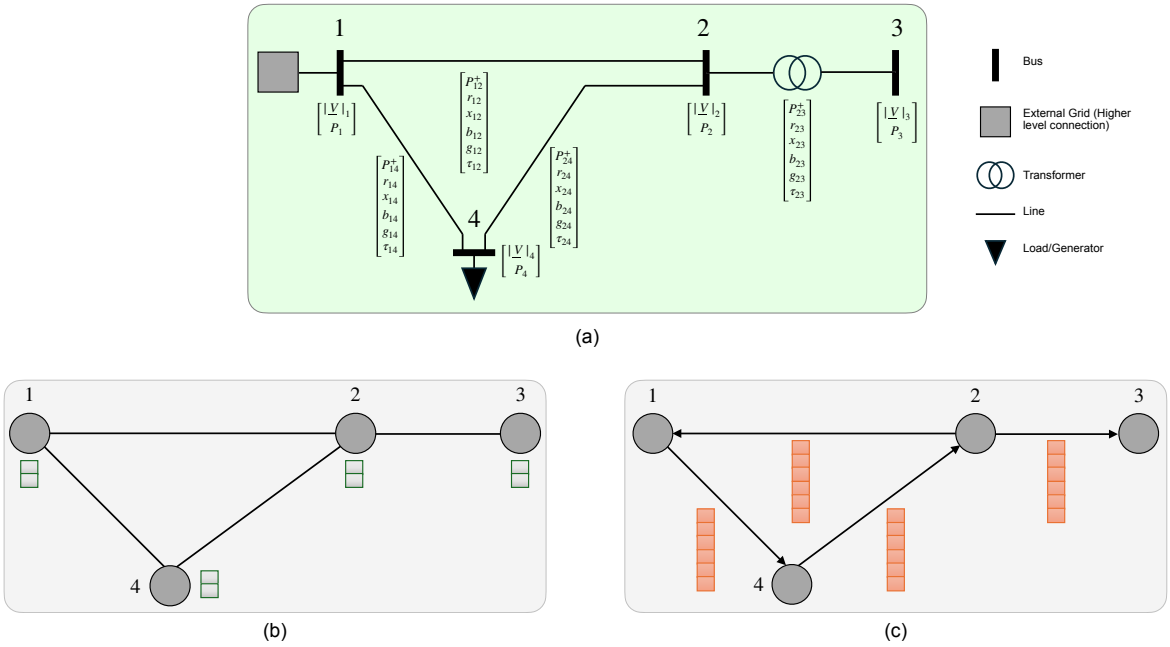


Figure 3-2: Hybrid graph representation for a 4-bus network representing nodes as PV buses and edges as lines and transformers (if edge is a line, $\tau = 0$). (b) and (c) represent the equivalent graph and simplicial complex representation of (a).

3-2 Core Model Architecture

The core model architecture proposed here uses the twin forces of graph and simplicial complex convolutional operations to independently derive high-dimensional node and edge embeddings. These embeddings are combined with a graph attention mechanism to output node- and edge-level spatial dependency-informed node embeddings that can be specialised further to generate pseudo-measurements and perform SPE.

Using the node-feature matrix from the proposed hybrid graph representation $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times 2}$, a GCNN from (2-39) using a normalised adjacency matrix, $\tilde{\mathbf{A}}$, to exploit the topology can be given as

$$\hat{\mathbf{Y}} = \text{GCNN}\left(\tilde{\mathbf{A}}, \mathbf{X}, \mathcal{H}^{\text{GCNN}}\right), \tag{3-1}$$

where $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, ..., \hat{\mathbf{y}}_{F_L}] \in \mathbb{R}^{|\mathcal{V}|\times F_L}$, with F_L denoting the output feature dimension produced by the final (i.e., L -th) layer of the GCNN.

Similarly, the high-dimensional edge-embeddings exploiting the topology of the directed graph with higher-order structures through an SCNN architecture from (2-47) can be given as

$$\hat{\mathbf{Y}}^1 = \text{SCNN}\left(\mathbf{L}_{1,l}, \mathbf{L}_{1,u}, \mathbf{X}^1, \mathcal{H}^{\text{SCNN}}\right). \tag{3-2}$$

where $\hat{\mathbf{Y}}^1 = \left[\hat{\mathbf{y}}_1^1, \hat{\mathbf{y}}_2^1, ..., \hat{\mathbf{y}}_{G_L}^1\right] \in \mathbb{R}^{|\mathcal{V}| \times G_L}$, with G_L denoting the output feature dimension produced by the final (i.e., L -th) layer of the SCNN. The node- and edge-embeddings are further passed to a graph attention neural network (GAT) network from (2-44) as

$$\mathbf{X}^o = \mathrm{GAT}\left(\mathbf{S}, \hat{\mathbf{Y}}, \hat{\mathbf{Y}}^1, \mathcal{H}^{\mathrm{GAT}}\right), \tag{3-3}$$

where $\mathbf{X}^o = \left[\mathbf{x}_1^o, \mathbf{x}_2^o, ..., \mathbf{x}_{F_o}^o\right] \in \mathbb{R}^{|\mathcal{V}| \times F_o}$, with F_o denoting the output node-feature dimension of the core model. This output embedding will be specialised for GAN and SPE, as discussed in the following sections. The architecture schematic of this model, with simplified notation, is shown in Figure 3-4 within the red box. The hyperparameters selected for the model architecture are discussed in the next chapter in Section 4-2.

3-3 Generating Pseudo-Measurements

This is the first component of the proposed framework, which utilises the core model architecture for generating pseudo-measurements for unobservable DG. Following the proposed graph representation from Section 3-1, consider a given network where the synthetic power flow results and network parameters are embedded into the node and edge feature matrices $\mathbf{X}$ and $\mathbf{X}^1$, respectively. To simulate the real measurement setup, distinguishing between available and unavailable measurements, mask matrices are introduced:

- $\mathbf{M}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times F}$ for nodes, and
- $\mathbf{M}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times G}$ for edges,

where the dimensions of these masks are equivalent to those of $\mathbf{X}$ and $\mathbf{X}^1$, respectively. Each element in the mask matrices is defined as:

$$[\mathbf{M}_{\mathcal{V}/\mathcal{E}}]_{ij} = \begin{cases} 1, & \text{if the corresponding measurement is available} \\ & \text{or the entry is a parameter (for edges),} \\ 0, & \text{otherwise.} \end{cases}$$

Using these matrices, the masked input matrices are defined as

$$\tilde{\mathbf{X}} = \mathbf{X} \odot \mathbf{M}_{\mathcal{V}} + \mathbf{Z}_{\mathcal{V}} \odot (\mathbf{1} - \mathbf{M}_{\mathcal{V}}), \tag{3-4}$$

$$\tilde{\mathbf{X}}^1 = \mathbf{X}^1 \odot \mathbf{M}_{\mathcal{E}} + \mathbf{Z}_{\mathcal{E}} \odot (\mathbf{1} - \mathbf{M}_{\mathcal{E}}). \tag{3-5}$$

Where $\odot$ denotes the element-wise (Hadamard) product, $\mathbf{1}$ is a matrix of ones of the same shape as the corresponding mask, and $\mathbf{Z}_{\mathcal{V}}$ and $\mathbf{Z}_{\mathcal{E}}$ are noise matrices (of the same shape as $\mathbf{X}$ and $\mathbf{X}^1$). This formulation of $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}^1$ embeds a latent distribution at the missing entries, enabling the training of generative/generator models. The following section discusses the generator model proposed in this work.

3-3-1 Generator Model

The generator model uses the proposed core-model architecture from Section 3-2. Given a network topology, consider the core model already conditioned with $\tilde{\mathbf{A}}, \mathbf{L}_{1,l}$ and $\mathbf{L}_{1,u}$. Using input feature matrices as $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}^1$, the core model produces node and edge embeddings using (3-3) and (3-2), respectively. This input-output relationship can be expressed as

$$(\bar{\mathbf{X}}, \bar{\mathbf{X}}^1) = G_\Phi(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}^1), \tag{3-6}$$

where $\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}^1$. The final layer dimensions of the core-model are kept such that $\bar{\mathbf{X}} \in \mathbb{R}^{|\mathcal{V}|\times F}$ and $\bar{\mathbf{X}}^1 \in \mathbb{R}^{|\mathcal{E}|\times G}$, to consider them as imputed node and edge-feature matrices. These matrices representing the G_Φ output distribution are distinguished from $\mathbf{X}$ and $\mathbf{X}^1$ representing the synthetic power flow distribution with a discriminator model discussed next.

Remark 3.2 From this point onward, the output matrices of the generator, $\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}^1$, will be referred to as *fake* measurements. In contrast, the ground-truth matrices $\mathbf{X}$ and $\mathbf{X}^1$, simulated from the synthetic power flow distribution, will be referred to as *real* measurements.

3-3-2 Discriminator Model

The discriminator model classifies the graph as real or fake. From a graph machine learning perspective, this boils down to binary graph classification–classify real as one and fake as zero. In this context, given the inputs to a discriminator model D_Υ: node-feature matrix $\mathbf{X}_D \in \{\mathbf{X}, \bar{\mathbf{X}}\}$, edge-feature matrix $\mathbf{X}_D^1 \in \{\mathbf{X}^1, \bar{\mathbf{X}}^1\}$ and graph adjacency $\mathbf{A}$, the D_Υ is designed to perform binary classification so that $D_\Upsilon\left(\mathbf{X}_D, \mathbf{X}_D^1, \mathbf{A}\right) \in [0, 1]$.

For such a graph classification task, the proposed work employs the DiffPool architecture [62]. This architecture uses a differentiable pooling mechanism that learns hierarchical representations by clustering nodes in a data-driven, end-to-end manner. The end-to-end differentiability of the DiffPool architecture enables the discriminator to learn structural and feature-based patterns that distinguish real graphs from generated ones. A step-by-step overview of this architecture, adapted for this work, is illustrated in Algorithm 1.

DiffPool performs two key operations at each layer: (i) learning a dense embedding of node features, and (ii) computing a soft assignment matrix to cluster nodes into fewer groups. The kth-layer of DiffPOOL maps: $\mathbb{R}^{|\mathcal{V}|\times F} \to \mathbb{R}^{|\mathcal{C}_k|\times F_k}$ with reduced number of nodes (clustered-nodes) as $|\mathcal{C}_k| < |\mathcal{V}|$ and transformed feature-space dimension F_k. These operations progressively reduce the number of clustered nodes across layers, $|\mathcal{C}_{k-1}| < |\mathcal{C}_k| < |\mathcal{C}_{k+1}|$, compressing the graph structure. The GCNN architecture (from (3-1)) is utilised at the core of these operations.

As shown in Algorithm 1, each layer outputs updated node features and a coarsened adjacency matrix based on the learned cluster assignments. This hierarchical compression continues until the final layer, where all nodes are aggregated into a single clustered node $|\mathcal{C}_L| = 1$. This is achieved by fixing the final assignment matrix to all-ones, $\mathbf{A}_L = \mathbf{1} \in \mathbb{R}^{|\mathcal{C}_{L-1}|\times 1}$, so that $\mathbf{X}_{D,L} \in \mathbb{R}^{1\times F_L}$, producing a graph-level representation.

The final output feature vector is passed through a linear transformation followed by a sigmoid activation to yield the discriminator score. The softmax$(\cdot)$ is applied across the rows. Since

Algorithm 1 DiffPool Architecture adapted for binary graph classification [62]

Require: Feature matrices $\mathbf{X}_{D,1} := \mathbf{X}_D$, $\mathbf{X}^1_{D,1} := \mathbf{X}^1_D$ and graph adjacency $\mathbf{A}_1 := \mathbf{A}$.
Require: Feature transformation dimensions at each layer $\{F_1, ..., F_i, ...F_L\}$.
Require: Reduced clustered-node sizes at each layer $\{|\mathcal{C}_1|, ..., |\mathcal{C}_i|, ..., |\mathcal{C}_L|\}$ with $|\mathcal{C}_L| := 1$.
Require: $\text{GCNN}_{\text{embed},i} : \mathbb{R}^{|\mathcal{V}|\times F_i} \to \mathbb{R}^{|\mathcal{V}|\times F_{i+1}}$
Require: $\text{GCNN}_{\text{pool},i} : \mathbb{R}^{|\mathcal{V}|\times F_i} \to \mathbb{R}^{|\mathcal{V}|\times |\mathcal{C}_i|}$
Require: Linear layer : $\mathbb{R}^{F_L} \to \mathbb{R}^1$
1: **for** $L-1$ layers **do**
2: $\quad \mathbf{Z}_{D,i} \leftarrow \text{GNN}_{\text{embed},i}(\mathbf{X}_{D,i}, \mathbf{A}_i)$
3: $\quad \mathbf{S}_i \leftarrow \text{GNN}_{\text{pool},i}(\mathbf{X}_{D,i}, \mathbf{A}_i)$
4: $\quad \mathbf{X}_{D,(i+1)} \leftarrow \text{softmax}(\mathbf{S}_i)^T \cdot \mathbf{Z}_{D,i}$
5: $\quad \mathbf{S}_{(i+1)} \leftarrow \text{softmax}(\mathbf{S}_i)^T \cdot \mathbf{A}_i \cdot \text{softmax}(\mathbf{S}_i)$
6: **end for**
7: $D_\Upsilon(\mathbf{X}_D, \mathbf{X}^1_D, \mathbf{A}) \leftarrow \text{sigmoid}(\text{Linear}(\mathbf{X}_{D,L}))$

$\text{softmax}(x) \in [0,1]\ \forall x \in \mathbb{R}$, the $D_\Upsilon\left(\mathbf{X}_D, \mathbf{X}^1_D, \mathbf{A}\right) \in [0,1]$. An output close to 1 indicates high confidence of D_Υ that the input graph is real. From this point onward, for brevity in notation, the discriminator model implicitly conditioned on the graph adjacency $\mathbf{A}$ is denoted as: $D_\Upsilon(\mathbf{X}, \bar{\mathbf{X}}^1, \mathbf{A}) := D_\Upsilon(\mathbf{X}, \bar{\mathbf{X}}^1)$.

3-3-3 Generative Adversarial Networks

Using the proposed G_Φ and D_Υ model architectures, the GAN setup is formulated here for adversarial training. Given the D_Υ model, the goal of the G_Φ model is to generate output $\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}^1$ such that $D_\Upsilon(\mathbf{X}, \bar{\mathbf{X}}^1) \to 1$: it classifies the graph as real. This requires training parameters Φ such that:

$$\begin{aligned} \Phi^* &= \arg\min_\Phi \quad -\log\left(D_\Upsilon\left(G_\Phi\left(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}^1\right)\right)\right) \quad \text{or, alternatively,} \\ &= \arg\min_\Phi \quad \log\left(1 - D_\Upsilon\left(G_\Phi\left(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}^1\right)\right)\right). \end{aligned} \tag{3-7}$$

Along this line, given the G_Φ model, the goal of the D_Υ is to classify the generated graph as a fake graph and a real graph as real. This can be expressed as learning the discriminator parameter set Υ such that:

$$\Upsilon^* = \arg\max_\Upsilon \quad \log D_\Upsilon\left(\mathbf{X}, \mathbf{X}^1\right) + \log\left(1 - D_\Upsilon\left(G_\Phi\left(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}^1\right)\right)\right). \tag{3-8}$$

Here, since $D_\Upsilon \in [0,1]$,

1. maximising the first term $\log D_\Upsilon\left(\mathbf{X}, \mathbf{X}^1\right)$ indicates maximising the probability that the discriminator model identifies the training sample as real so that $\log D_\Upsilon\left(\mathbf{X}, \mathbf{X}^1\right) \to 1$.

2. maximising the second term $\log\left(1 - D_\Upsilon\left(G_\Phi\left(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}^1\right)\right)\right)$ indicates maximising the probability that the discriminator model identifies the generated sample as fake so that $D_\Upsilon\left(G_\Phi\left(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}^1\right)\right) \to 0$.

Using (3-7) and (3-8), the GAN setup is formulated as a minimax objective function

$$\min_{\Phi} \max_{\Upsilon} \quad \log D_\Upsilon(\mathbf{X}, \mathbf{X}^1) + \log\left(1 - D_\Upsilon\left(G_\Phi(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}^1)\right)\right), \tag{3-9}$$

training G_Φ and D_Υ in an adversarial setting. Solving the above loss function is conceptualised as a two-player, zero-sum game: the gain of G_Φ is the loss of D_Υ and vice versa. The objective of the generator is to *deceive* the discriminator. Concurrently, the objective of the discriminator is to improve its accuracy in classifying real graphs as real and generator output as fake. This adversarial training is illustrated in Figure 3-3, and the chosen approach for solving the objective in (3-9) is discussed in Section 3-5. Given the GAN setup, which imputes missing measurements with realistic ones, the SPE can be performed as discussed in the next section.

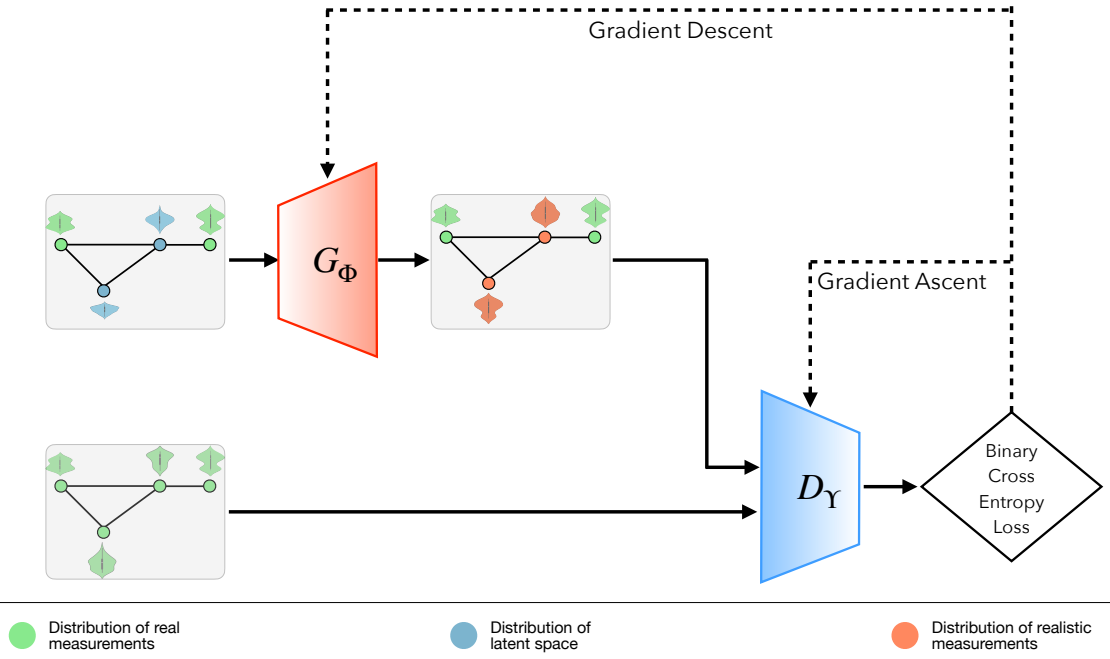


Figure 3-3: Adversarial training loop of GAN proposed for missing feature imputation. The dotted line represents the gradient descent and ascent updates to the parameters Φ and Ψ. These updates correspond to their respective minimisation and maximisation objectives, as defined in (3-7) and (3-8).

