



**POLITECNICO**  
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE  
E DELL'INFORMAZIONE

# Investigating Network Reduction with Ward's Equivalent: Balancing Accuracy and Computational Speed for Large-scale Power System Anal- ysis

TESI DI LAUREA MAGISTRALE IN  
MSc. IN ELECTRICAL ENGINEERING (R2E TRACK) - INGEG-  
NERIA ELETTRICA

Author: **Partha Pratim Konwar**

Student ID: 10898675

Advisor: Prof. Alberto Berizzi

Co-advisors: Prof. Valentin Ilea

Academic Year: 2022-24



# Abstract

This thesis explores the application of **Ward's Equivalent (Ward's Injection)** in network reduction to balance computational efficiency and accuracy for large-scale power systems. A detailed description of the algorithm for both network reduction and N-1 contingency simulation is provided, showcasing their implementation in grid analysis. By simplifying external nodes while preserving essential internal characteristics, Ward's Equivalent proves its value, particularly in steady-state and N-1 contingency analysis. The study demonstrates that the method can significantly streamline complex grid analyses, as evidenced through its application on **Rueda Test Grid**. Notably, the reduced models retained critical system behaviors such as voltage, current, and reactive power flow within acceptable error margins, even under contingency conditions. Additionally, boundary node selection was highlighted as a crucial factor affecting accuracy, and adjusting these nodes mitigated voltage, branch currents and Reactive Power discrepancies. Overall, the thesis reaffirms Ward's Equivalent as a powerful tool for improving computational efficiency while maintaining accuracy in power system analysis.

**Keywords:** Ward's Equivalent, Network Reduction, Power System Analysis, N-1 Contingency, Computational Efficiency, Boundary Nodes selection strategy.



## Abstract in lingua italiana

Questa tesi esplora l'applicazione dell'**equivalente di Ward (iniezione di Ward)** nella riduzione di rete per bilanciare efficienza computazionale e accuratezza per sistemi di potenza su larga scala. Viene fornita una descrizione dettagliata dell'algoritmo sia per la riduzione di rete che per la simulazione di contingenza N-1, mostrando la loro implementazione nell'analisi di rete. Semplificando i nodi esterni pur preservando le caratteristiche interne essenziali, l'equivalente di Ward dimostra il suo valore, in particolare nell'analisi di stato stazionario e di contingenza N-1. Lo studio dimostra che il metodo può semplificare in modo significativo le analisi di rete complesse, come dimostrato dalla sua applicazione su **Rueda Test Grid**. In particolare, i modelli ridotti hanno mantenuto comportamenti di sistema critici come tensione, corrente e flusso di potenza reattiva entro margini di errore accettabili, anche in condizioni di contingenza. Inoltre, la selezione del nodo di confine è stata evidenziata come un fattore cruciale che influenza la precisione e la regolazione di questi nodi ha attenuato le discrepanze di tensione, correnti di diramazione e potenza reattiva. Nel complesso, la tesi riafferma l'equivalente di Ward come un potente strumento per migliorare l'efficienza computazionale mantenendo l'accuratezza nell'analisi del sistema di potenza.

**Parole chiave:** Equivalente di Ward, Riduzione di rete, Analisi del sistema di alimentazione, Contingenza N-1, Efficienza computazionale, Strategia di selezione dei nodi di confine.



# Contents

<b>Abstract</b>	<b>i</b>
<b>Abstract in lingua italiana</b>	<b>iii</b>
<b>Contents</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Growing Complexity and the Need for Advanced Tools . . . . .	2
1.2 The Role of Network Reduction in Power Systems . . . . .	2
1.2.1 Streamlining Real-Time Analysis . . . . .	3
1.2.2 Enhancing Computational Efficiency . . . . .	3
1.2.3 Focusing on Critical Areas . . . . .	4
1.2.4 Challenges and Limitations . . . . .	4
1.3 Network Reduction Techniques in Power Systems . . . . .	5
1.3.1 Ward's Equivalent: A Core Network Reduction Technique . . . . .	5
1.3.2 Applications in Large-Scale Power Networks . . . . .	6
1.3.3 Other Network Reduction Techniques . . . . .	6
1.3.4 Recent Developments in Network Reduction . . . . .	7
1.4 Network Reduction Techniques in Distribution Networks . . . . .	7
1.5 N-1 Contingency Analysis and the Role of Ward's Equivalent . . . . .	8
1.6 Evaluating the Effectiveness and Limitations of Network Reduction Tech- niques . . . . .	8
1.6.1 Effectiveness . . . . .	8
1.6.2 Limitations . . . . .	8
1.7 Rationale for Selecting Ward's Equivalent . . . . .	9
1.8 Research Gaps and Future Directions . . . . .	9
<b>2 Network Reduction Techniques for Steady-State Analysis</b>	<b>11</b>
2.1 Ward's Equivalent . . . . .	11

2.1.1	Principles of Ward's Equivalent . . . . .	11
2.1.2	Mathematical Formulation . . . . .	12
2.1.3	Applications in Power Systems . . . . .	12
2.1.4	Variants of Ward's Equivalent . . . . .	13
2.1.5	Advantages and Limitations of Ward's Equivalent . . . . .	15
2.2	Radial Equivalent Independent (REI) model . . . . .	16
2.2.1	Principles of REI Equivalent . . . . .	16
2.2.2	Applications in Power Systems . . . . .	16
2.2.3	Advantages and Limitations . . . . .	17
2.3	Kron Reduction . . . . .	17
2.3.1	Principles of Kron Reduction . . . . .	17
2.3.2	Mathematical Formulation . . . . .	17
2.3.3	Applications in Power Systems . . . . .	18
2.3.4	Advantages and Limitations . . . . .	18
<b>3</b>	<b>Algorithm for Network Reduction and Analysis</b>	<b>21</b>
3.1	Power Flow Analysis (Stage 1) . . . . .	21
3.1.1	Data Loading and Initialization . . . . .	21
3.1.2	Generator Control Adjustments . . . . .	22
3.1.3	Load Flow Computation . . . . .	22
3.1.4	Complex Power Injection Calculation . . . . .	23
3.1.5	Executing the Load Flow . . . . .	23
3.1.6	Step-by-Step Execution . . . . .	23
3.1.7	Finalizing the Load Flow Solution . . . . .	25
3.2	Identifying Internal, Boundary, and External Nodes (Stage 2) . . . . .	25
3.2.1	Graph Representation of the Network . . . . .	25
3.2.2	Classification of Nodes: Internal, Boundary, and External . . . . .	25
3.2.3	Flowchart Description: Identifying Internal Nodes . . . . .	26
3.2.4	Visualization of Node Classification . . . . .	28
3.3	Impact of Node Classification on Network Reduction . . . . .	28
3.4	Admittance Matrix Partitioning and Ward's Reduction (Stage 3) . . . . .	29
3.4.1	Matrix Partitioning . . . . .	30
3.4.2	Ward's Reduction . . . . .	30
3.4.3	Partitioning the Reduced Admittance Matrix . . . . .	31
3.4.4	Flowchart: Ward's Reduction Process . . . . .	32
3.4.5	Why Ward's Reduction Matters . . . . .	33
3.5	Updating the Branch Matrix and Electrical Parameters (Stage 4) . . . . .	34