3-4 TapSEGNN Model for State and Parameter Estimation

This section extends the core model architecture to enable the prediction of transformer tap positions. The resulting model is referred to as the TapSEGNN model. The node and

edge feature matrices, denoted by $\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}^1$ respectively, are assumed to be imputed with pseudo-measurements generated by the trained model G_Φ. These matrices, together with the global network topology information in $\tilde{\mathbf{A}}, \mathbf{L}_{1,l}$ and $\mathbf{L}_{1,u}$, serve as inputs to the core model architecture. The core model is configured with an output feature dimension $F_o = 2$, corresponding to two quantities at each node: voltage magnitude $|\underline{V}|_i$ and phase angle θ_i, so that $\mathbf{X}^o \in \mathbb{R}^{|\mathcal{V}|\times 2}$. This output $\mathbf{X}^o$, accounting for the global network topology information, is passed to the transformer readout layers to put emphasis on the local network topology around the transformers, addressing the limitations in the method of state-vector augmentation and residual analysis as discussed in Section 2-2-1. To tailor these readout layers, consider the set of edges in the graph modelled as transformers:

$$\mathcal{N}_t = \{(i,j) : (i,j) \in \mathcal{E} \text{ and } (i,j) \text{ represents a transformer}\} \tag{3-10}$$

For each of such edges in $\mathcal{N}_t$, the set of h-hop neighborhood nodes can be given by the set

$$\mathcal{N}_b^{ij} = \{k \in \mathcal{V} |\ \text{dist}(k,i) \leq h \text{ or } \text{dist}(k,j) \leq h\}, \tag{3-11}$$

where $\text{dist}(i,j)$ represents the shortest path distance between node i and node j in terms of hops or edges. Let $\mathbf{M}^{ij}$ be the binary mask matrix for edge (i,j) of size $\mathbb{R}^{|\mathcal{N}_b^{ij}|\times|\mathcal{V}|}$. This mask selects the rows of $\mathbf{X}^o$ associated with the subset of nodes $\mathcal{N}_b^{ij} \subseteq \mathcal{V}$. The masked node features for edge (i,j) are then extracted as

$$\mathbf{X}^{o,ij} = \mathbf{M}^{ij}\mathbf{X}^o, \tag{3-12}$$

where $\mathbf{X}^{o,ij} \in \mathbb{R}^{|\mathcal{N}_b^{ij}|\times 2}$ consists of the node feature vectors $\boldsymbol{x}_k^o \in \mathbb{R}^2$ (see (2-30)) for each node $k \in \mathcal{N}_b^{ij}$, i.e.,

$$\mathbf{X}^{o,ij} = \begin{bmatrix} (\boldsymbol{x}_1^o)^\top \\ (\boldsymbol{x}_2^o)^\top \\ \vdots \\ (\boldsymbol{x}^o_{|\mathcal{N}_b^{ij}|})^\top \end{bmatrix}.$$

Using these node-feature vectors and given that the corresponding transformer (edge (i,j)) has total t tap positions, the output row-vector $\mathbf{t}^{ij} \in \mathbb{R}^{1\times t}$ is defined as

$$\mathbf{t}^{ij} = \left(\boldsymbol{a}^{ij}\right)^\top \left[\Big\|_{k\in\mathcal{N}_b^{ij}} \boldsymbol{\Omega}\boldsymbol{x}_k^o\right]^\top := \text{Tap}(\mathbf{X}^{o,ij}), \tag{3-13}$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{t\times 2}$ and $\boldsymbol{a}^{ij} \in \mathbb{R}^{|\mathcal{N}_b^{ij}|}$ are parameter matrices and $||$ denotes horizontal concatenation operator. The elements in $\mathbf{t}^{ij}$ are considered as unnormalised probabilities, or *logits*.

Note that the dimensions of $\mathbf{\Omega}$ are independent of $|\mathcal{N}_b^{ij}|$ and hence $\mathbf{\Omega}$ is proposed to be shared for all transformers $(i,j) \in \mathcal{N}_t$. For all transformers in the network, the total parameter set here will be

$$\mathcal{H}^{\text{Tap}} = \bigcup_{(i,j)\in\mathcal{N}_t} \boldsymbol{a}^{ij} \cup \mathbf{\Omega}. \tag{3-14}$$

At this point, comprising the core model and readout layers for transformer tap positions, the TapSEGNN model is mathematically referred to as M_Ψ with the parameter set $\Psi = \{\mathcal{H}^{\text{GCNN}}, \mathcal{H}^{\text{SCNN}}, \mathcal{H}^{\text{GAT}}, \mathcal{H}^{\text{Tap}}\}$. Considering M_Ψ has already incorporated the topology information in $\tilde{\mathbf{A}}, \mathbf{L}_{1,l}$ and $\mathbf{L}_{1,u}$ for (3-1), (3-2) and (3-3), a compact expression of input-output relationship for M_Ψ is given as

$$(\mathbf{X}^o, \mathbf{t}^{ij}) = M_\Psi(\bar{\mathbf{X}}, \bar{\mathbf{X}}^1) \quad \forall (i,j) \in \mathcal{N}_t. \tag{3-15}$$

To perform node-regression for state estimation (SE), consider node-label matrix $\mathbf{Z}$ given as

$$\mathbf{Z} = \begin{bmatrix} \boldsymbol{z}_1^\top \\ \boldsymbol{z}_2^\top \\ \vdots \\ \boldsymbol{z}_{|\mathcal{V}|}^\top \end{bmatrix}, \quad \boldsymbol{z}_i = [|\underline{V}|_i, \theta_i] \quad \forall i \in \mathcal{V}, \tag{3-16}$$

The quantities in $\boldsymbol{z}_i$ will be synthetic power flow results serving as labels to train for SE. The following loss function guides this training

$$L^{\text{SE}} = \sum_{i\in\mathcal{V}} \frac{1}{\sigma_i^2} \left\| \boldsymbol{x}_i^o - \boldsymbol{z}_i \right\|_2^2, \tag{3-17}$$

with σ_i as the standard deviation associated to measurement in $\boldsymbol{x}_i^o$ for node i.

Concurrently, the logit-vector $\mathbf{t}^{ij}$ from (3-15) is given as an input to the cross-entropy loss function [63] to predict the correct tap position for a single transformer as

$$L^{\text{tap-}ij} = -\sum_{k=1}^{t} y_k \log P(y = k|\mathbf{t}^{ij}), \tag{3-18}$$

where $P(y = k|\mathbf{t}^{ij})$ is the predicted probability for class k out of t classes, given input $\mathbf{t}^{ij}$ and y_k as the one-hot encoded label for true tap-position. Summing this classification loss for all transformers, $(i,j) \in \mathcal{N}_t$, will be

$$\begin{aligned} L^{\text{Tap}} &= \sum_{(i,j)\in\mathcal{N}_t} L^{\text{tap-}ij} \quad \text{or} \\ &= \sum_{(i,j)\in\mathcal{N}_t} \left(-\sum_{k=1}^{t} y_k \log P\left(y = k|\mathbf{t}^{ij}\right) \right). \end{aligned} \tag{3-19}$$

Combining the L^{SE} and L^{Tap}, the joint SPE loss function can be given as

$$L^{\text{TapSE}} = L^{\text{SE}} + \lambda_{\text{tap}} L^{\text{Tap}} + \lambda_{\text{reg}} \mathcal{R}(\Psi), \tag{3-20}$$

where λ_{tap} as the hyperparameter weighing L^{Tap} as compared to L^{SE}, $\mathcal{R}(\Psi)$ is the regularization term for all the trainable parameters and λ_{reg} denoting the weight on $\mathcal{R}(\Psi)$. The regularisation avoids overfitting of the model by penalising impractical parameter sizes. Typically implemented using absolute or Euclidean norms, it encourages model simplicity and improves generalisation to unseen grid conditions. So, the objective of the TapSEGNN model is to obtain

$$\Psi^* = \arg\min_{\Psi} \quad L^{\text{TapSE}}. \tag{3-21}$$

An architectural overview of TapSEGNN is illustrated in Figure 3-4. The following section elaborates on the chosen approach to solving the above objective, grounded in a systems and control perspective.

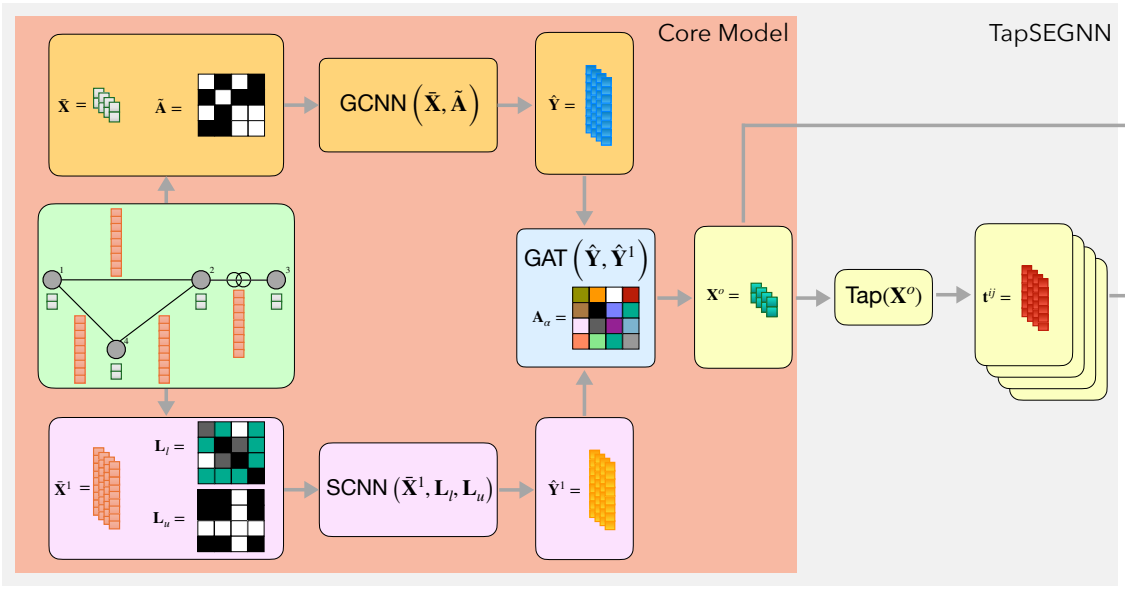


Figure 3-4: Architecture of TapSEGNN model proposed for SPE along with the core model highlighted within the orange box. A generalised Tap$(\mathbf{X}^o)$ layer is shown for all transformers. The output $\mathbf{X}^o$ is used for state estimation and $\mathbf{t}^{ij}$ $\forall (i,j) \in \mathcal{N}_t$ for tap position prediction.

3-5 Model Training

From a systems and control perspective, solving (3-9) and (3-21) requires identifying the underlying functions. Note that the functions proposed here are universal approximators or

so-called neural networks. They are considered non-convex functions [54] due to

1. Nonlinear activation functions like the sigmoid function in (3-19).
2. Weight space symmetry or non-unique configuration of model parameters leading to multiple local minima.

Thus, solving optimisation problems with neural networks involves iterative methods instead of deriving a closed-form analytical expression. Moreover, the differentiable nature of the proposed architectures allows employing *approximate*[1] first-order gradient-based iterative methods. Calculating the Hessian matrix often requires approximation, and the updates can be error-prone due to the typically poor condition number of these matrices [54]. While finite differences can be used to calculate the gradients, the cost of computation, for example, for M_Ψ, is $\mathcal{O}(|\Psi|^2)$ for $|\Psi|$ number of parameters. A better approach is the backpropagation algorithm [57], which is utilised in this work. Backpropagation allows for computing in $\mathcal{O}(|\Psi|)$–the exact cost to evaluate the model.

After calculating the gradient using backpropagation, the algorithms available to update the model parameters are either deterministic or stochastic. The former algorithms use all training examples, while the latter ones randomly select individual training examples. Recent advances in machine learning have demonstrated the dominance of stochastic methods over others, specifically the stochastic gradient descent method (SGD) [64]. However, the plain SGD update leads to problems like vanishing gradients at saddle points and erratic gradient behaviour due to the stochasticity. The Adam algorithm addresses these challenges by computing parameter-specific adaptive learning rates using gradient descent with estimated first and second moments of the gradients [65]. It uses exponential moving averages to accumulate the mean or *momentum* and uncentered[2] variance of past gradients, allowing for more stable and efficient updates. For these reasons, the Adam algorithm is used in this work to update the model parameters in solving the objective functions.

The next chapter presents a series of tests conducted to evaluate the proposed framework and its training aspects.

[1] $\text{ReLU}(x) = \max(0, x)$ isn't differentiable at $x = 0$, so in practice, gradients at that point are arbitrarily set

[2] Uncentered variance is $\mathbb{E}[x^2]$ instead of centered variance $\mathbb{E}[(x - \mathbb{E}[x])^2]$

Chapter 4

Results

This chapter presents a series of experiments to evaluate the proposed methodology using quantitative measures. It first proposes the case-study networks used in this study. Using these networks, it discusses an industrial-grade workflow for generating data in the real case study networks. This data, made up of power flow snapshots, is then customised into three separate datasets, each designed to train specific components of the proposed framework for their respective tasks. The trained models are then tested on unseen batches of the datasets to evaluate their performance, starting with the generative adversarial network (GAN) and followed by the TapSEGNN model. Since the core model architecture in these components is a graph neural network (GNN), the generalisability and scalability aspects of it are studied with a focus on state estimation (SE). The proposed core model is then compared with other baseline models to identify its pros and cons. Finally, a discussion summarises the main insights from the obtained results.

4-1 Dataset Generation

The case study network considered for this work is Net 42-A, as discussed in Section 3-1. Given the model hyperparameters and training configuration tuned for Net 42-A, this study tests the scalability and generalizability of the TapSEGNN model using the following networks:

- Net 320: An open-source synthetic network, known as MV Oberrhein. This network has 141 LV buses, 177 MV buses and two HV buses. It has 147 loads, 194 lines, 141 MV/LV and 2 HV/MV transformers. This network is used to test the scalability of TapSEGNN.
- Net 42-B: A real network from the southern Netherlands, similar to Net 42-A. This network has 18 LV buses, 22 MV buses and two HV buses. It has 19 loads, 20 lines, 18 MV/LV and 2 HV/MV transformers. This network is used to test the generalizability of TapSEGNN, but with a focus on SE only.

Using these networks, the dataset generation workflow includes two main steps:

1. Processing the real networks: The modern-day utility industries use Geographic Information Systems (GIS) to model the electrical networks. These systems are designed to analyse and manage the assets and present spatial and geographical data of the networks. The GIS network models are realised into Python-based dataframes using an open-source library called Pandapower [66]. The available measurement locations of the GIS network models are mapped to Pandapower networks. Typically, the SCADA measurements are available at some MV substations, and smart grid terminal (SGT) measurements are available at the LV terminal of MV/LV transformers. These measurements are mapped to the Pandapower network model at the buses and expressed in the mask matrices $\mathbf{M}_{\mathcal{V}}$ and $\mathbf{M}_{\mathcal{E}}$ as introduced in Section 3-3. Such mapping of measurements for a one-line diagram of Net 42-A is illustrated in Figure 2-15. For a synthetic network like Net 320, where no practical measurement locations are possible, 50% sparsity is randomly simulated in the mask matrices. In addition, historical measurements of active power P at the loads are also processed to derive the standard deviation σ_{load} in megawatts (MW). In summary, $\mathbf{M}_{\mathcal{V}}$, $\mathbf{M}_{\mathcal{E}}$ and σ_{load} are obtained in this step and stored in a dataframe as illustrated in the top cell of Figure 4-1.

2. Simulating the synthetic data: In this step, synthetic power flow snapshots are generated by perturbing the active power consumption/generation at the loads using zero-mean Gaussian noise with standard deviation σ_{load}. Two datasets are sampled from these snapshots: Dataset GAN and Dataset SE, under nominal transformer tap positions. Given a trained model G_{Φ}, the pseudo-measurements can be generated and $\bar{\mathbf{X}}, \bar{\mathbf{X}}^1$ can be sampled. This is useful to evaluate the influence of GAN performance for state and parameter estimation (SPE). Since this dataset is the imputed version of Dataset SE, it is referred to as Dataset ISE.

   Another dataset that reflects the variability in transformer settings, called Dataset TapSE, is generated. The tap positions are randomly perturbed for all MV/LV transformers. Each tap is sampled uniformly within its allowed operational range, defined by the minimum and maximum tap positions. This introduces variability in voltage regulation and network behaviour due to tap changes, in addition to load perturbations. All these datasets are mathematically expressed as follows:

   (a) Dataset GAN: This dataset is used to train the GAN in (3-9). Following the predefined notations, for a fixed network topology defined by $\tilde{\mathbf{A}}, \mathbf{L}_{1,l}$ and $\mathbf{L}_{1,u}$, this dataset is denoted for N samples as

   $$\mathcal{D}^{\text{GAN}} = \{(\mathbf{X}^{(i)}, (\mathbf{X}^1)^{(i)}, \mathbf{M}_{\mathcal{V}}^{(i)}, \mathbf{M}_{\mathcal{E}}^{(i)})\}_{i=1}^{N}. \tag{4-1}$$

   (b) Dataset SE: This dataset is used to evaluate the performance of the core model, assuming the network parameters are accurate and is trained for (3-17). Following predefined notations, for a fixed network topology defined by $\tilde{\mathbf{A}}, \mathbf{L}_{1,l}$ and $\mathbf{L}_{1,u}$, this dataset is denoted for N samples as

   $$\mathcal{D}^{\text{SE}} = \{(\mathbf{X}^{(i)}, (\mathbf{X}^1)^{(i)}, \mathbf{Z}^{(i)})\}_{i=1}^{N}. \tag{4-2}$$

Note the $\mathbf{X}, \mathbf{X}^1$ instead of the generator output $\bar{\mathbf{X}}, \bar{\mathbf{X}}^1$. The dataset sampled from the generator output $\bar{\mathbf{X}}, \bar{\mathbf{X}}^1$ is denoted as

$$\mathcal{D}^{\text{ISE}} = \{(\bar{\mathbf{X}}^{(i)}, (\bar{\mathbf{X}}^1)^{(i)}, \mathbf{Z}^{(i)})\}_{i=1}^{N}. \tag{4-3}$$

(c) Dataset TapSE: This dataset is used to assess the performance of TapSEGNN for joint state and parameter estimation. This dataset is denoted as

$$\mathcal{D}^{\text{TapSE}} = \{(\mathbf{X}^{(i)}, (\mathbf{X}^1)^{(i)}, \mathbf{Z}^{(i)}, \mathbf{T}^{(i)})\}_{i=1}^{N}. \tag{4-4}$$

In this dataset, the tap position τ_{jk} $\forall (j,k) \in \mathcal{N}_t$ is uniformly sampled from the range of minimum to maximum allowable tap positions. This tap setting is included in the feature vector as part of $[\bar{\mathbf{X}}^1]_{jk} = [P_{jk}^{+}, r_{jk}, x_{jk}, g_{jk}, b_{jk}, \tau_{jk}]$. The matrix $\mathbf{T}$ contains the actual tap positions, where $[\mathbf{T}]_{jk} = \tau_{jk}$, and these values are used to compute the resulting voltage magnitudes, angles, and active power injections and flows represented in $\mathbf{X}$ and $\mathbf{Z}$.

In the case study network Net 42-A, $\sigma_{\text{load}} \approx 10^{-4}$MW was identified. However, assigning this standard deviation to all loads results in limited variability in the power flow results, producing nearly identical training samples. This is problematic for training the model for SE, where a narrow range of noisy input features (voltage magnitude and active power) must map to a wider range of target values (filtered voltage magnitude and voltage angle). In particular, the relatively high variability of voltage angles can lead to one-to-many mapping, making it difficult for the model to learn the underlying nonlinear relationships. To address this, a standard deviation of $\sigma_{\text{load}} = 10^{-1}$ MW is assigned to all loads in the network when generating the snapshots for all datasets. The variability in this case is shown in Figure 4-2 with box-plots, corresponding to the $\mathcal{D}^{\text{SE}}$ dataset. On top of this, perturbation of transformer tap positions results in the voltage variability at the LV buses as shown in Figure 4-3.

The number of samples in all the datasets is set to 4096, a power of 2, to ensure efficient memory usage and computational performance on modern hardware. Selecting this number of samples instead of higher or lower powers of 2 is justified

- *theoretically*, because the standard error of the mean of the trained model as an estimator decreases with the square root of the number of samples. For example, for two datasets with 100 and 10,000 samples, the latter requires 100 times more computation, but estimator error is reduced only by a factor of 10 (see Section 5.4 of [54]).
- *empirically*, increasing the number of samples beyond 4096 did not significantly improve performance, indicating diminishing returns.

These datasets are used to train the models in the proposed framework: G_Φ, D_Υ and M_Ψ. The individual model performances are discussed in detail in the following section.

4-2 Performance of the Proposed Framework

In this section, the proposed framework is evaluated for Net 42-A. First, the task of generating pseudo-measurements is addressed, assessing the GAN. Second, the TapSEGNN model is evaluated using $\mathcal{D}^{\text{SE}}$ under the assumption of a fully observable network.

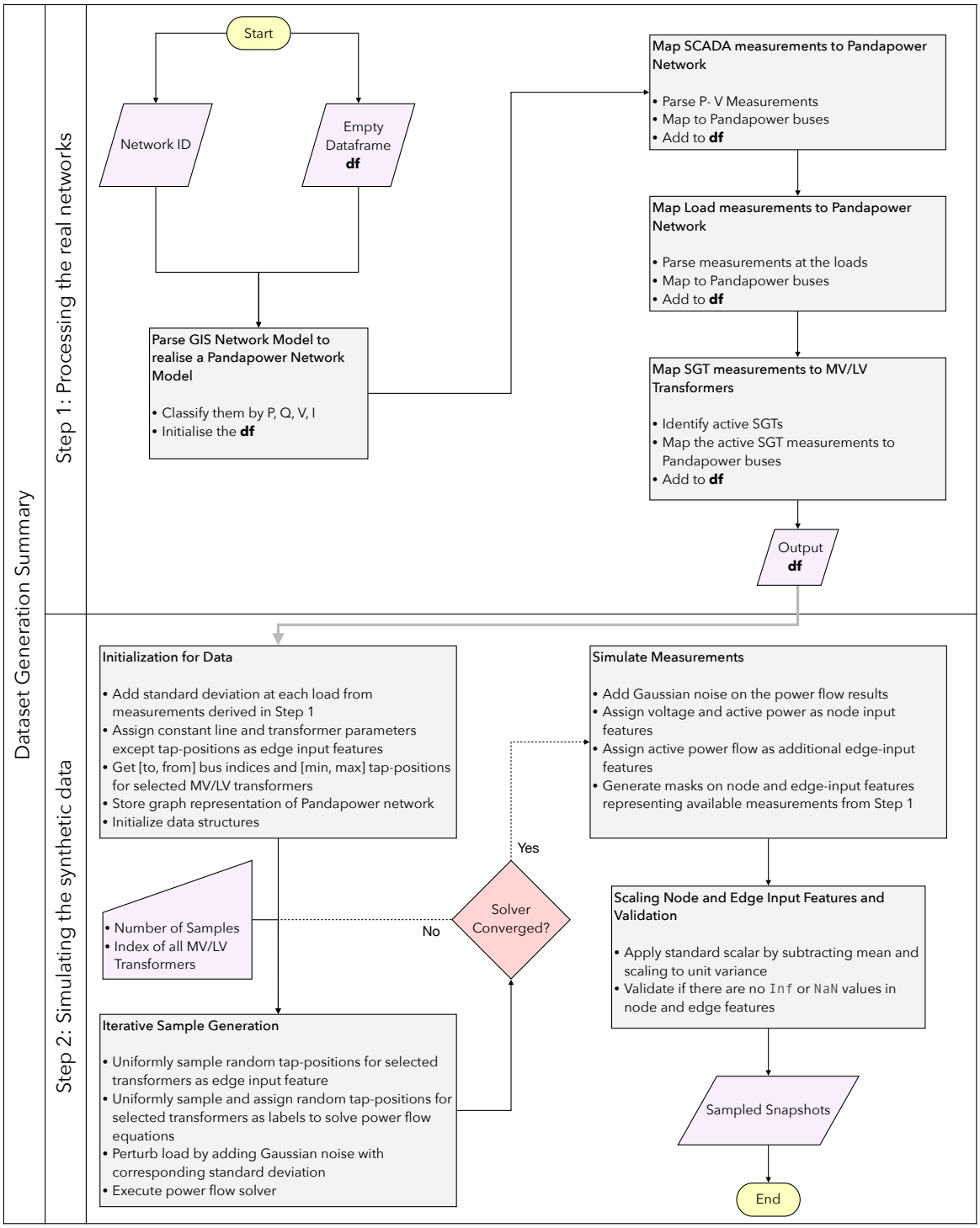
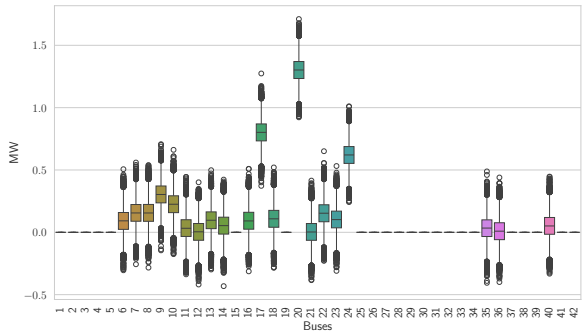
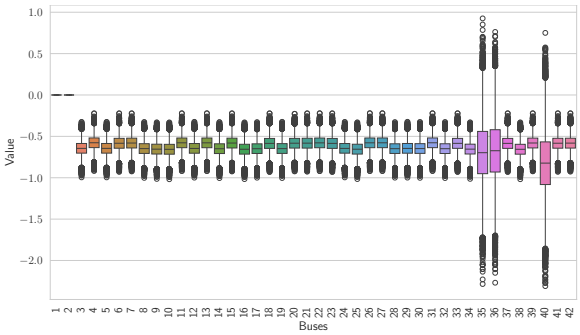


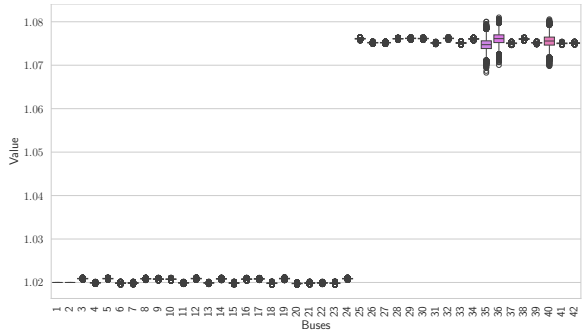
Figure 4-1: Workflow diagram for dataset generation process. Steps 1 and 2 of this workflow are shown in the top and bottom cells, respectively. The shapes used in the flowchart follow standard conventions.



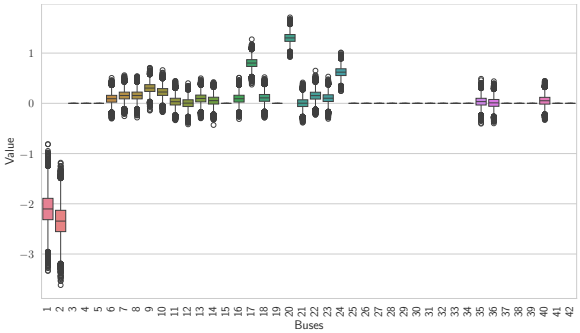
(a): Variability of load P_{load} (if present) at each bus



(b): Variability of voltage angle θ (degrees) at each bus



(c): Variability of voltage magnitude $|\underline{V}|$ (per unit) at each bus



(d): Variability of active power magnitude P (MW) at each bus

Figure 4-2: Variability (spread) of power flow results for $\sigma_{\text{load}} = 10^{-1}$ (MW) with box-plots. The spread of $|\underline{V}|$, P, and θ is comparable, allowing the neural networks to learn a one-to-one mapping for supervised learning. The colour coding differentiates the buses.

4-2-1 Performance of GAN

The models G_Φ and D_Υ are trained on $\mathcal{D}^{\text{GAN}}$ dataset. It is a standard practice to balance the capacity of G_Φ and D_Υ models to ensure a fair adversarial game [67]. If the D_Υ becomes too accurate, the gradient of generator model parameters: $\nabla\Phi$ may vanish. This could saturate the learning of parameters in G_Φ. To prevent this, the capacity of the two models is carefully balanced. The configuration of both these models is given in Table 4-1.

Given the comparable model capacity of both models, the training dynamics of GAN are more stable. This allows the discriminator to closely approximate its theoretical optimum, as discussed in Proposition 1 of [68]. Theoretically, at this optimum point (or Nash equilibrium from a game-theoretic perspective), the discriminator output satisfies $D_\Upsilon(\cdot) = 0.5$ for both the real and fake data, perfectly replicating the real data distribution. This idea is incorporated to examine the training of the GAN in this work. Therefore, in this work, the GAN is considered to be *successfully* trained if the following empirical criteria are met:

1. The discriminator accuracy stabilises around ≈ 0.5, indicating that it can no longer reliably distinguish between real and generated samples.

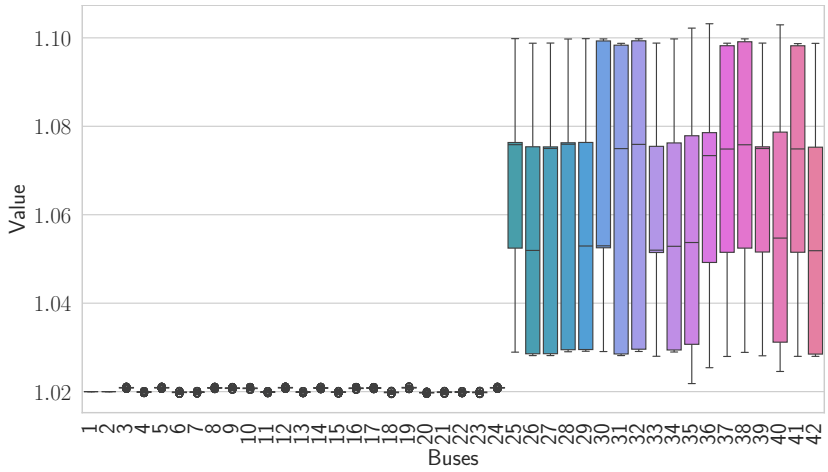


Figure 4-3: Variability of voltage magnitude $|\underline{V}|$ (per unit) at each bus for $\sigma_{\text{load}} = 10^{-1}$ (MW) and perturbing all transformer tap positions uniformly within their operational ranges. The colour coding differentiates the buses.

2. The generator and discriminator losses–defined in (3-7) and (3-8), respectively–plateau and no longer show improvement over time.

Note that these criteria are heuristic indicators used for this work, due to a lack of universal convergence guarantees for GANs [69, 70].

Experiments were conducted on the defined criteria and selected models, wherein the training process was executed. The corresponding hyperparameters utilised for configuring the training are summarised in Table 4-2. To ensure reproducibility of training outcomes, a range of random seed values was used for model initialisation and data shuffling involved in preparing the training set of $\mathcal{D}^{\text{GAN}}$. A random seed serves as an initial value for pseudo-random generators working in the backend of the processes. By fixing the seed, the randomness becomes deterministic, allowing for the consistent comparison of performances for various seeds and replicating results.

In this work, it was observed that the training dynamics were highly volatile and dependent on the initialisation of the model and data, a common characteristic of GANs. Therefore, the training was performed for 100 random seed values to evaluate diverse outputs. Figure 4-4 shows this volatility for generator loss from (3-7), discriminator loss from (3-8) and discriminator accuracy computed over 100 experimental runs. The average performance supports the viability and effectiveness of the GAN training configuration with accuracy converging to about ≈ 0.6 in 100 epochs.

Next, to evaluate the performance of the G_Φ model in these experiments, it is essential first to clarify the basis of comparison. The generator aims to learn the real measurement distribution; thus, quantifying the difference between the generated and real distributions provides a meaningful measure of its performance. Two distributions can be compared by metrics like Jensen–Shannon (JS) Divergence [71] and Kullback–Leibler (KL) Divergence [72]. Consider p_{data} denotes the real distribution and p_{model} the generator output distribution, then these divergences are given by

Table 4-1: Model Configuration for G_Φ and D_Υ. The number of parameters in both models is similar: 7363 for G_Φ and 7243 for D_Υ.

Model	Parameter	Value
G_Φ	Input channel for GCNN	2
	Hidden layer dimension for GCNN	64
	Filter order for GCNN	3
	Output channel for GCNN	32
	Input channel for SCNN	6
	Hidden layer dimension for SCNN	128
	Filter order for SCNN	1
	Output channel for SCNN	1
	Output channel for GAT	32
	Number of heads in GAT	1
	Total parameters in G_Φ ($\lvert\Phi\rvert$)	**7363**
D_Υ	Input channel for $\text{GNN}_{\text{embed}}$	2
	Hidden layer dimension for $\text{GNN}_{\text{embed}}$	64
	Output channel for $\text{GNN}_{\text{embed}}$	32
	Hidden two-layer dimension for GNN_{pool}	[21, 11]
	Total parameters in D_Υ ($\lvert\Upsilon\rvert$)	**7243**

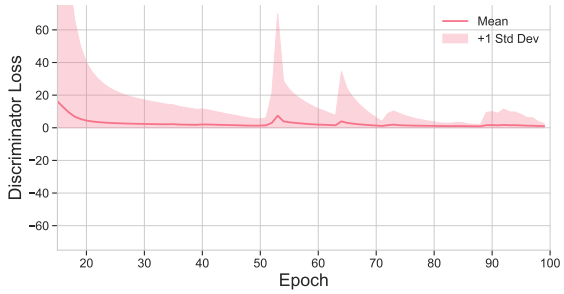
Table 4-2: Training configuration and hyperparameters for GAN models

Hyperparameter	Value
Number of epochs	100
Discriminator update steps per generator step (k_D)	2
Learning rate (Discriminator)	10^{-5}
Learning rate (Generator)	10^{-3}
L2 regularisation decay (Generator)	10^{-3}
L2 regularisation decay (Discriminator)	10^{-3}

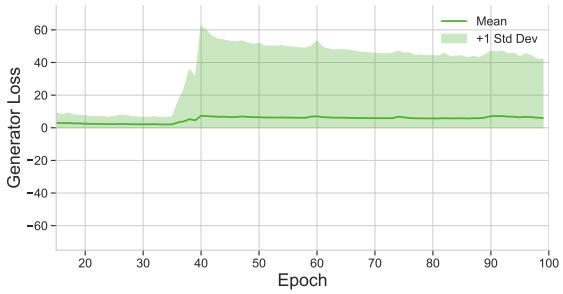
$$\text{KL}(p_{\text{data}}||p_{\text{model}}) = \sum_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)}, \quad \text{and} \tag{4-5}$$

$$\text{JS}(p_{\text{data}}||p_{\text{model}}) = \frac{1}{2}\text{KL}(p_{\text{data}}||p_{\text{mix}}) + \frac{1}{2}\text{KL}(p_{\text{model}}||p_{\text{mix}}), \tag{4-6}$$

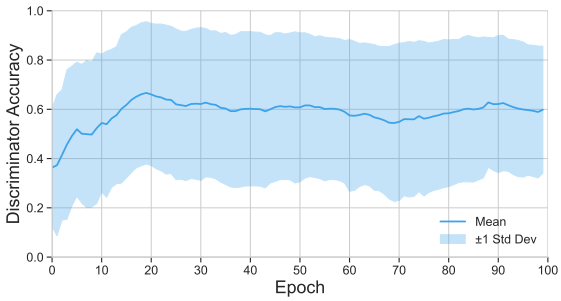
as KL and JS divergence, respectively. Here $p_{\text{mix}} = \frac{1}{2}(p_{\text{data}}+p_{\text{model}})$ is referred to as a mixture distribution and log is with base 2. A comprehensive discussion on theoretical justification for the success of GANs being linked to the minimisation of JS divergence rather than KL divergence can be found in [68, 69]. This topic lies beyond the scope of the present work. However, the main takeaway is that JS divergence offers more interpretable insights for evaluating the generator performance, primarily because it is bounded: $0 \leq \text{JS}(p_{\text{data}}||p_{\text{model}}) \leq 1$. For this reason, JS divergence is used in this work as a metric of comparison across the 100 experiments. The generator performance corresponding to the run with the lowest JS diver-



(a) Discriminator loss curve with confidence interval given by $+1$ standard deviation.