- 3.5.1 Filtering and Retaining Relevant Branches . . . . . 34
- 3.5.2 Recalculating Boundary Connections . . . . . 34
- 3.5.3 Flowchart: Updating the Branch Matrix . . . . . 34
- 3.5.4 Updating the Branch Matrix with New Parameters . . . . . 36
- 3.5.5 Flowchart: Updating Resistance and Reactance (R and X) . . . . . 36
- 3.6 Recalculating Power Injections, Voltages, and Shunt Admittances (Stage 5) 38
  - 3.6.1 Recomputing Real and Reactive Power Demand . . . . . 38
  - 3.6.2 Updating Power Demand for Internal and Boundary Nodes . . . . . 39
  - 3.6.3 Flowchart: Updating Power Demands . . . . . 39
  - 3.6.4 Recalculating and Updating Bus Voltage Profiles . . . . . 40
  - 3.6.5 Calculating and Updating Shunt Admittances . . . . . 40
- 3.7 Updating Generator Information (Stage 6) . . . . . 41
  - 3.7.1 Filtering Generators in the Reduced Network . . . . . 41
  - 3.7.2 Recalculating Real and Reactive Power Outputs . . . . . 42
  - 3.7.3 Flowchart: Updating Generator Matrix . . . . . 42
  - 3.7.4 Assigning a Slack Bus . . . . . 43
  - 3.7.5 Flowchart: Checking for Slack Bus . . . . . 43
- 3.8 Finalizing the Reduced Network Data (Stage 7) . . . . . 45
  - 3.8.1 Filtering Phase Shifters and Network Constraints . . . . . 46
  - 3.8.2 Filtering and Updating Generator Bid Data . . . . . 46
  - 3.8.3 Updating Node and Branch Information . . . . . 46
  - 3.8.4 Ensuring Consistency in Labels and Other Operational Data . . . . . 46
  - 3.8.5 Flowchart: Finalizing the Reduced Network . . . . . 47
  - 3.8.6 Ensuring Complete Finalization of the Reduced Network . . . . . 47
- 3.9 Renumbering and Saving the Reduced Network (Stage 8) . . . . . 49
  - 3.9.1 Renumbering Buses for Consistency and Simplification . . . . . 49
  - 3.9.2 Renumbering Branches for Compatibility with New Bus Numbers . . . . . 49
  - 3.9.3 Renumbering Generators and Branchset . . . . . 50
  - 3.9.4 Saving the Final Reduced Network . . . . . 50
  - 3.9.5 Flowchart: Renumbering and Mapping the Reduced Network . . . . . 50
  - 3.9.6 Ensuring Consistency and Usability of the Reduced Network . . . . . 51
- 3.10 Investigating the Efficacy of the Reduction and Future Applications . . . . . 51
  - 3.10.1 Investigating the Efficacy of the Reduction . . . . . 52
  - 3.10.2 Future Applications of the Reduced Model . . . . . 53

**4 Observation on the Reduced Network and Ward’s Reduction Accuracy Analysis** **55**

4.1	A Comparative Assessment of Power Flow Results between Full and Reduced Networks under Normal Conditions. . . . .	55
4.1.1	Understanding Boundary and Internal Buses . . . . .	56
4.1.2	Voltage Magnitude and Phase Angle Differences . . . . .	56
4.1.3	Bus Current Differences . . . . .	56
4.1.4	Interpretation of Results . . . . .	57
4.2	Contingency Analysis of the Reduced Network . . . . .	58
<b>5</b>	<b>N-1 Contingency Simulation and Power Flow Results for Full and Reduced Networks</b>	<b>61</b>
5.1	Methodology for N-1 Contingency Simulation . . . . .	61
5.2	Graph Representation and Branch Removal . . . . .	62
5.2.1	Network Representation . . . . .	62
5.2.2	Graphical Representation of Radial and Non-Radial Branches . . . . .	62
5.2.3	Flowchart Description: Radial and Non-Radial Branch Detection . . . . .	65
5.3	Contingency Mapping Before N-1 Contingencies Simulation . . . . .	67
5.4	Power Flow Analysis for Contingencies . . . . .	67
5.4.1	Full Network Simulation . . . . .	69
5.4.2	Reduced Network Simulation . . . . .	69
5.5	Recovering Original Indices and Updating Contingency Results for the Reduced Network . . . . .	69
5.5.1	Recovering Original Indices for the Reduced Network . . . . .	70
5.5.2	Updating Contingency Results with Original Bus Numbers . . . . .	70
5.6	Comparative Analysis of Contingencies in Full and Reduced Power Networks	73
5.6.1	Identification of Common Contingencies . . . . .	73
5.6.2	Comparative Evaluation of Electrical Parameters . . . . .	75
5.6.3	Significance of the Analysis . . . . .	76
5.7	Results interpretation and Discussion . . . . .	77
5.7.1	Voltage Difference and Relative Error Analysis . . . . .	77
5.7.2	Voltage Differences and Significance . . . . .	79
5.7.3	Generator Bus Status Change from PV-to-PQ . . . . .	81
5.7.4	Relative Error Analysis for Reactive Power (Q) . . . . .	84
5.7.5	Relative Error Analysis for Branch Current (I) Difference . . . . .	86
5.8	Experimenting with Boundary Nodes of the Reduced Network . . . . .	88
5.8.1	Relative Voltage Error ( $\Delta V/V$ ) . . . . .	91
5.8.2	Voltage Difference Plot . . . . .	92
5.8.3	Generator Bus Status Shift (PV to PQ) . . . . .	92

5.8.4	Relative Branch Current Error ( $\Delta I/I$ ) . . . . .	93
5.8.5	Relative Reactive Power Error ( $\Delta Q/Q$ ) . . . . .	93
<b>6</b>	<b>Conclusion</b>	<b>97</b>
	<b>Bibliography</b>	<b>99</b>
<b>A</b>	<b>Appendix A</b>	<b>101</b>
	<b>List of Figures</b>	<b>157</b>
	<b>List of Tables</b>	<b>159</b>
	<b>List of Symbols</b>	<b>161</b>
	<b>Acknowledgements</b>	<b>163</b>



# 1 | Introduction

The global energy landscape is undergoing a significant transformation, driven by the rapid integration of renewable energy sources (RES) such as wind, solar, and hydroelectric power. This shift is fundamentally reshaping the structure and operation of modern electricity grids, making them increasingly decentralized, dynamic, and complex. These changes are particularly evident in Europe, where the European Union (EU) has set ambitious climate and energy targets. These include achieving carbon neutrality by 2050 and reducing greenhouse gas emissions by at least 55% by 2030, relative to 1990 levels. The EU's "*Fit for 55*" initiative aims to achieve these reductions while significantly increasing the share of renewable energy in the energy mix [5]. Attaining these goals will require substantial adaptations to the European electricity grid, one of the largest and most interconnected systems globally.

The European electricity grid is an extensive and intricate network, comprising 312,693 kilometers of high-voltage transmission lines and serving approximately 532 million people across 35 countries [15][20][19]. The grid includes around 10,000 high-voltage substations and tens of thousands of transformers, all crucial to maintaining grid stability and reliability. Additionally, the network is characterized by nearly 400 cross-border interconnectors that facilitate the efficient transfer of electricity across national boundaries, promoting the integration of renewable energy [2].