(b) Generator loss curve with confidence interval given by $+1$ standard deviation.



(c) Discriminator accuracy with confidence interval given by ± 1 standard deviation.

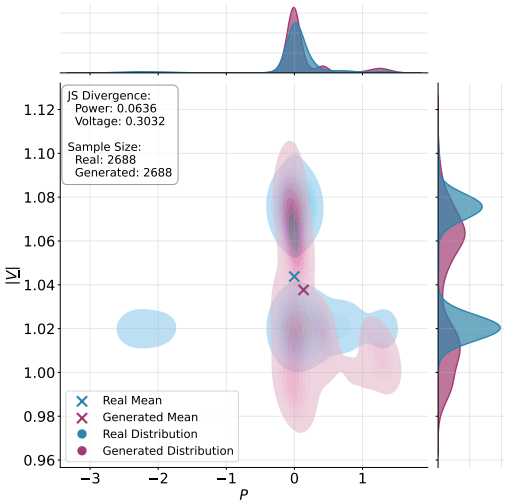
Figure 4-4: Loss curves and accuracy from GAN training initialised with 100 unique seeds. The high standard deviation for losses and accuracy displays the vulnerability of GAN to suboptimal performance. On average, the losses diminish and the discriminator accuracy converges to 0.6 (close to 0.5), indicating a balanced adversarial training.

gence is reported in Figure 4-5. Here, the distribution in $\mathbf{X}$ is compared with that in $\bar{\mathbf{X}}$, and similarly, the distribution of P^+ in $\mathbf{X}^1$ is compared with that in $\bar{\mathbf{X}}^1$. The distributions are visualised using kernel density estimation (KDE), a non-parametric method for estimating the probability density function of a continuous variable, allowing for a smooth approximation of the underlying data distribution. The contours in Figure 4-5a are 2-D KDE of the underlying scatter plots.

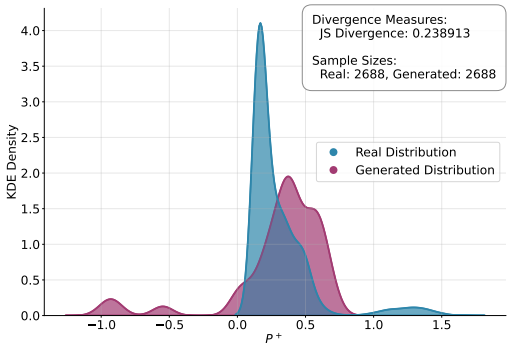
Remark 4.1 In this work, normalised KDE[1] is used as a visualisation tool–it does not model or learn the relationship between the inputs and outputs. As such, it cannot capture complex, high-dimensional dependencies or be used for tasks like imputing missing measurements.

Figure 4-5a shows that the G_Φ successfully captures the underlying distribution of active power P, while its approximation of the voltage magnitude distribution $|\underline{V}|$ is comparatively less accurate. Nevertheless, G_Φ demonstrates an ability to identify the major modes of the distributions, albeit with some bias. This shows that the model avoids mode collapse, a prevalent issue observed in GANs [67], thereby confirming a balance of the model capacity of both the G_Φ and D_Υ. The contour plots in the lower-right area of Figure 4-5a indicate that the model exhibits some sensitivity to outliers. Furthermore, the closeness of the real and generated two-dimensional means in Figure 4-5a suggests an effective learning by G_Φ. This closeness is quantified by JS divergence of 0.0636 and 0.3032 (out of 1.00) for active power

[1] Area under the curve is equal to 1.



(a) Real and generated distribution of $|\underline{V}|$ (in p.u.) and P (in MW) as node features in $\mathbf{X}$. The mean of the real and generated distributions is shown with a $\times$ marker.



(b) Real and Generated distribution of P^+ (in MW) as the first edge feature in $\mathbf{X}^1$.

Figure 4-5: Normalised KDE-based comparison of real and generated distributions of $|\underline{V}|$, P, and P^+ for Net 42-A, including JS divergence measure.

and voltage magnitude, respectively.

As seen in Figure 4-5b, G_Φ also identifies the dominant mode of P^+ distribution, though with some observable bias. Note that the model also captures a subtle feature near 0.5MW, indicating its sensitivity to finer details. The JS divergence of less than 0.5 for all distributions supports the notion that the GAN is learning in the right direction. However, comparatively higher divergence for $|\underline{V}|$ and P^+ suggest improvement for this setup.

Given a trained G_Φ, the generator outputs $(\bar{\mathbf{X}}, \bar{\mathbf{X}}^1)$ can be fed into the M_Ψ model for joint SPE. However, given the inadequate performance of G_Φ model, instead of $(\bar{\mathbf{X}}, \bar{\mathbf{X}}^1)$ sampled from $\mathcal{D}^{\text{ISE}}$, the real data $(\mathbf{X}, \mathbf{X^1})$ sampled from $\mathcal{D}^{\text{SE}}$ is used to independently assess the performance of M_Ψ model as discussed in the next section.

4-2-2 Performance of TapSEGNN model

This section examines the performance of the M_Ψ model for joint SPE using $\mathcal{D}^{\text{SE}}$ dataset. The configuration of M_Ψ is given in Table 4-3 for predicting the tap positions of all 18 MV/LV transformers in Net 42-A. This model is trained to minimise the loss function in (3-20) with $\lambda_{\text{reg}} = 0.001$ for 500 epochs with a learning rate scheduler having a minimum learning rate of 10^{-4}. A high number of epochs is observed in similar contexts [42] due to slow learning of the GNNs. Using this training configuration, in addition to the standard practices, to ensure that the training does not suffer from suboptimal learning rates or vanishing/exploding gradients, the gradient norm was monitored with respect to epochs as shown in Figure 4-6. Over time, the finite and gradually diminishing gradient norm suggests convergence of model parameters to either a local or a global minimum. The possibility of convergence to a saddle point is

unlikely, given that training employed the Adam optimiser with momentum, as discussed in Section 3-5.

Table 4-3: Model Configuration for M_Ψ including readout layers for 18 MV/LV Transformers

Parameters	**Value**
Input channel for GCNN	2
Hidden layer dimension for GCNN	64
Filter order for GCNN	3
Output channel for GCNN	32
Input channel for SCNN	6
Hidden layer dimension for SCNN	128
Filter order for SCNN	1
Output channel for SCNN	64
Output channel for GAT	32
Number of heads in GAT	1
Transformer edge neighbourhood (h)	1
Total parameters in M_Ψ ($\lvert\Psi\rvert$)	**17,857**

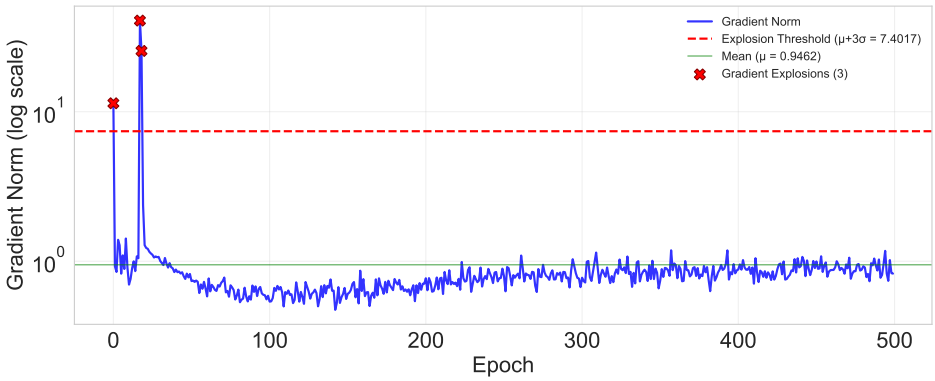


Figure 4-6: Gradient norm evolution in log-scale, training TapSEGNN model for Net 42-A over 500 epochs. Gradient explosion detected at epochs 0, 17 and 18, which commonly occurs due to initial parameter weights. Early gradient explosions are not catastrophic to model training, thereby supporting the proposed training configuration.

Moreover, for L^{TapSE} in (3-20), to study the over or under emphasis of L^{Tap} in L^{TapSE}, experiments are conducted over three values of the hyperparameter $\lambda_{\text{tap}} = [0.01, 0.1, 1.0]$ for various transformer deployment levels. The training results are presented in Table 4-4, quantifying the L^{SE} performance with root mean square error (RMSE) metric and that of L^{Tap} with prediction accuracy. A comparative analysis reveals that:

1. For $\lambda_{\text{tap}} = 0.01$: The RMSE values are lower for up to only 50% transformers indicating better SE. Accounting for more transformers results in higher RMSE than $\lambda_{\text{tap}} = 0.1$. Tap prediction accuracy is consistent across all deployment levels, achieving a 100% accuracy.

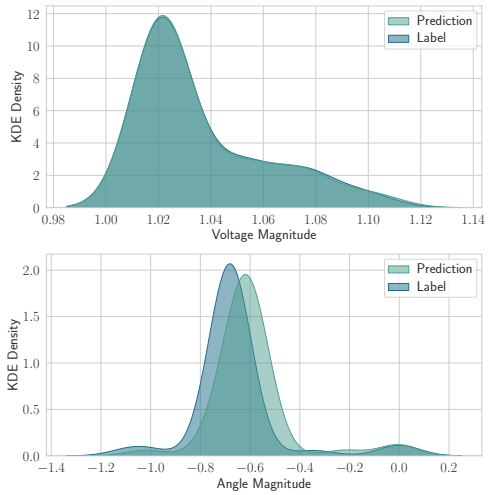
2. For $\lambda_{\text{tap}} = 0.1$: Better SE when considering more than 50% transformers. This indicates that tap loss becomes increasingly important as more transformers are accounted for in the network. Notably, the tap prediction accuracy is 100% for all cases.

3. For $\lambda_{\text{tap}} = 1.0$: Both SE and tap position prediction are significantly degraded in this setting. This behaviour highlights the critical influence of SE performance on the accuracy of tap position predictions, as it suggests that the model is not simply memorising tap classes through the transformer read-out layers. Instead, it is learning to map the complex, nonlinear relationships between the global topology–captured by the core model–and the local topology around the transformers. This supports the interpretability of the joint SPE formulation.

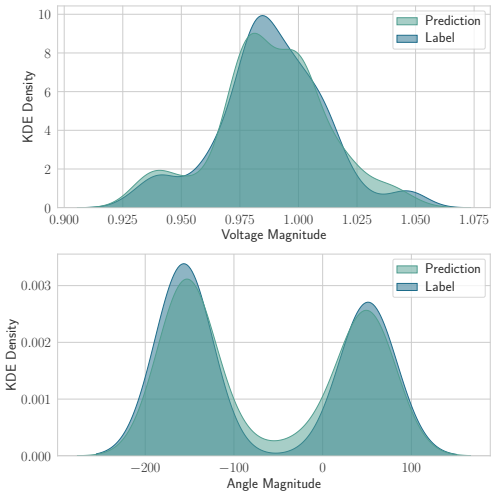
Table 4-4: TapSEGNN model performance for Net 42-A as a function of λ_{tap} across a single, 25%, 50%, 75%, and 100% of total 18 MV/LV transformers. The average tap prediction accuracy is a batch-wise average. Transformer coverage-wise, the minimum RMSE is highlighted in bold. For transformer coverage $\leq 50\%$ (approximately 9 MV/LV transformers), the model achieves better SE performance with $\lambda_{\text{tap}} = 0.01$. When coverage exceeds 50%, $\lambda_{\text{tap}} = 0.1$ yields improved SE. In contrast, $\lambda_{\text{tap}} = 1.0$ consistently results in poor performance for both SE and tap prediction.

Transformer Coverage	**Loss Tap Weight** λ_{tap}	**RMSE** $\lvert\underline{V}\rvert$ $\times 10^{-5}$ [p.u.]	**RMSE** θ $\times 10^{-4}$ [deg]	**Average Tap Prediction Accuracy** [%]
Single transformer	0.01	**2.113**	**2.811**	100
	0.1	3.531	5.404	100
	1.0	30.551	6.187	25
25% transformers	0.01	**2.346**	**2.916**	100
	0.1	2.882	4.472	100
	1.0	30.775	87.428	98.438
50% transformers	0.01	**2.5811**	**3.157**	100
	0.1	2.683	3.838	100
	1.0	36.421	10.270	41.146
75% transformers	0.01	2.987	3.782	100
	0.1	**2.812**	**3.462**	100
	1.0	33.614	7.832	49.279
100% transformers	0.01	3.146	5.800	100
	0.1	**3.028**	**4.437**	100
	1.0	40.939	18.696	28.906

These experiments analyse the effect of varying λ_{tap} values. However, since this study aims to obtain optimal performance for predicting tap positions for all transformers in the network, $\lambda_{\text{tap}} = 0.1$ is selected as the final configuration, resulting in better RMSE as compared to $\lambda_{\text{tap}} = 0.01$ and 1.0. In addition to the effect of λ_{tap}, Table 4-4 also shows that the RMSE θ is consistently poor as compared to RMSE $|\underline{V}|$. Upon visualising the predicted distributions in Figure 4-7a, it is observed that the model captures the underlying structure of the probability distribution, albeit with some bias, corresponding to high RMSE θ. Despite Figure 4-6 indicating a moderately stable training configuration, such suboptimal fitting of the angle distribution suggests a room for improvement in modelling M_{Ψ}.



(a) Predicted and label voltage magnitude (in p.u) and angle (in degrees) for Net 42-A. A bias of 0.2 degrees is observed in the angle prediction.



(b) Predicted and label voltage magnitude (in p.u) and angle (in degrees) for Net 320 testing scalability of TapSEGNN model. Some variation is observed in voltage magnitudes between 0.975 and 1.000 p.u. and angles from -100° to 0°.

Figure 4-7: Normalised KDE-based comparison of true and predicted $|\underline{V}|$ and θ for Net 42-A and Net 320. Note that the KDE density on the y-axis scales with the range of x-axis values so that the area under the curve equals 1, as expected for a normalised KDE.

In the next section, with this model and training configuration tuned for Net 42-A, training is performed on a larger network to study the scalability of the TapSEGNN model.

4-3 Scalability Analysis

In this section, the scalability of the proposed TapSEGNN model is evaluated on a larger power system network consisting of 320 buses: Net 320. The model architecture and training configuration, tuned for Net 42-A with $\lambda_{\text{tap}} = 0.1$, are employed to test scalability. The scalability results, tested across five cases, are presented in Table 4-5, aligning with the case setup in Table 4-4.

To illustrate the SE performance of TapSEGNN, the voltage magnitude and angle predictions for the final case (100% transformers accounted) are visualised in Figure 4-7b using KDE plots. It is important to note that the higher RMSE θ for Net 320 as compared to Net 42-A is due to a higher range of angle magnitude–the range of Net 42-A is about [-1.4,0.2] degrees, whereas that of Net 320 is [-200,100] degrees. Despite this expected increase in RMSE θ, RMSE $|\underline{V}|$ and tap prediction accuracy are comparable with those of Net 42-A, indicating promising scalability of the TapSEGNN model for bigger distribution grids.

The next section demonstrates the transferability of the core model in TapSEGNN, where it is trained on one network and tested on a similar network.

Table 4-5: TapSEGNN model performance for Net 320 across a single, and 100% of the total 141 MV/LV Transformers. The performance for SE and batch-wise average tap prediction shows minimal degradation from single to 100% transformer coverage. Column-wise best results highlighted in bold.

Transformer Coverage	**RMSE** $\lvert\underline{V}\rvert$ $\times 10^{-5}$ [p.u.]	**RMSE** θ [deg]	**Average Tap Prediction Accuracy** [%]
Single transformer	**2.408**	**0.373**	**100**
100% transformers	6.778	0.467	99.955

4-4 Generalisability Analysis

The work in [73] and [5] emphasises that for two networks with the same number of nodes and similar spectral density of the graph Laplacian, the signal propagation characteristics tend to be preserved. Keeping this in mind, a real network, Net 42-B, is selected with the same number of buses as Net 42-A. And the spectral radius of graph Laplacian for Net 42-A and Net 42-B are similar in magnitude: $\rho(\mathbf{L}_{0,\mathrm{A}}) = 5.099$ and $\rho(\mathbf{L}_{0,\mathrm{B}}) = 6.332$, respectively. Moreover, from a graph frequency response perspective [5], since $\rho(\mathbf{L}_{0,\mathrm{B}}) > \rho(\mathbf{L}_{0,\mathrm{A}})$, a model trained on Net 42-B *should* show better transferability on Net 42-A, instead of the other way around. Furthermore, the similarity between Net 42-A and Net 42-B can be analysed by studying the distribution of line and transformer parameters in the networks as shown in Figure 4-8 and Figure 4-9, respectively. The specific types of lines and transformers used in these two networks, highlighting the common types, are displayed in Table 4-6. It can be observed that the line parameters have more similar distributions than those for the transformer parameters. Since the simplicial complex neural network (SCNN) component in the TapSEGNN model (see (3-2)) relies on these parameters as edge features, the similar distribution of line parameters shows potential for generalisability.