Italy, with its strategic location, abundant energy resources, and diverse energy mix, plays a pivotal role within the broader European power system. The country's electricity infrastructure is deeply integrated into the European grid, consisting of approx. 72,000 kilometers of transmission lines and 881 substations [16][18]. Italy's energy mix heavily relies on renewable sources, with hydroelectric power contributing a significant share of its electricity generation, supplemented by solar, wind, and geothermal energy. The Italian grid is connected to neighboring countries, including France, Switzerland, Austria, and Slovenia, via cross-border interconnectors, further emphasizing its strategic importance in the European electricity market.

## 1.1. Growing Complexity and the Need for Advanced Tools

The increasing integration of intermittent renewable energy sources and the growing interconnection of national and international grids have made modern power systems more complex than ever before. These developments have created a significant demand for advanced analytical tools to handle the complexities inherent in large-scale power systems. The EU's climate and energy goals underscore these challenges, pushing power systems toward a future where high levels of renewable energy must be integrated without compromising grid stability.

Given these challenges, the need for optimal, accurate, and reliable power system models has never been more pressing. In recent decades, network reduction techniques have emerged as essential tools for managing the intricacies of power networks effectively. These methodologies simplify complex power systems into smaller, more manageable models—called equivalents—that preserve the essential electrical properties of the original system. This simplification allows for a focused study of specific areas of the network [9].

## 1.2. The Role of Network Reduction in Power Systems

Modern power systems are becoming increasingly complex due to the growing penetration of renewable energy sources and expanding interconnections between regional grids. This complexity requires advanced analytical tools capable of accurately modeling grid behavior under various scenarios. Due to the immense size and complexity of contemporary power systems, traditional methods that involve detailed models of the entire grid are becoming progressively impractical.

Network reduction techniques address these challenges by simplifying complex power systems into smaller, more manageable models that retain the essential characteristics of the original system. Reduced models allow engineers and system operators to focus on specific regions or components of the network, such as the Italian grid, without being overwhelmed by the complexity of the entire European grid.

### 1.2.1. Streamlining Real-Time Analysis

One of the key advantages of network reduction is its ability to facilitate real-time analysis and decision-making. In grid operations, where timely and accurate information is essential, the ability to rapidly assess the impact of contingencies or operational changes is highly valuable. By reducing the network to a manageable size, network reduction techniques allow operators to simulate various scenarios, such as N-1 contingencies, in real-time, enabling prompt responses to potential issues. This real-time capability is crucial in ensuring grid stability, particularly as networks become more dynamic with increased renewable energy integration.

As we approach the 2030 grid scenario, where renewable energy sources (RES) are expected to account for a larger share of the energy mix, the variability in generation will demand more flexible and responsive grid management. In this context, network reduction techniques will play an even more significant role, not only in enabling real-time contingency analysis but also in simulating hourly power flow and market solutions. This extended capability is essential for evaluating grid performance and market operations over longer periods, such as a month or a year, under the influence of intermittent RES. By integrating predictive models for RES output, network reduction can accommodate future scenarios that reflect the fluctuating nature of renewable energy and its impact on national or regional grids.

These simulations will help operators anticipate and plan for fluctuations in energy supply, ensuring that the grid remains balanced and reliable. Additionally, by streamlining power flow solutions and market forecasts over extended time frames, network reduction enhances decision-making processes for both operational and market-related strategies, making it an indispensable tool for managing the evolving grid landscape in the 2030s.

### 1.2.2. Enhancing Computational Efficiency

The use of network reduction techniques significantly improves computational efficiency. Large-scale power systems, such as the European grid, involve complex interconnections between numerous nodes, transmission lines, and generation units. Analyzing these interactions in detail can be computationally demanding, making it difficult to perform comprehensive studies or real-time assessments. Network reduction techniques simplify the network by reducing the number of nodes and connections that need to be analyzed, resulting in faster computation times and more efficient use of computing resources. This allows for detailed studies on smaller systems without sacrificing accuracy.

For example, the European grid, with its 312,693 kilometers of transmission lines and numerous substations and transformers, requires substantial computational resources for accurate modeling. By applying network reduction techniques, it becomes possible to focus on specific regions, such as the Italian grid, which, while still complex with 72,000 kilometers of transmission lines and several interconnections, is more manageable for detailed analysis. These reductions save time and enable more frequent and iterative studies, which are crucial for adapting to rapidly evolving grid configurations.

### 1.2.3. Focusing on Critical Areas

Network reduction techniques also allow for a more focused analysis of critical areas of the network. In large power systems like the European grid, certain regions or nodes may have a greater impact on overall grid stability and reliability than others. Reducing the network to a smaller equivalent that retains the key characteristics of these critical areas enables operators to conduct more detailed studies that would otherwise be infeasible on the full network model. For instance, Italy, with its strategic interconnections to neighboring countries, plays a crucial role in maintaining the stability of the European network. A focused analysis of a reduced model of the Italian grid allows system operators to better understand the interactions between this region and the larger system, identifying vulnerabilities or areas for improvement.

### 1.2.4. Challenges and Limitations

While network reduction offers several advantages, it also comes with its own set of challenges and limitations. One major challenge is maintaining the accuracy of the reduced model, particularly in representing interactions with the external system. The Ward's equivalent method, while designed to preserve key electrical characteristics, always involves a trade-off between simplicity and accuracy. An oversimplified representation of dynamic interactions between the internal and external systems can lead to potential inaccuracies in the analysis. Additionally, the accuracy of the reduced model may be affected by changes in operating conditions, such as variations in load, generation, or network topology, which may not be fully captured in the reduced model.

These challenges are particularly pronounced in large interconnected systems like the European grid, where cross-border flows and oscillations between regions play a significant role. The difficulty lies in ensuring that the reduced model accurately captures the interconnections and feedback mechanisms inherent in a network of this scale. It is essential to ensure that these factors are properly accounted for in the reduced model to maintain

the accuracy of the analysis, particularly when making decisions that affect the stability and reliability of the grid.

### 1.3. Network Reduction Techniques in Power Systems

Techniques for network reduction, such as Ward's Equivalent (Ward's Injection), are growing in importance in contemporary power system analysis, particularly given the immense scale and complexity of networks like the European grid. These techniques simplify complex systems while retaining their essential characteristics, allowing engineers and system operators to focus on specific regions of the network.

Network reduction has proven indispensable for various types of analyses, including load flow analysis, stability evaluations, and contingency studies. These analyses are crucial for ensuring the reliability and stability of power systems, particularly as they evolve to meet future energy needs. Additionally, network reduction methods are invaluable when computing resources and time are limited, such as in real-time operational analysis and contingency planning [4].

#### 1.3.1. Ward's Equivalent: A Core Network Reduction Technique

Ward's Equivalent, introduced by John B. Ward in 1949, is one of the most established network reduction techniques and has been widely researched for its effectiveness in steady-state analysis and contingency testing [17]. This method simplifies the external network by converting it into an equivalent system of admittances or currents, preserving the fundamental electrical properties of the original network. The transformation allows for a more manageable analysis of the internal network while still accounting for the influence of the external system.