To quantitatively analyse the generalisability, the focus is limited to state estimation only, discarding transformer tap position prediction. This is because the TapSEGNN model relies on $\mathcal{N}_t$; it is not theoretically transferable from one network to another with different transformer terminal bus indexing. However, the architecture of TapSEGNN up to training L^{SE} (in (3-17)) is theoretically transferable–inherently independent of the specific number of nodes and edges in a graph (not the topology itself). This part of the TapSEGNN model, discarding the transformer readout layers, is hereby referred to as the SEGNN model. Using this model, several experiments are carried out with varying $\sigma_{\mathrm{load}} = [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1.0]$ MW as shown in Table 4-7. In that table, Case 1 refers to deploying the SEGNN model trained on Net 42-A to Net 42-B, while Case 2 refers to deploying the model trained on Net 42-B to Net 42-A. Several observations emerge from Table 4-7:

1. Across nearly all levels of σ_{load}, Case 2 results in better generalisation performance than Case 1. This could be attributed to the previously highlighted property, $\rho(\mathbf{L}_{0,\mathrm{B}}) > \rho(\mathbf{L}_{0,\mathrm{A}})$, suggesting that the broader spectral content of Net 42-B enables the model to learn more expressive and transferable representations.

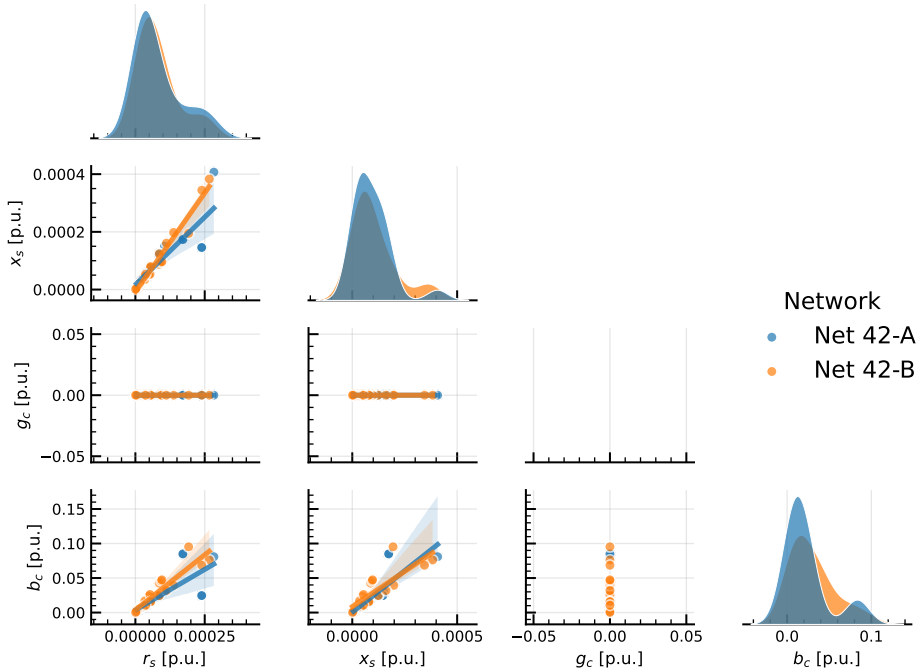


Figure 4-8: Statistical distribution of line parameters for Net 42-A and Net 42-B in p.u. values. Diagonal: normalised KDE plots show similar marginal distributions for series resistance (r_s), series reactance (x_s), and shunt susceptance (b_c); shunt conductance (g_c) is zero. Off-diagonal: Scatter plots with regression lines show positive *statistical* correlations among r_s, x_s, and b_c.

2. Higher loading perturbations, σ_{load}, tend to improve the generalisability of voltage magnitudes. For example, at $\sigma_{\text{load}} = 1.0$, the lowest RMSE for $|\underline{V}|$ ($1.489{\times}10^{-5}$) is achieved. This supports the hypothesis that higher input variation enables the model to capture the underlying nonlinearities of power flow more effectively.

   Although the RMSE θ show less consistent improvements on deployment, up to $\sigma_{\text{load}} =$ 0.1MW, RMSE θ remains in a narrow range for all Case 2s, e.g., 19.400, 19.738, 20.021 and 26.399 $\times 10^{-4}$, indicating reasonably good generalisability, albeit not as good as for voltage magnitude.

These observations warrant further investigation to draw sound conclusions regarding the generalisability of the SEGNN model. To address this, the next section trains the individual sub-components of the SEGNN model as standalone baselines, alongside other methods, to identify which parts of the model are responsible for learning the underlying data distributions. For instance, if the trained parameters weigh the node-embeddings more than the edge-embeddings, then the edge-level information is implicitly ignored. This indicates that the model is not leveraging the similarity in distribution of line parameters, as shown in Figure 4-8. These aspects are analysed in detail in the following sections.

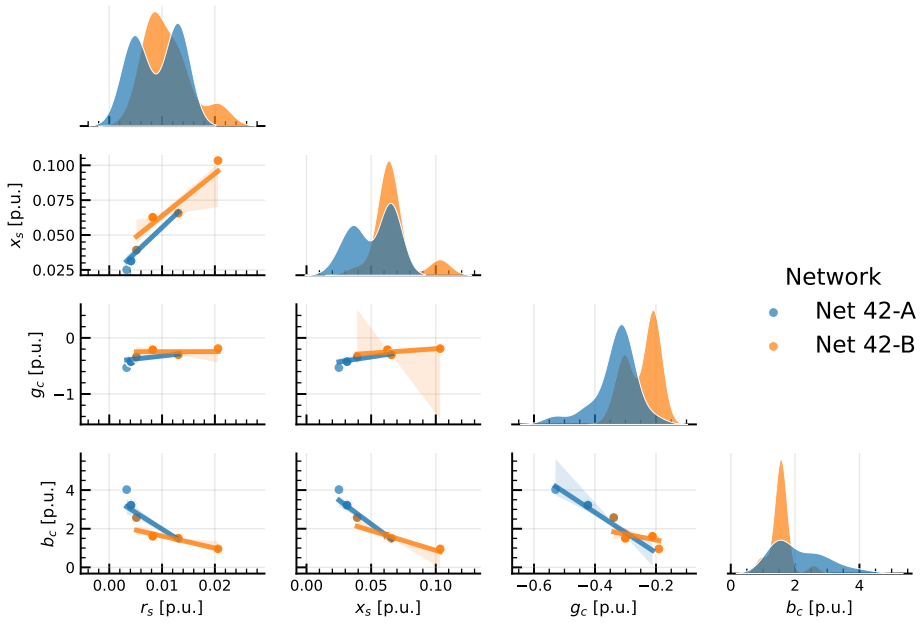


Figure 4-9: Statistical distribution of effective transformer parameters for Net 42-A and Net 42-B in p.u. values keeping the nominal tap position. Diagonal: normalised KDE plots show distinct marginals for r_s, x_s, g_c, and b_c. Apart from some mode alignment for x_s and b_c, these distributions show no real similarity. Off-diagonal: scatter plots with regression lines indicate positive correlation between x_s and r_s, negative correlation of b_c with all other parameters, and weak correlation of g_c with r_s and x_s. Note that these trends are statistical, as r_s and x_s depend on winding design, while g_c and b_c relate to core losses.

4-5 Comparison of Baseline Methods

In this section, the proposed core model architecture is empirically compared with a few baseline methods to evaluate the theoretical advantages it offers. The focus is limited to SE performance using SEGNN model, because this part of the model contains the largest number of parameters, and the transformer readout layers in $\text{Tap}(\mathbf{X}^o)$ (from (3-13)) can be flexibly concatenated or reused across different core architectures. Additionally, SE performance strongly impacts tap position prediction accuracy, as shown in Figure 4-4 for $\lambda_{\text{tap}} = 1.0$, making it the primary basis for comparing the core model with baseline methods. From a learning standpoint, since the GAN framework relies on implicit density modelling, which reduces interpretability, baseline methods are not used for comparison for the task of imputing missing measurements. Along this line, since the G_Φ model in this work inherently uses the core model architecture, the regression performance should positively correlate with the generative capabilities. However, this relationship may not always hold.

Some baseline methods are selected as sub-models of the SEGNN to investigate whether other components–particularly the SCNN layer from (3-2)–contribute to enhancing the overall performance of the SEGNN model. The configuration of these sub-models is kept as proposed in Table 4-3 unless otherwise specified. To ensure a rigorous evaluation of these models, experiments were designed to assess the aspect of scalability and generalisability in addition to SE. The baseline methods selected in this work are:

Table 4-6: Line and transformer types used in Net 42-A and Net 42-B. The common labels in both networks are highlighted in bold. Line label, for example, 3×1*240 AL XLPE 12/20 trefoil, means 3-phase single-core cables with a cross-section of 240mm^2 and an aluminium conductor insulated with cross-link polyethene, with voltage ratings of 12kV phase-to-ground and 20kV phase-to-phase. And 23/0.420 V - 630 kVA indicates an MV/LV transformer stepping down voltage from 23 kV to 420 V with a power rating of 630 kVA.

Component	Net 42-A	Net 42-B
Lines	**3×1*240 AL XLPE 12/20 trefoil** **3×1*400 AL XLPE 18/30 trefoil** 3×1*150 AL XLPE 12/20 trefoil	**3×1*240 AL XLPE 12/20 trefoil** **3×1*400 AL XLPE 18/30 trefoil** 3×1*400 AL XLPE 12/20 trefoil
Transformers	**23/0.420 kV – 630 kVA** **23/0.420 kV – 1000 kVA** **23/0.420 kV – 1600 kVA** 23/0.420 kV – 2000 kVA 23/0.420 kV – 2500 kVA	23/0.420 kV – 400 kVA **23/0.420 kV – 630 kVA** **23/0.420 kV – 1000 kVA** **23/0.420 kV – 1600 kVA**

1. Conventional Weighted Least Squares (WLS): Using the Gauss-Newton method as discussed in Section 2-2-1.
2. Topology-agnostic fully connected neural network (FCNN): This architecture contains 21,716 model parameters.
3. Topology-aware neural networks:
   (a) Graph Attention Network (GAT): This model directly captures the node and edge features to calculate attention. It contains 450 parameters.
   (b) Graph Convolutional Neural Network followed by GAT (GCNN+GAT): This model first uses the GCNN architecture to get high-dimensional node-embeddings, and then these node-embeddings are used by GAT along with the edge-features. The model architecture contains 4,194 parameters.
   (c) GCNN and Edge-regression with Linegraph Laplacian followed by GAT (GCNN + LGL + GAT): This architecture builds on top of GCNN+GAT, where it calculates the high-dimensional edge-embeddings from edge-features, which are then input to the GAT architecture. In this architecture, the SCNN layer in the core model architecture can be replaced by introducing the linegraph Laplacian instead of the Hodge-Laplacian. The linegraph Laplacian is the graph Laplacian of the edge-to-node dual graph of the original undirected graph $\mathcal{G}$ [34]. This model contains 15,970 parameters using configuration from Table 4-3.

Using these models, Table 4-8 presents a comparison for SE under two levels of load variability $\sigma_{\text{load}} = [0.1, 1.1]$ MW. Although $\sigma_{\text{load}} = 1.1$ MW will not represent the practical operating conditions, it serves as a more rigorous benchmark to assess the capability of the models to learn the underlying nonlinear relationships. Another test to investigate whether the model is learning the nonlinear relationship underlying the data is to compare the generalisability (as discussed in Section 4-4) of the models, which is presented in Table 4-9. In addition to these

Table 4-7: Performance of SE model for generalizability. Column-wise minimum RMSE is highlighted in bold.

σ_{load}	Case	Training		Deployment	
		RMSE $\lvert\underline{V}\rvert$ $\times 10^{-5}$ [p.u.]	**RMSE** θ $\times 10^{-4}$ [deg]	**RMSE** $\lvert\underline{V}\rvert$ $\times 10^{-5}$ [p.u.]	**RMSE** θ $\times 10^{-4}$ [deg]
0.0001	1	1.759	2.855	33.684	320.862
	2	1.153	**1.738**	4.417	**19.400**
0.001	1	1.745	2.600	35.175	328.640
	2	1.179	2.284	6.795	19.738
0.01	1	1.221	3.144	28.488	315.720
	2	1.369	2.861	12.712	20.021
0.1	1	1.695	13.236	24.946	369.531
	2	**1.145**	4.994	1.725	26.399
1.0	1	2.045	111.507	2.377	102.180
	2	1.564	66.463	**1.489**	128.907

tests, the scalability factor for practical large-scale networks is also evaluated. Table 4-10 presents this comparison using Net 320 as the network. A method-specific evaluation across these three tests is discussed next.

The Gauss-Newton method-based WLS consistently shows poor performance in terms of SE and scalability compared to other models due to linearisation errors. Moreover, the reliance of WLS on an accurate network model constrains its applicability for joint SPE as discussed in Section 2-2 with other aspects. And since the WLS approach relies on a model-specific Jacobian matrix, it does not permit assessing the generalisability.

The FCNN model inherently lacks inductive biases such as parameter sharing, which are leveraged in graph convolution operations (see Section B-0-1). As a result, the FCNN relies entirely on dense matrix operations to learn the explicit mappings between input and output. This enables a high representational capacity, which, under low variability conditions ($\sigma_{\text{load}} = 0.1$ MW), allows the FCNN to achieve the lowest RMSE for both voltage magnitude and angle. However, this same capacity becomes a liability in terms of generalisation. Under higher variability ($\sigma_{\text{load}} = 1.1$MW), a degradation in RMSE for voltage magnitude is observed. Further evidence supporting the lack of generalisability of FCNN is provided in Table 4-9. Despite achieving low RMSE values when trained on Net 42-B, the model shows a sharp degradation in performance when deployed for Net 42-A, resulting in one of the worst generalisation results among all other models.

Across all three tests, the graph attention neural network (GAT) consistently demonstrates poor performance compared to other baseline models. An initial hypothesis attributes this underperformance to the limited capacity of the model, reflected in the relatively low number of trainable parameters–only 450–as compared to other baseline models, which contain parameters in the order of 10^3. To verify this, the number of attention heads and hidden units in the GAT architecture was increased to match the parameter count with the GCNN+GAT model, resulting in 5,378 parameters. Despite this adjustment, no notable improvement was

Table 4-8: Comparison of baseline models for SE. Column-wise minimum RMSE is highlighted in bold. (*) indicates that all models are trained for 200 epochs due to the slow learning rate observed from the gradient norm over time.

σ_{load} [MW]	**Model**	**RMSE** $\lvert\underline{V}\rvert$ **[p.u.]** $\times 10^{-5}$	**RMSE** θ **[deg]** $\times 10^{-4}$
0.1	WLS	13.098	161.351
	FCNN	**0.348**	**3.002**
	GAT	2.198	19.672
	GCNN+GAT	1.588	11.920
	GCNN+LGL+GAT	1.485	12.254
	Proposed	1.760	11.925
1.1*	WLS	14.408	126.187
	FCNN	3.755	**35.980**
	GAT	2.540	200.527
	GCNN+GAT	**1.528**	101.056
	GCNN+LGL+GAT	1.714	104.078
	Proposed	1.599	105.994

observed in its performance, suggesting that the issue lies not in model capacity but in its structural design. Reconsidering Section 2-4-2, the attention weights in GAT are computed from terminal nodes only. This design inherently restricts the ability of the model to capture 1-hop and multi-hop neighbourhood information. This is conceptually complemented by the graph convolutional neural network (GCNN) architecture.

The GCNN architecture effectively captures spatial dependencies by aggregating information from deeper neighbourhoods to compute high-dimensional node embeddings. When these enriched embeddings are subsequently fed into the GAT layer, they implicitly provide the GAT with access to multi-hop neighbourhood information, thus addressing the inherent limitation of the standalone GAT. This combination enables the GCNN+GAT model to benefit from both global and local contexts of the network. As a result, the GCNN+GAT architecture consistently outperforms the standalone GAT model, as demonstrated in Table 4-8, Table 4-9, and Table 4-10.

While the node-level embeddings capture spatial dependencies effectively, Section 2-3-2 motivates extending this to edge-level representations using frameworks such as line graph Laplacians and simplicial complexes to model edge interactions, such as power flows. However, empirical results show that architectures incorporating these embeddings–GCNN+LGL+GAT and the proposed model–do not outperform the simpler GCNN+GAT model. This suggests that either the node embeddings already capture sufficient spatial structure or the edge embedding methods fail to align with the underlying spatial dynamics of power networks, resulting in limited utility. Another possible explanation is that, since SE is inherently a node-level task, the advantages of edge-level embeddings may remain untapped unless applied to an edge-specific learning objective. A physics-aware soft constraint over the edge encodings could potentially warrant an improved performance of GCNN+LGL+GAT or the proposed model.

Table 4-9: Comparison of baseline models for generalisability. Column-wise minimum RMSE is highlighted in bold. (*) indicates that all models are trained for 200 epochs due to the slow learning rate observed from the gradient norm over time.

		Training on Net 42-B		**Deploying on Net 42-A**	
σ_{load} [MW]	**Model**	**RMSE** $\lvert\underline{V}\rvert$ **[p.u.]** $\times 10^{-5}$	**RMSE** θ **[deg]** $\times 10^{-4}$	**RMSE** $\lvert\underline{V}\rvert$ **[p.u.]** $\times 10^{-5}$	**RMSE** θ **[deg]** $\times 10^{-4}$
1.1*	FCNN	2.133	10.974	6.635	179.392
	GCNN+GAT	**1.762**	77.313	**1.562**	**131.778**
	GAT	2.641	222.509	2.730	205.292
	GCNN+LGL+GAT	1.940	77.118	1.733	132.607
	Proposed	1.952	**77.104**	1.908	132.156

Table 4-10: Comparison of baseline models for scalability. Column-wise minimum RMSE is highlighted in bold. (*) indicates that all models are trained for 200 epochs due to the slow learning observed from the gradient norm over time.

σ_{load} [MW]	**Model**	**RMSE** $\lvert\underline{V}\rvert$ **[p.u.]** $\times 10^{-5}$	**RMSE** θ **[deg]** $\times 10^{-4}$
0.3*	WLS	16.673	869.749
	FCNN	13.491	**3.021**
	GCNN+GAT	**3.945**	66.781
	GAT	4.734	198.702
	GCNN+LGL+GAT	4.052	71.095
	Proposed	4.010	80.075

The next section sums up the main insights from all the experiments discussed in this chapter.

4-6 Summary and Discussion

The proposed framework is evaluated through various experiments that assess the models from multiple perspectives. It begins with generating snapshots of the system states. These snapshots are then classified into separate datasets used by different models in the framework. Next, the performance of the generative model is evaluated employing balanced adversarial training of the GAN setup. The generative model successfully captures the underlying distribution of active power, but shows less accuracy for the voltage magnitude and active power flow at the edges, identifying the major modes with some bias.

Following that, the performance of TapSEGNN is investigated, with a focus on its accuracy and scalability. Its training demonstrates convergence with diminishing gradients, indicating that the model likely reaches a local or global minimum. Notably, when the tap position prediction objective is over-emphasised compared to the SE objective, the accuracy of both tap predictions and SE declines. This shows that the transformer readout layers do not memorise the class labels and are highly influenced by the performance of SE to optimise the

overall performance. The scalability of TapSEGNN for a larger network with 141 MV/LV transformers is tested, demonstrating promising performance with approximately 99.95% tap prediction accuracy. Furthermore, the generalisability of TapSEGNN for the objective of SE is evaluated by examining the spectral properties of the graph Laplacians and the distribution of line and transformer parameters across the two networks. The findings suggest that GNNs generalise better when they're learned on networks with higher spectral density (in terms of the graph Laplacian) and then applied to networks with lower spectral density.

Finally, a comparison of the core model with alternative architectures reveals that all the GNN-based models perform similarly, except for GAT, which lags behind. Among these, the combined GCNN+GAT model achieves slightly better results. This marginal improvement suggests that the model either places greater emphasis on node-level information or that node-level features alone are sufficient to capture the spatial dynamics of power flow across the network. Despite all this, the core model consistently outperforms the traditional weighted least squares (WLS) method by 10 times, further justifying the motivation for adopting emerging graph-based approaches, as discussed in Section 2-2-2.

These findings inform the conclusions, address the limitations, and guide future recommendations discussed in the next chapter.

Chapter 5

Conclusion

This is the final chapter of this report, providing a concise summary of all the work undertaken and highlighting the main takeaways. Based on these insights, it addresses the research question posed in Section 1-3. Finally, it offers recommendations for future work to address the limitations of this study and proposes directions that could complement and extend this research.

5-1 Summary

This thesis project aimed to address the problem of SPE in unobservable DGs with a focus on data-driven topology-aware machine learning techniques. The primary motivation for this work stems from both academic and industrial perspectives, highlighting the need for an efficient and scalable approach.

The first chapter sets the stage by explaining the global challenges caused by the ongoing congestion, underlining the critical impact on the situational awareness of the DGs. It introduces the recent developments in machine learning and explains how these advances can help tackle the issue of state and transformer tap position estimation in unobservable DGS. By separating the main issue into two parts–addressing the challenges of an unobservable grid and predicting accurate state and tap position estimates–this chapter proposes the use of generative models and GNNs, given their recent success in related domains. It then frames the problem as a clear research question and outlines the main contributions that this thesis aims to make.

The second chapter establishes the theoretical foundations, supported by an analysis of practical networks, to provide an understanding of the contributions of this work. It employs standard mathematical modelling practices for components in the power grid, resulting in the formulation of the PF problem and measurement model for SPE. Based on the measurement model, the reader is then introduced to both conventional and emerging approaches for SPE. By highlighting the limitations of conventional Gauss-Newton methods and physics-agnostic neural networks, the discussion motivates a shift towards physics-informed neural networks,

with a particular emphasis on topology-aware architectures, namely GNN. Finally, the chapter addresses the observability challenges in the DG by proposing the use of generative models, with major emphasis on the generative adversarial network (GAN) architecture, to impute missing measurements across the grid.

Chapter 3 proposes a framework utilising the twin forces of generative models and GNNs to address the unobservability and improve SPE. It introduces a hybrid graph representation that captures both node and edge-level spatial dependencies. This graph not only models the physical structure of the network but also incorporates the practical measurement setup. Using this graph representation, the core model architecture is designed, which applies convolution operations on nodes and edges with GCNN and SCNN, respectively. This core model is refined as a generative model to generate or impute missing measurements as node and edge features in the graph representation. This generative model is trained through an adversarial setup against a discriminative model that classifies real graphs from fake ones.

Beyond imputation, the same core model is extended by adding transformer readout layers. This extension, referred to as the TapSEGNN model, enables joint state estimation and transformer tap position prediction. Each of these setups–one for GAN and the other for TapSEGNN–has its specific objective function. This chapter concludes by justifying the training strategies proposed to optimise these objective functions.

In Chapter 4, the proposed framework is evaluated through experiments that generate system state snapshots, classify them into datasets, and assess each model's performance. The generative model, trained with balanced adversarial learning, accurately captures the active power distribution but shows slight bias in voltage magnitude and edge power flow. TapSEGNN demonstrates good accuracy, convergence, and scalability, with results showing that overemphasising tap position prediction degrades overall performance, highlighting the benefit of balanced objectives. Its generalisability is supported by the properties of the graph Laplacian and analyses of line and transformer parameters. Finally, core model comparisons show that all GNN-based models perform similarly, with GCNN+GAT slightly better than others, and all outperform the traditional WLS method, reinforcing the value of adopting graph-based approaches discussed in Section 2-2-2.

5-2 Answer to the Research Question

Reconsidering the central research question:

How can graph-based methods effectively leverage topological information to perform generalizable, computationally efficient, and joint state and transformer tap position estimation in electric networks with limited observability?

To address this question, the limited observability aspect is treated as a separate solution. It is independently developed from the method for joint state and transformer tap position estimation, maintaining both solutions graph-based.

The challenge of limited observability is addressed through the integration of a state-of-the-art generative framework of GANs. Since the real grids have sparse measurements, deriving virtual or pseudo-measurements from the known quantities becomes solving an underdetermined problem. The GAN framework constrains the infinite solution space by leveraging the

synthetic data and network topology using a graph-based core model architecture in the G_Φ model. As a result, the trained G_Φ performs reasonably well in learning the full distribution of active power injection at buses and capturing the modes in the bus voltage magnitudes and active power flows in the network.

For joint state estimation and predicting transformer tap positions, the proposed work demonstrates that graph-based methods can indeed exploit network topology to achieve more accurate state estimation compared to the conventional WLS method. Moreover, it shows superior performance in generalisability across a similar network as compared to the physics-agnostic FCNN architecture. The inherent properties of GCNN and GAT architectures, such as parameter sharing and localised neighbourhood aggregation, ensure that the computational cost scales linearly with the network size–specifically, with the number of edges–thereby maintaining computational efficiency.

5-3 Recommendations for Future Work

This section outlines the limitations of this work and proposes a few future recommendations to address them. Tracing back the suboptimal performance of the GANs in Figure 4-5 at learning the distribution of voltage magnitude $|\underline{V}|$ and active power flow P^+, the author suggests complementing the current GAN setup by:

- Adding feature-specific loss functions and independently weighing them for both the generator and discriminator models. This allows the models to prioritise and better fit the marginal distributions of $|\underline{V}|$ and P^+ separately.
- Incorporating moment matching losses or Maximum Mean Discrepancy (MMD) measures for $|\underline{V}|$ and P^+ within the objective functions of both the generator and discriminator. For example, an L_2 norm penalising the discrepancy between the first and second order moments (mean and standard deviation) of the generated versus real distributions of $|\underline{V}|$, inspired by the approach in [74], can be effective in improving statistical alignment.

In addition to the suggested improvements to the current GAN formulation, other ideas can be explored to enhance physical consistency in the generated snapshots, as outlined below.

1. **Physics-informed Uncertainty Quantification (UQ)**: As the name suggests, UQ is the technique to quantify the uncertainty in the quantity of interest for any mathematical model. With deep learning, UQ is assessed by propagating the uncertainties in the input parameters through a model. Combining UQ with physics-informed objectives in GANs could enable the generator model to learn the underlying distribution of data, rather than relying on point estimates. This not only ensures producing physically consistent and generalisable solutions, but also theoretically resolves the imbalance of gradients in GANs. A seminal work by Daw et al. in [75] tests these ideas on real-world datasets with both ideal and imperfect physics, as well as benchmark datasets involving partial differential equations. Complementing the proposed work with PID-GANs can make the adversarial training more balanced and accurate.

2. **Objective-Reinforced GANs for Foundation Models**: In addition to generating physically consistent network snaphots, as discussed in the previous idea, it is also valuable to generate snapshots that meet specific operational criteria, such as N-1 contingency compliance, acceptable short-circuit levels, balanced line-loading, and adherence to thermal limits, among others. Evaluating these operational properties is straightforward and more reliable when performed using specialised external tools, such as PSS ([76]) and PowerFactory ([77]). Although these tools cannot be directly embedded into machine learning models, they can still be interfaced as non-differentiable metrics by employing a reward network within a reinforcement learning framework. In this approach, a sample generated by the model is assessed externally, and the reward network learns to align its evaluations with the scores provided by the external tools. A similar concept has been successfully applied in the context of graph-based GANs for molecular synthesis, as demonstrated by Cao et al. in MolGAN [78]. Extending this idea could help develop more robust and realistic foundation models for electrical grids, as proposed in [79].

Now, recalling the discussion from the comparison of baseline models, it was observed that the proposed core model in TapSEGNN and the GCNN+LGL+GAT architecture–which incorporates edge-level embeddings in the learning process–did not demonstrate improvements compared to models relying solely on node-level learning, such as GCNN+GAT. A plausible hypothesis is that since the primary objectives of SE are node-focused, specifically through the loss term L^{SE} defined in (3-17), the model primarily utilises node-level information during training. Consequently, edge-level embeddings may not be effectively leveraged.

To investigate this hypothesis, the following approach is proposed:

3. **Incorporate edge-level objectives during training:** To encourage the model to learn meaningful edge-level representations explicitly, soft constraints can be imposed on the edge embeddings. For instance, the output edge embeddings may be designed to match the dimensionality corresponding to the number of known parameters of lines or transformers, which is equal to 4 in this work. These known physical parameters can be incorporated as soft targets in the training objective, encouraging the core model architecture to capture and utilise edge information explicitly. By imposing such soft constraints on the edge embeddings, the overall performance can be potentially enhanced.

Finally, from an implementation perspective, several important considerations must be made to compute the Hodge–Laplacians in SCNNs efficiently and to optimise the forward and backward propagation of the transformer readout layers. These aspects are reiterated and discussed in the project's repository [[16]].

Summarising this work, given the growing importance of SE for DGs and the advent of generative models, these ideas and improvements present exciting opportunities to advance the field and unlock more reliable modelling approaches.

Appendix A

Iterative Solution Methods

The nature of the PF equations is nonlinear due to the violation of the superposition principle, indicating that the sum of the inputs does not result in the sum of their individual outputs. This nonlinearity arises from the presence of second-order polynomial terms combined with trigonometric functions in the equations. As a result, analytical solutions are not feasible, and traditional methods such as graphical analysis, substitution, or elimination are unsuitable due to the high number of variables involved. Therefore, iterative root-finding methods are conventionally employed.

The general idea of these methods for a nonlinear vector-valued function $\boldsymbol{g}(\mathbf{q}) = 0$, involves starting with an initial estimate $\mathbf{q}_0$, evaluating $\boldsymbol{g}(\mathbf{q}_0)$, and systematically refining $\mathbf{q}_0$ and $\boldsymbol{g}(\mathbf{q}_0)$ to obtain subsequent estimate $\mathbf{q}_1$ that is closer to the true solution $\mathbf{q}^*$ than $\mathbf{q}_0$. These iterations continue until the distance between the estimate and the true solution is not below some tolerance, which is defined by a stopping criterion. Of the many techniques used for solving the power flow (PF) problem, the Gauss-Seidel (GS) and Newton-Raphson (NR) methods are the most prominent and discussed below.

A-0-1 Gauss-Seidel Method

GS method is a fixed-point iteration algorithm. It formulates the PF equations such that it solves for a fixed point as the solution of the equations.

Definition 2. *([80]) For a real vector-valued function $\boldsymbol{g} : D \subset \mathbb{R}^n \to \mathbb{R}^n$, if $\boldsymbol{g}(\mathbf{q}) = \mathbf{q}$, for some $\mathbf{q} \subset D$, then $\mathbf{q}$ is said to be a fixed-point of $\boldsymbol{g}$.*

For this method, the steady-state state-vector can be defined with respect to each bus as,

$$\mathbf{q} = \begin{bmatrix} [\mathbf{q}]_1 \\ [\mathbf{q}]_2 \\ \vdots \\ [\mathbf{q}]_n \end{bmatrix} = \begin{bmatrix} |\underline{V}_1| & \theta_1 & P_1 & Q_1 \\ |\underline{V}_2| & \theta_2 & P_2 & Q_2 \\ |\underline{V}_3| & \theta_3 & P_3 & Q_3 \\ \vdots & \vdots & \vdots & \vdots \\ |\underline{V}_n| & \theta_n & P_n & Q_n \end{bmatrix} \tag{A-1}$$

- For PQ Bus: $|\underline{V}|$ and θ are unknown,

$$[\mathbf{q}]_i = \begin{bmatrix} q_{i1} & q_{i2} & P_i & Q_i \end{bmatrix}$$

- For PV Bus: θ and Q are unknown,

$$[\mathbf{q}]_i = \begin{bmatrix} |\underline{V}|_i & q_{i2} & P_i & q_{i4} \end{bmatrix}$$

- For Slack Bus: P and Q are unknown,

$$[\mathbf{q}]_i = \begin{bmatrix} |\underline{V}|_i & \theta_i & q_{i3} & q_{i4} \end{bmatrix}$$

The structure of (2-15), (2-16), (2-17) and (2-18) allows to explicitly isolate each variable ($|\underline{V}|, \theta, P$ and Q) as $\mathbf{q} = \boldsymbol{g}(\mathbf{q})$ with a vector-valued function $\boldsymbol{g}(\cdot)$. For this method at each bus, $\boldsymbol{g}(\cdot)$ can be defined for:

- Bus i as PQ Bus: Using (2-15) and (2-16) only to update q_{i1} and q_{i2}, keeping P_i and Q_i as it is; $\mathbf{q}_i^{k+1} \leftarrow \boldsymbol{g}_i(\mathbf{q}^k)$
- Bus i as PV Bus: Using (2-16) and (2-18) only to update q_{i2} and q_{i4}, keeping $|\underline{V}_i|$ and P_i as it is; $\mathbf{q}_i^{k+1} \leftarrow \boldsymbol{g}_i(\mathbf{q}^k)$
- Bus i as Slack Bus: Using (2-17) and (2-18) only to update q_{i3} and q_{i4}, keeping $|\underline{V}_i|$ and θ_i as it is; $\mathbf{q}_i^{k+1} \leftarrow \boldsymbol{g}_i(\mathbf{q}^k)$

Typically, the initial guess is $|\underline{V}| = 1.0$ pu and $\theta = 0$ for unspecified angles and voltages for power flow analysis (PFA). Then they are iteratively updated until either the changes fall below a specified tolerance or a maximum number of iterations is reached. If convergence isn't achieved within the limit, the algorithm reports failure.

A-0-2 Newton-Raphson Method

Tracing back to the general approach of iterative methods from the introduction of this chapter, in NR method, the systematic refinement between $\mathbf{q}_0$ and $\boldsymbol{g}(\mathbf{q}_0)$ is performed by using the first-order derivative of $\boldsymbol{g}(\mathbf{q}_0)$. For NR method, consider a network with n buses with one slack bus (as a reference) having m PQ buses, therefore, $n - m - 1$ PV buses. In this network with,

- bus 1 as Slack Bus: As voltage and angles are fixed, and typically defined $|\underline{V}_1| = 1$ pu and $\theta_1 = 0^\circ$.
- bus i as PQ Bus with $2 \leq i \leq (m+1)$: The expression for known active and reactive power equations can be rewritten to calculate the mismatch [18] as

$$\begin{aligned} \Delta P_i &= P_i^{\text{specified}} - |\underline{V}_i| \sum_{k=1}^{n} |\underline{V}_k| \left(g_{ik} \cos\theta_{ik} + b_{ik} \sin\theta_{ik}\right) \\ &:= f_{i,P}(|\underline{V}_1|, \theta_1, |\underline{V}_2|, \theta_2, ..., |\underline{V}_n|, \theta_n) \quad \text{for} \quad 2 \leq i \leq (m+1). \end{aligned} \tag{A-2}$$

Similarly,

$$\begin{aligned}\Delta Q_i &= Q_i^{\text{specified}} - |\underline{V}_i| \sum_{k=1}^{n} |\underline{V}_k| \left(g_{ik} \sin\theta_{ik} - b_{ik} \cos\theta_{ik}\right) \\ &:= f_{i,Q}(|\underline{V}_1|, \theta_1, |\underline{V}_2|, \theta_2, ..., |\underline{V}_n|, \theta_n) \quad \text{for} \quad 2 \leq i \leq (m+1), \end{aligned} \tag{A-3}$$

- bus i as PV bus with $(m+2) \leq i \leq n$: The voltage is fixed and only expression of active PF is used.

$$\begin{aligned}\Delta P_i &= P_i^{\text{specified}} - |\underline{V}_i| \sum_{k=1}^{n} |\underline{V}_k| \left(g_{ik} \cos\theta_{ik} + b_{ik} \sin\theta_{ik}\right) \\ &:= g_{i,P}(|\underline{V}_1|, \theta_1, |\underline{V}_2|, \theta_2, ..., |\underline{V}_n|, \theta_n) \quad \text{for} \quad (m+2) \leq i \leq n, \end{aligned} \tag{A-4}$$

Writing in matrix form,

$$\mathbf{f}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) = \begin{bmatrix} f_{2,P}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ f_{3,P}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ \vdots \\ f_{m+1,P}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ f_{2,Q}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ f_{3,Q}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ \vdots \\ f_{m+1,Q}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ g_{m+2,P}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ g_{m+3,P}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \\ \vdots \\ g_{n,P}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) \end{bmatrix} \tag{A-5}$$

where $\mathbf{f} : \mathbb{R}^{2n} \to \mathbb{R}^{n+m-1}$, $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^T$ and $|\underline{\mathbf{V}}| = [|\underline{V}_1|, |\underline{V}_2|, \ldots, |\underline{V}_n|]^T$. Now, the goal of NR is to solve $\mathbf{f}(\boldsymbol{\theta}, |\underline{\mathbf{V}}|) = \mathbf{0}$.

The NR method applies the Taylor series expansion to solve the problem by iteratively linearizing the nonlinear PF equations around the current estimate using the Jacobian matrix, $\mathbf{J} \in \mathbb{R}^{(n+m-1)\times(2n)}$ [18]. At each step, it solves a linear system to update voltage magnitudes and angles. Similar to GS, the process continues until the mismatch function is below a specified tolerance or the maximum number of iterations is reached. If convergence is not achieved, the method reports failure.