The primary advantage of Ward's Equivalent lies in its ability to simplify network infrastructure without sacrificing accuracy in analyzing key system parameters such as voltages, power flows, and currents. This approach is especially effective for large interconnected power networks, like the European grid, where the vast number of nodes and interconnections would otherwise make comprehensive analysis impractical. Ward's Equivalent has been extensively used in academic research and practical power system studies, proving its versatility across a range of applications [9].

Monticelli et al. (1979) explored the use of Ward’s Equivalent for real-time contingency analysis, demonstrating its effectiveness in streamlining complex networks and reducing computational demands. Their study showed that Ward’s Equivalent could produce accurate results for critical system parameters, such as voltages and power flows, even in large systems like the European grid [9].

### 1.3.2. Applications in Large-Scale Power Networks

The European grid, with its 312,693 kilometers of high-voltage transmission lines, hundreds of substations, transformers, and cross-border interconnectors, presents significant challenges for system analysis, particularly in real-time operational analysis and contingency planning [15][20][19]. Ward’s Equivalent offers a practical solution by simplifying the analysis of these large-scale networks while preserving essential system characteristics. Monticelli et al. (1979) has shown that Ward’s Equivalent can produce accurate and reliable results in large-scale networks like the European grid [9].

### 1.3.3. Other Network Reduction Techniques

While Ward’s Equivalent is a widely used method, several other network reduction techniques have been developed to address some of its limitations, particularly in dynamic studies.

The **Radial Equivalent Independent (REI) Model**, developed by P. Dimo in 1975, simplifies external nodes by consolidating them into a single equivalent node. This method is particularly useful in systems with strong interconnections between internal and external components, such as the European grid. However, the REI model has limitations, especially in accurately representing inter-zonal power flows and the dynamics of external generators [3][1].

The **Extended Ward’s Equivalent (WX)** expands on the original Ward’s Equivalent by incorporating additional details, such as generators and loads that are directly connected to the internal system. This method provides a more accurate representation of the external system’s impact on the internal network, making it valuable for dynamic analysis. However, its higher computational complexity makes it less suitable for real-time applications [8].

**Kron Reduction**, introduced by Gabriel Kron in 1939, simplifies power systems by removing internal nodes while maintaining the external network’s connections. This method is particularly effective for small to medium-sized networks but can become computation-

ally expensive for large-scale systems [7][1].

**Dynamic Equivalents** are used to simulate the behavior of power systems during transient events, such as faults or sudden changes in load or generation. These equivalents provide a more accurate depiction of the system's response to changing conditions, but they are computationally demanding and require more detailed data and expertise for implementation. Despite these challenges, dynamic equivalents are essential in studying the impact of disturbances on system stability and reliability, especially in real-time operational scenarios [10].

### 1.3.4. Recent Developments in Network Reduction

Recent advances in machine learning and artificial intelligence have opened new possibilities for improving network reduction techniques. For example, Ivica Pavić et al.(2001) proposed a neural network-based approach to develop more adaptable and accurate equivalents for external systems [11]. These methods show promise for enhancing the accuracy of simplified models, particularly in dynamic environments requiring real-time analysis. However, applying these methods to large systems like the European grid is still an emerging field with challenges that need to be addressed.

## 1.4. Network Reduction Techniques in Distribution Networks

Network reduction techniques, traditionally applied to transmission networks, are increasingly important in distribution networks due to the growing integration of distributed energy resources (DERs) like rooftop solar panels, wind turbines, and energy storage. These resources are transforming distribution networks from passive systems into active ones, necessitating advanced monitoring and control. The bidirectional power flows and fluctuating generation from renewable sources introduce challenges related to voltage management and system reliability.

Shapovalov et al. (2013) emphasize that network reduction simplifies unobserved areas of the grid, enabling more accurate analysis of critical nodes even when real-time data is limited. This technique aggregates unmonitored sections of the network, allowing operators to focus on essential grid behaviors [13]. The introduction of forecast-based reconfiguration algorithms further enhances network management by predicting power flows and minimizing the need for frequent switching operations. The reduced network models generated by these algorithms improve operational efficiency, allowing for timely

reconfiguration and optimization [14].

## 1.5. N-1 Contingency Analysis and the Role of Ward's Equivalent

N-1 contingency analysis is a critical aspect of power system planning and operation. It involves simulating scenarios where a single component, such as a transmission line or generator, fails and assessing the system's ability to continue operating without violating operational limits. Ward's Equivalent (WI) has been extensively validated in the literature for its effectiveness in handling N-1 contingency scenarios [12].

One of Ward's Equivalent's key strengths is its ability to maintain the accuracy of voltage and power flow characteristics in the reduced model, which is crucial for reliable contingency analysis. Research by Housos et al. (1980) and M. Ramirez-Gonzalez et al.(2022) demonstrated that Ward's Equivalent (WI) could accurately represent the external system's impact on the internal network during N-1 contingencies, providing a reliable basis for operational decision-making [6][12].

## 1.6. Evaluating the Effectiveness and Limitations of Network Reduction Techniques

### 1.6.1. Effectiveness

The effectiveness of network reduction techniques is determined by several factors, including the accuracy of the reduced model in representing the original system, computational speed, and the ability to retain the system's key characteristics. Empirical studies have demonstrated that Ward's Equivalent (Ward's Injection) is highly effective at preserving voltage and power flow characteristics in steady-state analysis [12]. Its simplicity and computational efficiency make it a practical choice for a wide range of power system studies.

### 1.6.2. Limitations

Every network reduction technique has inherent limitations that must be carefully considered when selecting the appropriate method for a particular study. Ward's Equivalent is primarily designed for steady-state analysis, meaning that it may not fully capture the dynamic behavior of the system as operating conditions change. The accuracy of the

method is also affected by the choice of the boundary between the internal and external networks, which can result in errors in the equivalent system.

## 1.7. Rationale for Selecting Ward's Equivalent

Ward's Equivalent (Ward's Injection) was selected for this research due to its proven effectiveness in maintaining the balance between simplicity and accuracy in large-scale power system analyses [12]. Its ability to preserve the critical electrical properties of the original system while significantly reducing computational complexity makes it highly suitable for studying intricate networks. This method's reliability in steady-state analysis and contingency planning, particularly in large interconnected systems, establishes its relevance in this study.

In this thesis, the focus is on applying Ward's Equivalent (Ward's Injection) to a complex 66-bus grid, referred to as *Rueda\_network\_3\_1* through MATLAB codes. The objective is to evaluate the method's performance in reducing the network while maintaining accuracy under normal and stressed conditions, such as during N-1 contingency analysis. A key aspect of this study is the identification and selection of boundary nodes, which play a critical role in ensuring that the reduced network accurately reflects the behavior of the full system. By doing so, the analysis not only assesses the reduction's computational efficiency but also its precision in replicating the system's response to contingencies.

To support this, Figure 1.1 below presents the single-line diagram of the *Rueda\_network\_3\_1*. This diagram illustrates the intricate structure of the grid, highlighting key components such as buses, transformers, and transmission lines across different voltage levels. The visualization underscores the complexity of the network and justifies the need for an efficient reduction technique like Ward's Equivalent (Ward's Injection).

Through the application of Ward's Equivalent (Ward's Injection), the reduced model will serve as a simplified yet accurate representation of this complex grid, allowing for detailed steady-state and contingency analyses without overwhelming computational demands.

## 1.8. Research Gaps and Future Directions

While network reduction techniques like Ward's Equivalent offer valuable tools for simplifying complex power grids, several research gaps remain. Most notably, existing techniques are designed primarily for steady-state analysis and may not fully capture dynamic behaviors, particularly in real-time operational contexts. Dynamic equivalents, for example, can provide a more accurate depiction of system responses during transient events,

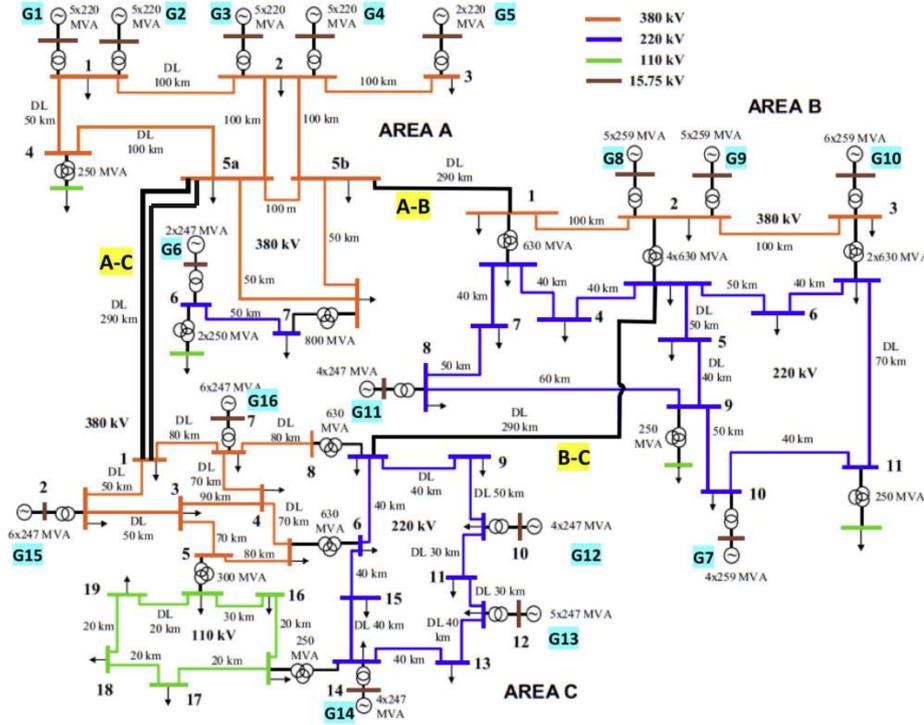


Figure 1.1: Single-line diagram of the 66-bus *Rueda\_network\_3\_1* used in this thesis for applying Ward's network reduction technique.

but they are computationally intensive and require detailed data [10]. Further research is needed to develop network reduction techniques that can efficiently balance the trade-off between simplicity and accuracy in both steady-state and dynamic analyses.

Additionally, emerging technologies such as machine learning and artificial intelligence (AI) offer promising opportunities for enhancing network reduction techniques. Ivica Pavić et al.(2001) proposed a neural network-based approach that could potentially create more adaptable and accurate equivalents for external systems. These methods show potential in improving the accuracy of reduced models in dynamic environments but remain underexplored in large-scale systems like the European grid. The development of machine learning-based equivalents capable of real-time analysis could address some of the limitations currently encountered in network reduction methods. However, more research is needed to explore these techniques and overcome the challenges of applying AI to power system models [11].

# 2 | Network Reduction Techniques for Steady-State Analysis

As discussed in the previous chapter, network reduction techniques are essential for simplifying large and complex electrical networks, particularly for steady-state analyses such as load flow, contingency studies, and stability assessments. These techniques reduce computational complexity while maintaining the key electrical characteristics of the system at boundary nodes of interest. This chapter discusses three prominent network reduction techniques: **Ward's Equivalent**, **Radial Equivalent Independent (REI) model**, and **Kron Reduction**. Each of these techniques is explored in terms of its mathematical formulation, advantages, limitations, and application in power system analysis. Special emphasis is also given to the variants of Ward's Equivalent, showcasing its adaptability depending on the analysis needs.

## 2.1. Ward's Equivalent

Developed by J.B. Ward in 1949, Ward's Equivalent is a classical method for simplifying large power networks by eliminating external nodes while preserving the electrical characteristics at the internal nodes. This technique is widely used in load flow studies, contingency analysis, and stability assessments, providing a reduced model that retains the system's essential electrical properties [17].

### 2.1.1. Principles of Ward's Equivalent

Ward's Equivalent divides the network into *internal* and *external* nodes. The internal nodes are retained for detailed analysis, while the external nodes are eliminated. The effect of the external nodes on the internal nodes is modeled using an equivalent admittance matrix, allowing the reduced system to behave similarly to the original network from the perspective of the internal nodes [17].

### 2.1.2. Mathematical Formulation

Ward's Equivalent begins with the system's *admittance matrix*,  $Y$ , which describes the electrical relationships between all nodes (buses) in the network. The matrix is partitioned into internal ( $i$ ) and external ( $e$ ) nodes as follows:

$$Y = \begin{bmatrix} Y_{ii} & Y_{ie} \\ Y_{ei} & Y_{ee} \end{bmatrix}$$

Where:

- $Y_{ii}$ : Admittance matrix of the internal nodes,
- $Y_{ee}$ : Admittance matrix of the external nodes,
- $Y_{ie}$  and  $Y_{ei}$ : Coupling admittance between internal and external nodes.

Assuming no current injections at the external nodes ( $I_e = 0$ ), the voltage at the external nodes can be expressed as:

$$V_e = -Y_{ee}^{-1}Y_{ei}V_i$$

Substituting this into the equation for the internal nodes gives the *equivalent admittance matrix*:

$$Y_{ii}^{\text{equivalent}} = Y_{ii} - Y_{ie}Y_{ee}^{-1}Y_{ei}$$

This reduced matrix retains the interaction between internal nodes, incorporating the influence of the eliminated external nodes.

### 2.1.3. Applications in Power Systems

Ward's Equivalent is widely used in several areas of power system analysis:

- **Load Flow Analysis:** It simplifies external network components, enabling efficient computation of load flow for specific regions.
- **Contingency Analysis:** Reduces the number of nodes to allow faster and more focused reliability assessments [12].
- **Stability Studies:** Allows detailed analysis of internal nodes by modeling the

external system as an equivalent network.

#### 2.1.4. Variants of Ward's Equivalent

Ward's Equivalent has evolved into several variants, each of which addresses different aspects of network reduction. The three main variants are the **Ward Admittance Method (WA)**, **Ward Injection Method (WI)**, and **Extended Ward Method (WX)**.

##### Ward Admittance Method (WA)

The **Ward Admittance Method (WA)** is the simplest version of Ward's Equivalent. It eliminates external nodes by converting their power injections into equivalent *shunt admittances*, modeling the external nodes as passive elements that are connected to the internal system.