A-0-3 Limitations of Iterative Methods

One of the main limitations of iterative methods is convergence. The order of convergence can be defined as,

Definition 3. *Let q_n be a sequence that converges to q^*, where $q_n \neq q^*$. If constants $\beta, \nu > 0$ exist such that*

$$\lim_{n\to\infty} \frac{|q_{n+1} - q_n|}{|q_n - q|^{\nu}} = \beta. \tag{A-6}$$

Then the sequence is said to converge with order ν and constant β.

Definition 4. *The sequence q_n is said to be linearly convergent if q_n converges to q^* with order $\nu = 1$, for constant $\beta < 1$.*

Definition 5. *The sequence q_n is said to be quadratically convergent if q_n converges to q^* with order $\nu = 2$, for constant $\beta < 1$.*

In other words, if the update in the $(n + 1)$-th iteration is proportional to the error in the n-th iteration, the convergence is considered linear. The GS method exhibits linear convergence, requiring that $\boldsymbol{g}'(\mathbf{q}^*) < 1$. On the other hand, the NR method is quadratically convergent, meaning the update in the $(n + 1)$-th iteration is proportional to the square of the error in the n-th iteration. Consequently, the NR method converges faster than the GS method. However, this faster convergence comes with increased complexity, as Newton's method requires calculating the Jacobian matrix at each iteration. Nonetheless, because the Jacobian is sparse, utilising sparse matrix operations such as decomposition and forward-backwards substitution [9] can provide better performance compared to the GS method. On a final note, the solutions by these methods are approximate and only locally optimal because of the non-convex nature of the equations.

Appendix B

Graph-Signal Processing

graph signal processing (GSP) adapt methods such as filtering and the Fourier transform, from algebraic signal processing to non-Euclidean domains [81]. The data indexed on nodes (and sometimes edges) is referred to as a graph signal $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$ and the space of all graph signals on set $\mathcal{V}$ is defined as $\mathbb{X}^{\mathcal{V}}$. The systems that process these signals, preserving or extracting relevant information, are known as graph filters. Graph filters extend the concept of filtering from discrete-time signals to graph-structured data, as discussed in [5]. The animation in Figure B-1 illustrates how graph filters can serve as a generalisation of finite impulse response (FIR) filters.

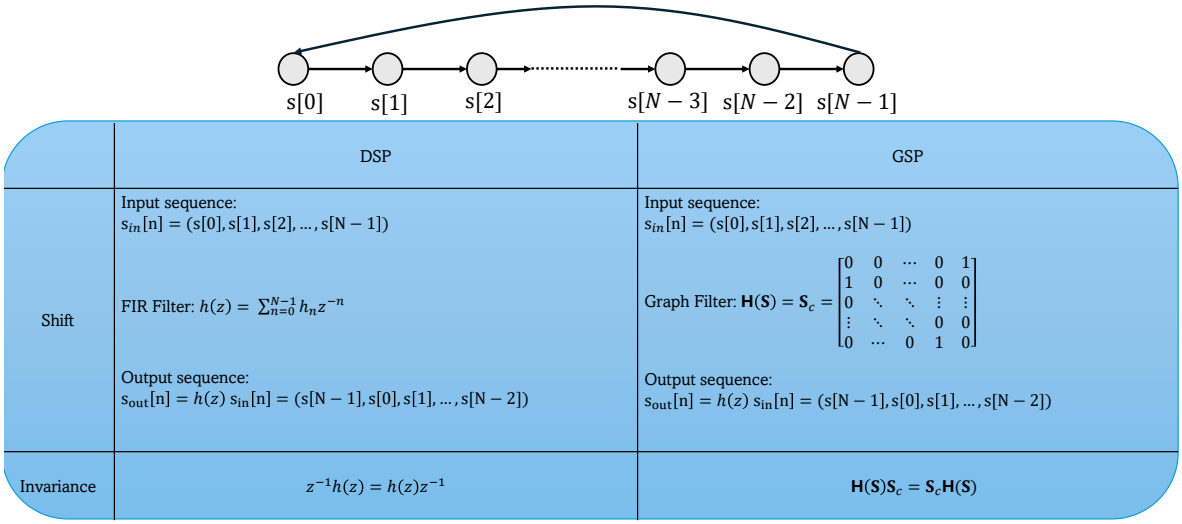


Figure B-1: Animation illustrating the equivalence between DSP and GSP for a finite impulse response (FIR) filter: $h(z)$ (with z-transform) and its graph filter counterpart $H(\mathbf{S})$ (using a circulant matrix $\mathbf{S}_c$). The invariance demonstrates that a shift followed by filtering is equivalent to filtering followed by a shift for $h(z)$ and $H(\mathbf{S})$ filter.

Further, within the graph domain, the discrete Fourier transform (DFT) can be interpreted as a projection of the graph signal (as a temporal signal) onto the eigenvectors of the cyclic

graph adjacency matrix $\mathbf{S}_c$, similar to Figure B-1. This insight characterises the frequency response of the graph filters and aids in tailoring the filter architecture for a desired spectral response. Note that the frequency here can be viewed as the variability of the graph signal. To quantify it, quadratic variation (QV) can be given as

$$\begin{aligned} \mathrm{QV}(\mathbf{x}) &= \mathbf{x}^T\mathbf{L}\mathbf{x} \\ &= \frac{1}{2}\sum_{i\in\mathcal{V},j\in\mathcal{N}(i)} [\mathbf{L}]_{ij}(x_i - x_j)^2. \end{aligned} \tag{B-1}$$

It measures how much the signal at node i differs from the signals at its neighbouring nodes. Smoother the signal $\mathbf{x}$, lower the $\mathrm{QV}(\mathbf{x})$. Such a spectral perspective allows for designing a filter by capturing the modes of variability present in a graph signal. Similar to how the forward DFT decomposes signals into sinusoidal components, the graph Fourier transform (GFT) decomposes a graph signal into the eigenvectors of a graph shift operator (GSO). These eigenvectors, which act as orthonormal basis functions, represent different frequency components of the graph signal. The GFT can be defined as below.

Definition 6. *(Graph Fourier Transform). Given the eigen-decomposition of the GSO* $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$ *with eigenvectors* $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_{|\mathcal{V}|}]$ *and eigenvalues* $\mathbf{\Lambda} = diag(\lambda_1, ..., \lambda_{|\mathcal{V}|})$*, the GFT of a signal* $\mathbf{x}$ *is defined as*

$$\mathbf{x}^{GFT} = \mathbf{V}^{-1}\mathbf{x}, \tag{B-2}$$

and the inverse GFT is defined as $\mathbf{x} = \mathbf{V}\mathbf{x}^{GFT}$.

Thus, $\tilde{\mathbf{x}}$ represents the coefficients of contribution for each frequency component to the graph signal. Using $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$, the QV of the eigenvector $\mathbf{v}_i$ can be expressed as

$$\mathrm{QV}(\mathbf{v}_i) = \mathbf{v}_i^T\mathbf{L}\mathbf{v}_i = \lambda_i.$$

Here, the eigenvalue λ_i represents one of the frequency components of the graph signal $\mathbf{x}$. The GFT $\mathbf{x}_i^{\mathrm{GFT}}$ indicates how much of this frequency component associated with λ_i contributes to $\mathbf{x}$. Similar to how the DFT decomposes a sinusoidal signal into amplitude and frequency, the GFT represents the amplitude of the graph signal as $\mathbf{x}_i^{\mathrm{GFT}}$ and its frequency with eigenvalues λ_i. Based on the spectral response, filters analogous to low-pass and high-pass filters in the Fourier domain can be designed for graph signals. Alternatively, these filters can be learnt using data-driven approaches, which is of primary interest in this context.

B-0-1 Graph Filters and their properties

Most of the graph filters rely on the shift-and-sum operation of the input signal: convolution principle (see Figure B-2). These filters are referred to as the graph convolutional filter (GCF), which are discussed next.

Graph Convolutional Filters

Definition 7. *(Graph Convolution Filter) A graph convolutional filter* $H_c : \mathbb{X}^{\mathcal{V}} \to \mathbb{X}^{\mathcal{V}}$ *is a linear transformation with shifted signals of order* K

$$\mathbf{y} = H_c(\mathbf{x}) = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x}, \tag{B-3}$$

with $H(\mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ *as a polynomial filtering matrix.*

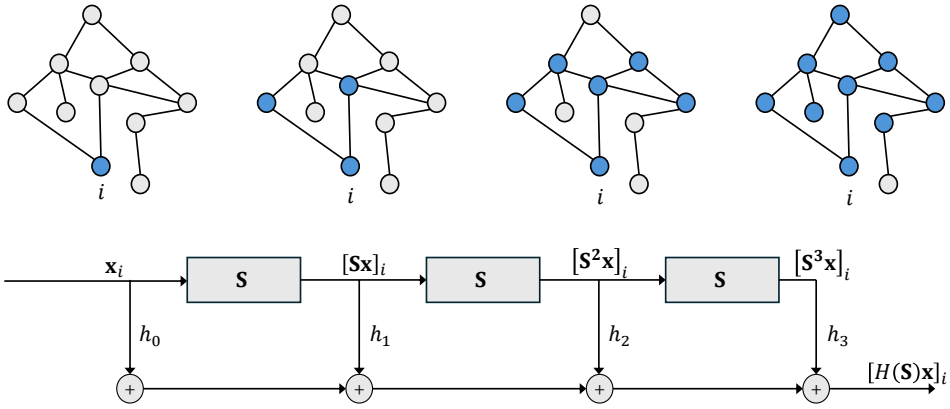


Figure B-2: Shift-and-sum operations in graph convolutional filter [5] using GSO as $\mathbf{A}$

The frequency response of the filter output $\mathbf{y}$ can be given by substituting $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$ in (B-3), resulting in the filter frequency response relation as

$$\tilde{\mathbf{y}} = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k \tilde{\mathbf{x}}. \tag{B-4}$$

Note that (B-4) affirms the convolution theorem, stating that the shift-and-sum operation in the nominal (vertex) domain acts as a pointwise multiplication between the filter frequency response $\mathrm{h}(\lambda) = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k$ and $\tilde{\mathbf{x}}$. Therefore, for the ith component of the filter output,

$$\tilde{y_i} = h(\lambda_i)\tilde{x}_i. \tag{B-5}$$

Another insight here is that $h(\lambda_i) = h_0 + h_1\lambda_i + h_2\lambda_i^2 + \cdots + h_K\lambda_i^K$ represents a polynomial function of the eigenvalue λ_i, where the coefficients $h_0, h_1, h_2, \ldots, h_K$ define the polynomial. This relationship holds for all eigenvalues (graph frequencies) λ_i of the GSO. Therefore, designing a graph filter in the spectral domain corresponds to learning the coefficients of the polynomial $h(\lambda_i)$, which effectively defines the filter response over the range of frequencies $[\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum eigenvalues of the GSO.

For the case when multiple features are attributed to the graph signal, i.e., $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_F] \in \mathbb{R}^{|\mathcal{V}|\times F}$, the GCF architecture can be modified to output multiple filters (known as filter banks), given as

$$\mathbf{Y} = \sum_{k=0}^{K} \mathbf{S}^k \mathbf{X} \mathbf{H}_k, \tag{B-6}$$

where $\mathbf{H}_k \in \mathbb{R}^{F \times F}$ and filter output $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_F] \in \mathbb{R}^{|\mathcal{V}| \times F}$.

The wide adoption of GCFs in the Graph ML literature can be attributed to properties like linearity, shift-invariance, permutation equivariance, parameter sharing, linear cost of computation and Lipschitz continuity for changes in the $\mathbf{S}$. These properties imply transferability and scalability of the trained filter coefficients over other graphs bounded by some measure [73]. Despite their advantages, GCFs struggle to balance descriptive power and stability, where stability refers to a filter's robustness to small graph perturbations [73]. Linear filters cannot simultaneously ensure stability and strong high-frequency discrimination. This trade-off can be alleviated by introducing pointwise nonlinearities and using filter banks that capture information across multiple frequency bands. Together, these elements motivate the design of graph convolutional neural network (GCNN) architectures. Now, similar to graphs, simplicial complexes (SC) representation has simplicial convolutional filters (SCFs) as discussed next.

B-0-2 Simplicial Convolutional Filters

Using SC representation in Section 2-3-2 similar to the shift-and-sum operation used in GCFs, convolutional filtering on SCs leverages the Hodge-Laplacian as a shift operator. Such filters are known as SCFs. Following the notations in Section 2-3-2, a $k-$simplicial signal can be given as a mapping, $\mathbf{X}^k : \mathcal{S}^k \to \mathbb{R}^{|\mathcal{S}^k| \times G_k}$ with G_k is the number of features on the $k-$simplicial signal. This way, the graph signal is a $0-$simplicial signal, $\mathbf{X} = \mathbf{X}^0$, the $1-$ simplicial signal on the edges can be given as $\mathbf{X}^1$ with G (for convenience rather than G_1) as the cardinality of edge-feature space. Leveraging the concept of convolution filters and Hodge Laplacian from (2-35), a simplicial convolutional filter (SCF) for $1-$simplicial signal on edges can be defined as

$$\mathbf{Y}^1 = H_s(\mathbf{X}^1) = \left(\sum_{q_1=0}^{Q_1} (\mathbf{L}_{1,l})^{q_1} \mathbf{X}^1 \mathbf{H}_{q_1}^l + \sum_{q_2=0}^{Q_2} (\mathbf{L}_{1,u})^{q_2} \mathbf{X}^1 \mathbf{H}_{q_2}^u \right), \tag{B-7}$$

where Q_1 and Q_2 are the filter orders for lower Laplacian $\mathbf{L}_{1,l} = \mathbf{B}_1^T \mathbf{B}_1$ and upper Laplacian $\mathbf{L}_{1,u} = \mathbf{B}_2 \mathbf{B}_2^T$ respectively, and $\mathbf{H}^l, \mathbf{H}^u \in \mathbb{R}^{G \times G}$, [49]. The complexity of the filter is given by $\mathcal{O}(|\mathcal{E}|(Q_1 + Q_2)DG^2)$ where D is the maximal number of edge-neighbours. This complexity is linear in the number of edges.

Other filters like rational filters, node-varying and edge-varying filters enhance flexibility and offer increased modelling power, but face limitations such as high computational cost, numerical instability and poor generalizability. An extension to graph filters are graph regularizers that incorporate prior knowledge like smoothness to refine noisy or incomplete signals. They serve key functions like de-noising, imputation and node-classification, offering promising tools to address network observability challenges.

Bibliography

[1] TenneT BV. Grid capacity map. https://www.tennet.eu/grid-capacity-map. Accessed: 2024-11-21.

[2] Ben Gulick and Mac Spiller. Transformer Tap Changers & Voltage Adjustment Taps, June 2023. Accessed: 2025-05-27.

[3] Erich Friedman. Squares Covering Triangles. https://erich-friedman.github.io/packing/squcotri/, 2009. Accessed: 2025-05-28.

[4] Kursat Rasim Mestav, Jaime Luengo-Rozas, and Lang Tong. Bayesian State Estimation for Unobservable Distribution Systems via Deep Learning. *IEEE Transactions on Power Systems*, 34(6):4910–4920, 2019.

[5] Elvin Isufi, Fernando Gama, David I Shuman, and Santiago Segarra. Graph Filters for Signal Processing and Machine Learning on Graphs. *IEEE Transactions on Signal Processing*, 72:4745–4781, 2024.

[6] Phase To Phase B.V. *Netten voor Distributie van Elektricitiet, 5th Edition*. Phase To Phase B.V., 2020.

[7] Hannah Ritchie, Max Roser, and Pablo Rosado. Renewable energy. https://ourworldindata.org/renewable-energy, 2020. Accessed: 2025-05-23.

[8] Fred C Schweppe and J Wildes. Power System Static-State Estimation, Part 1: Exact Model. *IEEE Transactions on Power Apparatus and systems*, (1):120–125, 1970.

[9] Ali Abur and Antonio Gomez Exposito. *Power System State Estimation: Theory and Implementation*. CRC press, 2004.

[10] Frederik Geth, Marta Vanin, and Dirk Van Hertem. Data Quality Challenges in Existing Distribution Network Datasets. *CoRR*, abs/2308.00487, 2023.

[11] Wenyuan Li, Ebrahim Vaahedi, and Paul Choudhury. Power System Equipment Aging. *IEEE Power and Energy Magazine*, 4(3):52–58, 2006.

[12] Tim Krause, Raphael Ernst, Benedikt Klaer, Immanuel Hacker, and Martin Henze. Cybersecurity in power grids: Challenges and opportunities. *Sensors*, 21(18), 2021.

[13] Karl Popper. *The Logic of Scientific Discovery*. Routledge, 2005.

[14] Frederik Geth, Marta Vanin, Werner Van Westering, Terese Milford, and Amritanshu Pandey. Making Distribution State Estimation Practical: Challenges and Opportunities. *https://arxiv.org/abs/2311.07021*, 2023.

[15] Anuradha Tomar, Muhammad Babar, and Phuong H Nguyen. Data-driven Congestion Management for LV Distribution Networks. *CIRED*, 2020:166–169, 2021.

[16] Soham Prajapati. A GAN-GNN framework for Topology-Aware State and Transformer Tap Position Estimation in Unobservable Distribution Grids. https://github.com/prajapati-incontrol/TapSEGNN, 2025. GitHub repository.

[17] James L Kirtley. *Electric Power Principles: Sources, Conversion, Distribution and Use*. John Wiley & Sons Ltd, 2020.

[18] Dwarkadas Kothari, IJ Nagrath, and RK Saket. *Modern Power System Analysis*. McGraw Hill Education Private Limited, 2022.

[19] Jody Verboomen, Dirk Van Hertem, Pieter H Schavemaker, Wil L Kling, and Ronnie Belmans. Phase Shifting Transformers: Principles and Applications. In *2005 International Conference on Future Power Systems*, pages 6–pp. IEEE, 2005.

[20] Pedro Zarco and Antonio Gomez Exposito. Power System Parameter Estimation: A Survey. *IEEE Transactions on Power Systems*, 15(1):216–222, 2000.

[21] Jubair Yusuf, Joseph A Azzolini, and Matthew J Reno. Data-Driven Methods for Voltage Regulator Identification and Tap Estimation. In *2022 IEEE Kansas Power and Energy Conference (KPEC)*, pages 1–6. IEEE, 2022.

[22] Milton Brown Do Coutto Filho. *Power System State Estimation and Forecasting*. Springer Cham, 2024.