**Mathematical Formulation** In WA, the system's admittance matrix is partitioned as follows:

$$\begin{bmatrix} I_i \\ I_e \end{bmatrix} = \begin{bmatrix} Y_{ii} & Y_{ie} \\ Y_{ei} & Y_{ee} \end{bmatrix} \begin{bmatrix} V_i \\ V_e \end{bmatrix}$$

Assuming  $I_e = 0$ , the voltage at the external nodes is:

$$V_e = -Y_{ee}^{-1}Y_{ei}V_i$$

Substituting this into the equation for internal nodes gives:

$$I_i = Y_{ii}V_i - Y_{ie}Y_{ee}^{-1}Y_{ei}V_i$$

Thus, the equivalent admittance matrix for the internal nodes is:

$$Y_{ii}^{\text{equivalent}} = Y_{ii} - Y_{ie}Y_{ee}^{-1}Y_{ei}$$

**Applications and Benefits** WA is suitable for large power systems where the external nodes can be treated passively, making it an efficient method for load flow studies or preliminary network reductions.

**Limitations** WA assumes that external nodes are passive, which limits its accuracy in cases where external nodes contribute actively to the power flow. This method is less appropriate for scenarios involving complex interactions between external and internal nodes [12].

## Ward Injection Method (WI)

The **Ward Injection Method (WI)** enhances the WA method by retaining the *current injections* at the external nodes, providing a more accurate representation of the external system as active components influencing the internal network.

**Mathematical Formulation** The system's current and voltage relationships can be expressed as:

$$\begin{bmatrix} I_i \\ I_e \end{bmatrix} = \begin{bmatrix} Y_{ii} & Y_{ie} \\ Y_{ei} & Y_{ee} \end{bmatrix} \begin{bmatrix} V_i \\ V_e \end{bmatrix} \quad (1)$$

Where  $I_i$  and  $V_i$  correspond to the current and voltage at the retained buses, while  $I_e$  and  $V_e$  represent the same quantities at the buses being eliminated. By manipulating  $V_e$  as shown in equation (2) and substituting it into the first row of (1), we obtain a generalized expression for the equivalent network in equation (3).

$$V_e = Y_{ee}^{-1}(I_e - Y_{ei}V_i) \quad (2)$$

$$I_i = (Y_{ii} - Y_{ie}Y_{ee}^{-1}Y_{ei})V_i + Y_{ie}Y_{ee}^{-1}I_e \quad (3)$$

In cases where the external network's generation and loads are converted into shunt admittances, we can assume  $I_e = 0$  in equation (1). Consequently, the equivalent network is fully characterized by the first term in equation (3), describing the retained buses and the shunt components connecting them. Alternatively, if the external generation and loads are converted into constant current injections, the second term of equation (3) becomes relevant, providing the equivalent injected currents necessary to replicate the impact of the eliminated sections of the network.

**Applications and Benefits** WI is particularly useful in **contingency analysis** and **stability studies**, where external nodes actively influence the internal system. By retaining the external current injections, WI provides a more realistic representation of the

power flow in complex networks [12].

**Limitations** Retaining external injections increases the *computational complexity* of the reduction process, making WI more resource-intensive than WA.

## Extended Ward Method (WX)

The **Extended Ward Method (WX)** builds upon WI by introducing *fictitious generator buses* at boundary nodes to model *reactive power flows*, improving voltage stability and power flow representation.

**Mathematical Formulation** In WX, fictitious generator buses are added to support or absorb reactive power at the boundary nodes, thus enhancing the modeling of voltage stability. The method retains all the features of WI, with additional reactive power compensation through the fictitious generators [8].

**Applications and Benefits** WX is ideal for power systems that require accurate **reactive power compensation** and **voltage stability analysis**, especially in scenarios where the external network plays a significant role in the system's voltage stability [8].

**Limitations** The inclusion of fictitious generators increases the *computational complexity*, making WX more resource-intensive than both WA and WI.

### 2.1.5. Advantages and Limitations of Ward's Equivalent

Ward's Equivalent and its variants provide flexible tools for simplifying large power systems while offering a balance between computational efficiency and accuracy:

- **Advantages:**

- *Computational Efficiency:* The WA variant is especially efficient for reducing system complexity.
- *Adaptability:* WI and WX provide more detailed modeling of external nodes as active participants, with WX offering improved voltage stability through reactive power compensation.

- **Limitations:**

- *Linearity Assumptions:* All variants assume linearity, limiting their application in dynamic or highly non-linear systems.

- *Increased Complexity*: WI and WX introduce additional complexity, which may lead to higher computational costs.

## 2.2. Radial Equivalent Independent (REI) model

The **Radial Equivalent Independent (REI) model** is a network reduction technique that preserves both the load characteristics and impedance interactions between internal and external nodes. Unlike Ward's Equivalent, which directly modifies the admittance matrix, the REI method introduces *fictitious reference buses* (REI buses) to represent the external system. This approach is particularly useful in cases where the impedance and load characteristics of the external system need to be preserved.

### 2.2.1. Principles of REI Equivalent

The REI method involves:

1. **Identifying boundary buses**: The system is divided into internal and external nodes, similar to Ward's Equivalent.
2. **Creating REI buses**: Fictitious REI buses are introduced to represent the aggregated effect of external nodes on the internal nodes.
3. **Recalculating admittances**: The admittance values between the internal nodes and REI buses are recalculated to reflect the external system's influence accurately.

By doing this, the external system is replaced by its REI equivalent, which retains both impedance and load characteristics while reducing the computational complexity [12].

### 2.2.2. Applications in Power Systems

The REI method is particularly useful in:

- **Contingency analysis**: Simplifies external networks while maintaining accurate load and impedance characteristics.
- **Load flow studies**: Especially in cases where detailed modeling of the external system is unnecessary, but its effects on the internal system need to be accounted for.

### 2.2.3. Advantages and Limitations

- **Advantages:**

- *Flexibility:* The introduction of REI buses allows for a more accurate representation of external systems compared to Ward’s Equivalent.
- *Load Representation:* The REI method accurately preserves the load characteristics and impedance of the external system.

- **Limitations:**

- *Computational Complexity:* The creation of REI buses increases computational complexity.
- *Linearity Assumption:* Like Ward’s method, the REI method assumes a linear system, limiting its applicability in non-linear analyses.

## 2.3. Kron Reduction

**Kron Reduction**, also known as *Kron’s Elimination*, is a mathematical technique for reducing the size of an admittance matrix by eliminating non-boundary nodes, thereby retaining only the electrical relationships between boundary nodes. Kron Reduction is widely used in power system and circuit analysis due to its mathematical rigor and applicability to a wide range of problems.

### 2.3.1. Principles of Kron Reduction

Kron Reduction eliminates *interior nodes* (non-boundary nodes), reducing the size of the system’s admittance matrix while maintaining the electrical relationships between the *boundary nodes*. The goal is to compute a new admittance matrix for the boundary nodes that reflects the effect of the eliminated interior nodes.

### 2.3.2. Mathematical Formulation

Let the system’s admittance matrix be partitioned into boundary ( $b$ ) and interior ( $i$ ) nodes as follows:

$$Y = \begin{bmatrix} Y_{bb} & Y_{bi} \\ Y_{ib} & Y_{ii} \end{bmatrix}$$

Where:

- $Y_{bb}$  corresponds to the boundary nodes,
- $Y_{bi}$  and  $Y_{ib}$  represent the coupling between boundary and interior nodes,
- $Y_{ii}$  corresponds to the interior nodes.

Using *Schur's Complement*, the reduced admittance matrix for the boundary nodes is given by:

$$Y_{bb}^{\text{reduced}} = Y_{bb} - Y_{bi}Y_{ii}^{-1}Y_{ib}$$

This results in a system where only the boundary nodes remain, while the interior nodes are eliminated, and their influence is incorporated into the equivalent admittance matrix.

### 2.3.3. Applications in Power Systems

Kron Reduction is useful for:

- **Impedance-based load flow analysis:** When only a subset of the network (i.e., boundary nodes) is required for analysis.
- **Stability studies:** Kron Reduction is often used to simplify networks for dynamic simulations while preserving key electrical relationships between boundary nodes.

### 2.3.4. Advantages and Limitations

- **Advantages:**
  - *Mathematically Rigorous:* Provides an exact method for eliminating nodes while maintaining accurate relationships between the remaining nodes.
  - *Applicable to Dynamic Studies:* Kron Reduction is not limited to steady-state analysis but can also be applied in dynamic studies.
- **Limitations:**
  - *Computational Complexity:* Kron Reduction involves matrix inversion, which can be computationally expensive for large systems.
  - *Linearity Assumption:* Similar to other reduction techniques, Kron Reduction assumes the network is linear, which limits its application to non-linear systems [7].

## Comparison of Network Reduction Techniques

The three network reduction techniques discussed—**Ward’s Equivalent**, **REI Equivalent**, and **Kron Reduction**—each have distinct strengths, applications, and limitations in electrical network analysis. Here’s a comparative analysis:

Technique	Application	Advantages	Limitations
<b>Ward’s Equivalent</b>	Steady-state analysis, contingency studies, load flow analysis	Simple, computationally efficient	Assumes linearity, less effective for non-linear or dynamic systems
<b>REI Equivalent</b>	Load-flow studies, contingency analysis, operational planning	Flexible, preserves load characteristics	More complex, computationally intensive
<b>Kron Reduction</b>	Impedance and admittance matrix reduction, dynamic studies	Rigorous, reduces size of network	Requires matrix inversion, assumes linearity

Table 2.1: Comparison of Network Reduction Techniques

Network reduction techniques, including **Ward’s Equivalent**, **REI Equivalent**, and **Kron Reduction**, play a vital role in simplifying large power systems and improving computational efficiency without compromising accuracy. Ward’s Equivalent offers multiple methods—WA, WI, and WX—that provide varying levels of accuracy and complexity depending on the needs of the analysis. The REI Equivalent method preserves critical load and impedance characteristics, making it valuable in specific scenarios, while Kron Reduction provides a mathematically rigorous approach to reducing network size. Selecting the appropriate reduction technique depends on the analysis objectives, network characteristics, and the desired balance between computational efficiency and accuracy.



# 3 | Algorithm for Network Reduction and Analysis

This chapter introduces a robust algorithm designed to reduce the complexity of a large-scale power grid model, providing a streamlined yet accurate representation of the system. The focus of the algorithm is to retain the critical electrical characteristics of the grid's internal and boundary nodes, particularly in regions of interest, i.e., Reduced Network. By employing an efficient **network reduction technique**, the algorithm strategically simplifies the external network, eliminating non-essential nodes while preserving key dynamic and operational attributes. This method strikes a balance between computational efficiency and modeling fidelity, ensuring the reduced model accurately reflects the behavior of the full system.

The reduction process unfolds through a series of carefully orchestrated steps, each tailored to maintain the integrity of the grid's core characteristics while optimizing for computational manageability. These steps serve as the foundation for a model that facilitates high-precision analysis while significantly reducing computational overhead, making it suitable for in-depth power flow analysis under normal and contingency scenario simulations, and long-term planning.

## 3.1. Power Flow Analysis (Stage 1)

The first stage of the algorithm performs a comprehensive **power flow analysis** of the Full network. This is a foundational step to ensure the system's operational conditions are accurately captured before network reduction.

### 3.1.1. Data Loading and Initialization

The process begins by loading essential network data, including bus, branch, and generator information, from a pre-defined `.mat` file, such as `Rueda_network_3_1.mat`. Key parameters such as the base MVA and operational limits are initialized to reflect real-world

conditions.

### 3.1.2. Generator Control Adjustments

To ensure realistic simulation, the generator reactive power ( $Q$ ) control is adjusted, classifying generators as  $PQ$ ,  $PV$ , or reference buses based on their operational modes and voltage regulation requirements.

### 3.1.3. Load Flow Computation

The core of this stage lies in solving the load flow equations using the **Ybus matrix**, which models the system's electrical interactions.

### Ybus Matrix Construction

The Ybus matrix is constructed by calculating the following:

- **Self-admittance for each bus,  $Y_{ii}$ :** This is the sum of the admittances of all lines connected to the bus, including line charging admittances and the shunt admittance at the bus. The self-admittance for bus  $i$  is calculated as:

$$Y_{ii} = \sum_{i \neq k} y_{ik} + \left( \frac{\sum_{i \neq k} (G_{ik} + jB_{ik})}{2} \right) + jB_s$$

Where:

- $y_{ik} = g_{ik} + jb_{ik}$  is the series admittance between buses  $i$  and  $k$  (consisting of conductance  $g_{ik}$  and susceptance  $b_{ik}$ ).
  - $G_{ik}$  is the line charging conductance between buses  $i$  and  $k$ .
  - $B_{ik}$  is the line charging susceptance between buses  $i$  and  $k$ .
  - $B_s$  is the shunt susceptance at bus  $i$ .
- **Mutual admittance between buses  $i$  and  $k$ ,  $Y_{ik}$ :** The mutual admittance between connected buses is given by:

$$Y_{ik} = -y_{ik}$$

Where  $y_{ik}$  is the series admittance between bus  $i$  and bus  $k$ .

- **Transformer effects:** If transformers are present, the effects of transformer impedance

and tap ratios are incorporated into the Ybus matrix. This typically alters both the self-admittance and mutual admittance between the buses connected by the transformer.

### 3.1.4. Complex Power Injection Calculation

The algorithm computes the **complex power injection vector (Sbus)** for each bus to represent the net power injected or drawn by each bus, calculated as:

$$S_{\text{bus}} = \frac{(P_{\text{generation}} - P_{\text{demand}}) + j(Q_{\text{generation}} - Q_{\text{demand}})}{\text{baseMVA}}$$

### 3.1.5. Executing the Load Flow

The load flow analysis proceeds iteratively using the **Newton-Raphson method**, adjusting bus voltages and generator outputs until convergence is achieved.

### 3.1.6. Step-by-Step Execution

Figure 3.1 outlines the steps followed in the algorithm:

1. **Start Function launch\_PF:** Initialize the power flow function.
2. **Load Network Data:** Load the network model from a `.mat` file.
3. **Adjust Generator Control:** Set generator reactive power limits and control modes.
4. **Run Load Flow:** Execute the power flow calculations using the calculated Ybus matrix.
5. **Update System Data:** Adjust bus voltages and generator outputs based on the results.
6. **Calculate  $S_{\text{bus}}$ :** Compute the net complex power injection for each bus.
7. **Ensure Full Ybus:** Verify the completeness of the Ybus matrix after updates.
8. **Save Results:** Store the updated system parameters in a `.mat` file.
9. **Calculate Current  $I$ :** Determine the currents for further analysis.
10. **End Function:** Finalize the computation and prepare results for post-processing.