[23] RAM van Amerongen. On Convergence Analysis and Convergence Enhancement of Power System Least-Squares State Estimators. *IEEE Transactions on Power Systems*, 10(4):2038–2044, 1995.

[24] Slobodan Pajic and Kevin A Clements. Power System State Estimation via Globally Convergent Methods. *IEEE Transactions on Power Systems*, 20(4):1683–1689, 2005.

[25] John E Dennis and Robert B Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996.

[26] Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum Width for Universal Approximation. *arXiv preprint arXiv:2006.08859*, 2020.

[27] Andrea Bernieri, Giovanni Betta, Consolatina Liguori, and Arturo Losi. Neural Networks and Pseudo-Measurements for real-Time Monitoring of Distribution Systems. *IEEE Transactions on Instrumentation and Measurement*, 45(2):645–650, 1996.

[28] Bin Huang and Jianhui Wang. Applications of Physics-Informed Neural Networks in Power Systems-A Review. *IEEE Transactions on Power Systems*, 38(1):572–588, 2022.

[29] Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing Hard Constraints on Deep Networks: Promises and Limitations. *arXiv preprint arXiv:1706.02025*, 2017.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32, 2019.

[31] Jonatan Ostrometzky, Konstantin Berestizshevsky, Andrey Bernstein, and Gil Zussman. Physics-Informed Deep Neural Network Method for Limited Observability State Estimation. In *Workshop on Autonomous Energy Systems*, pages 1–6. NREL Golden CO, 2020.

[32] Wenlong Liao, Birgitte Bak-Jensen, Jayakrishnan Radhakrishna Pillai, Yuelong Wang, and Yusen Wang. A Review of Graph Neural Networks and Their Applications in Power Systems. *Journal of Modern Power Systems and Clean Energy*, 10(2):345–360, 2022.

[33] Qiuling Yang, Alireza Sadeghi, Gang Wang, Georgios B Giannakis, and Jian Sun. Power System State Estimation using Gauss-Newton Unrolled Neural Networks with Trainable Priors. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6. IEEE, 2020.

[34] Hui Lin and Yan Sun. EleGNN: Electrical-Model-Guided Graph Neural Networks for Power Distribution System State Estimation. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 5292–5298. IEEE, 2022.

[35] Nan Lin, Stavros Orfanoudakis, Nathan Ordonez Cardenas, Juan S Giraldo, and Pedro P Vergara. PowerFlowNet: Power Flow Approximation using Message Passing Graph Neural Networks. *International Journal of Electrical Power & Energy Systems*, 160:110112, 2024.

[36] Shiva Moshtagh, Anwarul Islam Sifat, Behrouz Azimian, and Anamitra Pal. Time-Synchronized State Estimation Using Graph Neural Networks in Presence of Topology Changes. In *2023 North American Power Symposium (NAPS)*, pages 1–6. IEEE, 2023.

[37] Laurent Pagnier and Michael Chertkov. Physics-Informed Graphical Neural Network for Parameter and State Estimations in Power Systems. *arXiv preprint arXiv:2102.06349*, 2021.

[38] Zhiwei Wang, Min Xia, Min Lu, Lingling Pan, and Jun Liu. Parameter Identification in Power Transmission Systems based on Graph Convolution Network. *IEEE Transactions on Power Delivery*, 37(4):3155–3163, 2021.

[39] Rahul Madbhavi, Balasubramaniam Natarajan, and Babji Srinivasan. Graph Neural Network-Based Distribution System State Estimators. *IEEE Transactions on Industrial Informatics*, 19(12):11630–11639, 2023.

[40] Quang-Ha Ngo, Bang LH Nguyen, Tuyen V Vu, and Tuan Ngo. State Estimation for Power Distribution System Using Graph Neural Networks. In *2023 IEEE Electric Ship Technologies Symposium (ESTS)*, pages 441–446. IEEE, 2023.

[41] Martin Ringsquandl, Houssem Sellami, Marcel Hildebrandt, Dagmar Beyer, Sylwia Henselmeyer, Sebastian Weber, and Mitchell Joblin. Power to the Relational Inductive Bias: Graph Neural Networks in Electrical Power Grids. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1538–1547, 2021.

[42] Benjamin Habib, Elvin Isufi, Ward van Breda, Arjen Jongepier, and Jochen L Cremer. Deep Statistical Solver for Distribution System State Estimation. *IEEE Transactions on Power Systems*, 39(2):4039–4050, 2023.

[43] Lek-Heng Lim. Hodge Laplacians on Graphs. *Siam Review*, 62(3):685–715, 2020.

[44] Sergio Barbarossa and Stefania Sardellitti. Topological Signal Processing over Simplicial Complexes. *IEEE Transactions on Signal Processing*, 68:2992–3007, 2020.

[45] Maosheng Yang, Elvin Isufi, and Geert Leus. Simplicial Convolutional Neural Networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8847–8851. IEEE, 2022.

[46] Elson Cibaku, Fernando Gama, and SangWoo Park. Boosting Efficiency in State Estimation of Power Systems by Leveraging Attention Mechanism. *Energy and AI*, 16:100369, 2024.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, 2017.

[48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. *arXiv preprint arXiv:1710.10903*, 2017.

[49] Elvin Isufi and Maosheng Yang. Convolutional Filtering in Simplicial Complexes. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5578–5582. IEEE, 2022.

[50] Amlan Ghosh, David Lubkeman, and RH Jones. Load Modeling for Distribution Circuit State Estimation. *IEEE Transactions on Power Delivery*, 12(2):999–1005, 1997.

[51] Sai Suprabhath Nibhanupudi, Anton Ishchenko, Simon H Tindemans, and Peter Palensky. State Estimation in Medium Voltage Distribution Networks using Pseudo Measurements. In *CIGRE Session 2022*, 2022.

[52] Priya L Donti, Yajing Liu, Andreas J Schmitt, Andrey Bernstein, Rui Yang, and Yingchen Zhang. Matrix Completion for Low-Observability Voltage Estimation. *IEEE Transactions on Smart Grid*, 11(3):2520–2530, 2020.

[53] Xiaoxiao Shi and Philip S Yu. Limitations of Matrix Completion via Trace Norm Minimization. *ACM SIGKDD Explorations Newsletter*, 12(2):16–20, 2011.

[54] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep Learning*, volume 1. MIT press Cambridge, 2016.

[55] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly Accurate Protein Structure Prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[56] Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery. Technical report, Technical report, Google DeepMind, 05 2025, 2025.

[57] Shun-ichi Amari. Backpropagation and Stochastic Gradient Descent Method. *Neurocomputing*, 5(4-5):185–196, 1993.

[58] Mohasinina Kamal, Wenting Li, Deepjyoti Deka, and Hamed Mohsenian-Rad. Physics-Conditioned Generative Adversarial Networks for State Estimation in Active Power Distribution Systems with Low Observability. In *2022 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*, pages 1–6. IEEE, 2022.

[59] Y Raghuvamsi and Kiran Teeparthi. Distribution System State Estimation with Convolutional Generative Adversarial Imputation Networks for Missing Measurement Data. *Arabian Journal for Science and Engineering*, 49(5):6641–6656, 2024.

[60] Jinsung Yoon, James Jordon, and Mihaela Schaar. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *International Conference on Machine Learning*, pages 5689–5698. PMLR, 2018.

[61] Lars Ruthotto and Eldad Haber. An Introduction to Deep Generative Modelling. *GAMM-Mitteilungen*, 44(2):e202100008, 2021.

[62] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. *Advances in Neural Information Processing Systems*, 31, 2018.

[63] Zhilu Zhang and Mert R Sabuncu. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8792–8802, 2018.

[64] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent Advances in Deep Learning for Speech Research at Microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8604–8608. IEEE, 2013.

[65] Diederik P Kingma. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] Leon Thurner, Alexander Scheidler, Florian Schafer, Jan-Hendrik Menke, Julien Dollichon, Florian Meier, Simon Meinecke, and Martin 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.

[67] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv preprint arXiv:1701.00160*, 2016.

[68] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *Communications of the ACM*, 63(11):139–144, 2020.

[69] Ferenc Huszár. How (Not) to Train Your Generative Model: Scheduled Sampling, Likelihood, Adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[70] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which Training Methods for GANs do Actually Converge? In *International Conference on Machine Learning*, pages 3481–3490. PMLR, 2018.

[71] Bent Fuglede and Flemming Topsoe. Jensen-Shannon Divergence and Hilbert Space Embedding. In *International Symposium on Information Theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE, 2004.

[72] James M Joyce. Kullback-Leibler Divergence. In *International Encyclopedia of Statistical Science*, pages 720–722. Springer, 2011.

[73] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability Properties of Graph Neural Networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

[74] Yujia Li, Kevin Swersky, and Rich Zemel. Generative Moment Matching Networks. In *International conference on machine learning*, pages 1718–1727. PMLR, 2015.

[75] Arka Daw, Maruf Maruf, and Anuj Karpatne. PID-GAN: A GAN Framework based on a Physics-informed Discriminator for Uncertainty Quantification with Physics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 237–247, 2021.

[76] Siemens Energy. PSS – Power System Simulation Software. https://www.siemens.com/global/en/products/energy/grid-software/planning/pss-software.html, 2025. Accessed: 2025-07-01.

[77] DIgSILENT GmbH. PowerFactory: Power System Analysis Software, 2025. Accessed: 2025-07-01.

[78] Nicola De Cao and Thomas Kipf. MolGAN: An Implicit Generative Model for Small Molecular Graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[79] Hendrik F Hamann, Blazhe Gjorgiev, Thomas Brunschwiler, Leonardo SA Martins, Alban Puech, Anna Varbella, Jonas Weiss, Juan Bernabe-Moreno, Alexandre Blondin Massé, Seong Lok Choi, et al. Foundation Models for the Electric Power Grid. *Joule*, 8(12):3245–3258, 2024.

[80] Courtney Remani. Numerical Methods for Solving Systems of Nonlinear Equations. *Lakehead University Thunder Bay, Ontario, Canada*, 77, 2013.

[81] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph Signal Processing: Overview, Challenges, and Applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

Glossary

List of Acronyms

DG	distribution grid
TG	transmission grid
SE	state estimation
GNN	graph neural network
SCNN	simplicial complex neural network
SPE	state and parameter estimation
PE	parameter estimation
GAN	generative adversarial network
RES	renewable energy sources
PF	power flow
EMS	Energy Management System
HV	high voltage
MV	medium voltage
LV	low voltage
GSP	graph signal processing
SCADA	Supervisory Control and Data Acquisition
PMU	Phasor Measurement Unit
RMS	root mean square
ML	machine learning
WLS	weighted least squares
DSOs	distribution system operators
DGSPE	distribution grid state and parameter estimation
PFA	power flow analysis
MLE	maximum likelihood estimation

GN	Gauss-Newton
GS	Gauss-Seidel
NR	Newton-Raphson
NN	neural network
PANNs	physics-agnostic neural networks
PINNs	physics-informed neural networks
FCNNs	fully connected neural networks
CNNs	convolutional neural networks
DGSE	distribution grid state estimation
GSO	graph shift operator
SC	simplicial complexes
GCNN	graph convolutional neural network
GAT	graph attention neural network
NLP	natural language processing
CNN	convolutional neural network
FCNN	fully connected neural network
GIS	Geographic Information Systems
RMSE	root mean square error
FIR	finite impulse response
DFT	discrete Fourier transform
GFT	graph Fourier transform
GCF	graph convolutional filter
SCF	simplicial convolutional filter
SCFs	simplicial convolutional filters

List of Symbols

δ	Transformer Phase-Shifting Angle
λ	Eigen-vector of graph shift operator
λ_{PINN}	Hyperparameter for physics-informed loss function
λ_{reg}	Regularisation weight on loss function
λ_{tap}	Weight for tap-loss in SPE loss function
$\mathbf{\Omega}$	Parameter matrix for transformer readout layer
ω	Angular Frequency
Φ	Parameter set of the generator model
$\rho(\mathbf{D}_v^{\text{elegnn}})$	Spectral radii of a matrix
σ	Standard deviation

τ	Transformer Tap-Ratio
θ_i	Current Phase-Shift Angle
θ_v	Voltage Phase-Shift Angle
Υ	Parameter set of the discriminator model
$\boldsymbol{a}^{ij}$	Parameter vector to linearly combine node features in transformer readout layer
$\boldsymbol{g}$	Nonlinear Vector-Valued Function
$\boldsymbol{u}$	Sample of synthetic power flow results
$\boldsymbol{v}$	Sample from latent space
χ	Distribution of synthetic power flow results
$\hat{\mathbf{Y}}$	Output node embeddings of graph convolutional neural network
$\mathbf{\Lambda}$	Diagonal matrix containing eigen-values
$\mathbf{A}$	Undirected graph adjacency matrix
$\mathbf{a}_e$	Edge parameter vector in graph attention
$\mathbf{a}_s$	Source node parameter vector in graph attention network
$\mathbf{a}_t$	Target node parameter vector in graph attention network
$\mathbf{A}_\alpha$	Attention-induced graph shift operator
$\mathbf{B}_1$	Node-edge incidence matrix
$\mathbf{B}_2$	Edge-triangle incidence matrix
$\mathbf{D}$	Graph Diagonal Matrix
$\mathbf{D}^{\text{elegnn}}$	Diffusion matrix for Ele-GNN
$\mathbf{e}$	Measurement-error vector
$\mathbf{H}_{l,k}$	Graph convolutional neural network parameter matrix at lth layer and for kth order
$\mathbf{I}_{\text{bus}}$	Complex Current Vector for all Buses
$\mathbf{L}$	Graph Laplacian matrix
$\mathbf{M}^{ij}$	Binary mask matrix for edge *(i,j)*
$\mathbf{R}$	Covariance matrix for state measurements
$\mathbf{R}_p$	Covariance matrix for parameter measurements
$\mathbf{S}$	Undirected graph shift operator
$\mathbf{t}^{ij}$	Output logit transformer readout layer
$\mathbf{u}$	State Vector
$\mathbf{V}$	Matrix of eigen-vectors of graph shift operator
$\mathbf{V}_{\text{bus}}$	Complex Voltage Vector for all Buses
$\mathbf{W}$	Weight matrix on individual node features
$\mathbf{W}_e$	Weight matrix on individual edge features
$\mathbf{X}$	Node feature matrix
$\mathbf{x}$	Node feature vector
$\mathbf{x}'$	Output Graph Signal Embedding from the Graph Perceptron
$\mathbf{X}^1$	Edge feature matrix
$\mathbf{x}^1$	Edge feature vector
$\mathbf{Y}_{\text{bus}}$	Bus-Admittance Matrix

$\mathbf{Z}$	Node-label matrix
$\mathbf{z}$	Measurement Vector
$\mathcal{B}$	Set of Buses in the Network
$\mathcal{Z}$	Distribution of Latent Space
$\tilde{\mathbf{A}}$	Normalized graph adjacency
$\tilde{\mathbf{L}}$	Normalized graph Laplacian
$\underline{N}$	Complex Tap-Ratio
$\underline{S}$	Complex Power Injection
$\underline{S}_{ij}$	Complex Power Flow
$\underline{y}_s$	Series Admittance
$\underline{y}_s$	Series Admittance
$\underline{y}_{sh}$	Shunt-Admittance
C_k	Space of k chains
d	Number of Branches
F	Node feature dimension
G	Edge feature dimension
G_Φ	Generative Model
h	Hop for neighborhood
L^{SE}	Loss function for state estimation
L^{Tap}	Loss function for tap position estimation
M	Number of heads in multi-head graph attention network
N	Number of training samples
n	Number of Buses
P	Active Power Injection
P_{ij}	Active Power Flow
Q	Reactive Power Injection
Q_{ij}	Reactive Power Flow
t	Total tap positions in transformers
t	Total transformer tap positions
$\mathbf{L}_0$	Graph Laplacian
$\mathbf{L}_1$	Hodge-Laplacian
$\mathbf{L}_{1,l}$	Lower Hodge-Laplacian
$\mathbf{L}_{1,u}$	Upper Hodge-Laplacian
$\mathbf{X}$	Graph Signal or Node Feature Matrix for Multiple Features
$\mathbf{X}_l$	Output Embedding Node Feature Matrix
*	Conjugate Transpose
$\boldsymbol{g}$	Nonlinear vector valued function
$\mathbb{E}$	Expectation operator
$\mathbf{G}$	Gain Matrix
$\mathbf{J}$	Jacobian Matrix

$\mathbf{q}$	State-vector for Gauss-Seidel method
$\mathbf{r}$	Residual vector in state estimation
$\mathcal{A}$	Hessian matrix in Gauss-Newton method
$\mathcal{D}^{\text{GAN}}$	Dataset GAN
$\mathcal{D}^{\text{SE}}$	Dataset SE
$\mathcal{D}^{\text{TapSE}}$	Dataset TapSE
$\mathcal{E}$	Set of edges in the graph
$\mathcal{E}_d$	Set of directed edges in the graph
$\mathcal{G}$	Undirected unweighted graph
$\mathcal{G}_d$	Directed graph
$\mathcal{H}^{\text{GCNN}}$	Graph convolutional neural network parameter set
$\mathcal{H}^{\text{SCNN}}$	Parameter set for simplicial convolutional neural network
$\mathcal{N}(i)$	Neighbourhood set of node i
$\mathcal{N}_t$	Set of edges modelled as transformers
$\mathcal{R}$	Regularization term
$\mathcal{S}^k$	Subset of node-set consisting of k+1 unique elements
$\mathcal{U}$	Set of Branches
$\mathcal{V}$	Set of nodes in the graph
$\mathcal{W}$	Parameter Set of the Readout Layer in GNNs
$\mathcal{X}^K$	Simplicial Complex of order K
Ψ	Parameter set of TapSEGNN model
$\sigma(x)$	Pointwise nonlinear activation function on x
$\underline{a}$	Generalized Circuit Constant
$\underline{i}$	Per-unit Complex Current
$\underline{v}$	Per-unit Complex Voltage
$\underline{Z}$	Complex Impedance
$\underline{z}$	Per-unit Complex Impedance
b_{ik}	Magnetising Susceptance (Line Charging Susceptance) of Line connecting Bus i and k
g_{ik}	Magnetising Conductance of Line connecting Bus i and k
L^{PINN}	Physics-Informed Loss Function
p	Dimension of space of synthetic power flow results
q	Dimension of latent space
W	Nonlinear weighted least squares objective function