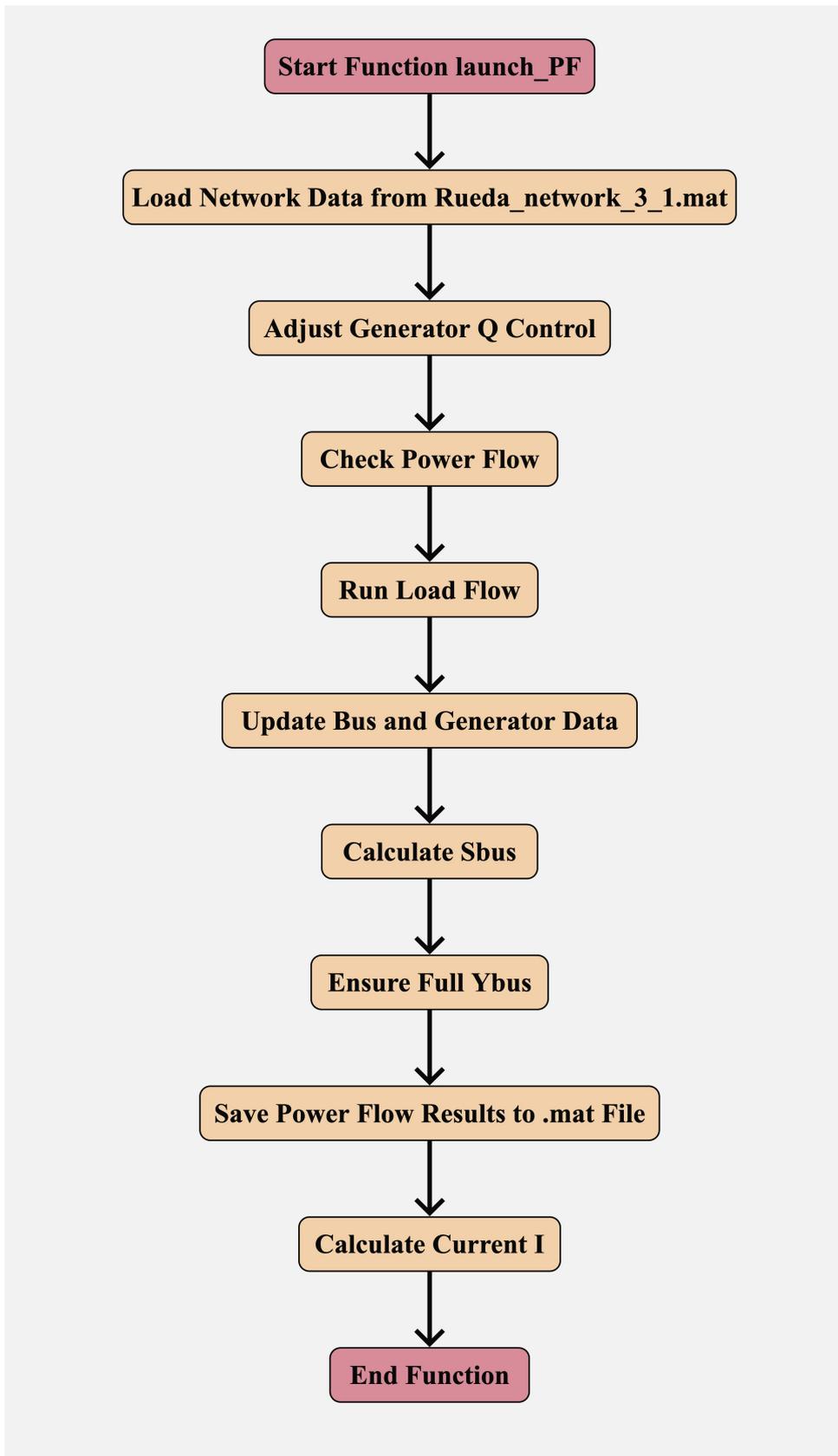


Figure 3.1: Flowchart for Power Flow Analysis Algorithm

### 3.1.7. Finalizing the Load Flow Solution

Once convergence is achieved, the results, including bus voltages, generator outputs, and power flows, are saved. These results form the foundation for network reduction and further contingency analysis, ensuring the system's operational characteristics are preserved.

## 3.2. Identifying Internal, Boundary, and External Nodes (Stage 2)

Once the baseline operating conditions of the network are established, the next critical step is to categorize the nodes into **internal, boundary, and external nodes**. This classification is essential for modelling the Reduced Network from Full network while preserving key interconnections that influence grid operations. The algorithm achieves this classification through a graph-based approach, leveraging the structural properties of the network to ensure accurate identification of node types.

### 3.2.1. Graph Representation of the Network

To efficiently navigate the network and classify nodes, the algorithm first constructs a **network graph**, where:

- **Nodes** represent buses in the power system.
- **Edges** represent branches (transmission lines or transformers) that connect these buses.

By treating the system as a graph, the algorithm can traverse the network efficiently, analyze node adjacencies, and apply classification rules based on electrical connections.

### 3.2.2. Classification of Nodes: Internal, Boundary, and External

The node classification process involves several key steps, focusing on identifying and categorizing the nodes based on their role in the network.

## 1. Loading Known Nodes

The algorithm begins by loading a predefined list of known **boundary nodes** (those connected to regions outside the Reduced Network) and **known internal nodes** (nodes within the Reduced Network connected to boundary nodes). These form the foundation for further classification.

## 2. Identifying Internal Nodes

Using a **breadth-first search (BFS)** approach, the algorithm iteratively expands the list of internal nodes by examining neighboring nodes. Starting from the known internal nodes, the algorithm traverses the network, marking any adjacent nodes that are not boundary nodes as internal.

## 3. Classifying Boundary and External Nodes

Boundary nodes are loaded directly into the algorithm, while any remaining nodes that are not classified as either internal or boundary are labeled as **external nodes**. These external nodes are not essential to the reduced model and are removed from further analysis to simplify the network, reducing computational complexity.

### 3.2.3. Flowchart Description: Identifying Internal Nodes

Figure 3.2 shows the flowchart of the BFS-like process used to identify internal nodes within the network.

The process follows these steps:

1. **Create Graph GNetB:** The graph is created using the network's branch and bus data, where nodes represent buses and edges represent connections.
2. **Load Boundary Nodes:** The algorithm loads predefined boundary nodes, which are nodes connected to external regions.
3. **Load Known Internal Nodes:** The algorithm then loads a list of known internal nodes, which are nodes within the Reduced Network that are connected to boundary nodes.
4. **Ensure Nodes as Row Vectors:** Both the boundary and internal nodes are stored as row vectors to maintain consistency during processing.
5. **Initialize Internal Nodes and Nodes to Check:** The list of internal nodes is initialized, and a queue of nodes to check is created, starting with known internal nodes.
6. **Mark Known Internal Nodes as Visited:** All known internal nodes are marked as visited to prevent revisiting during the iterative process.
7. **Iterative Search (While Loop):** The algorithm enters a loop where it checks each node for neighbors:

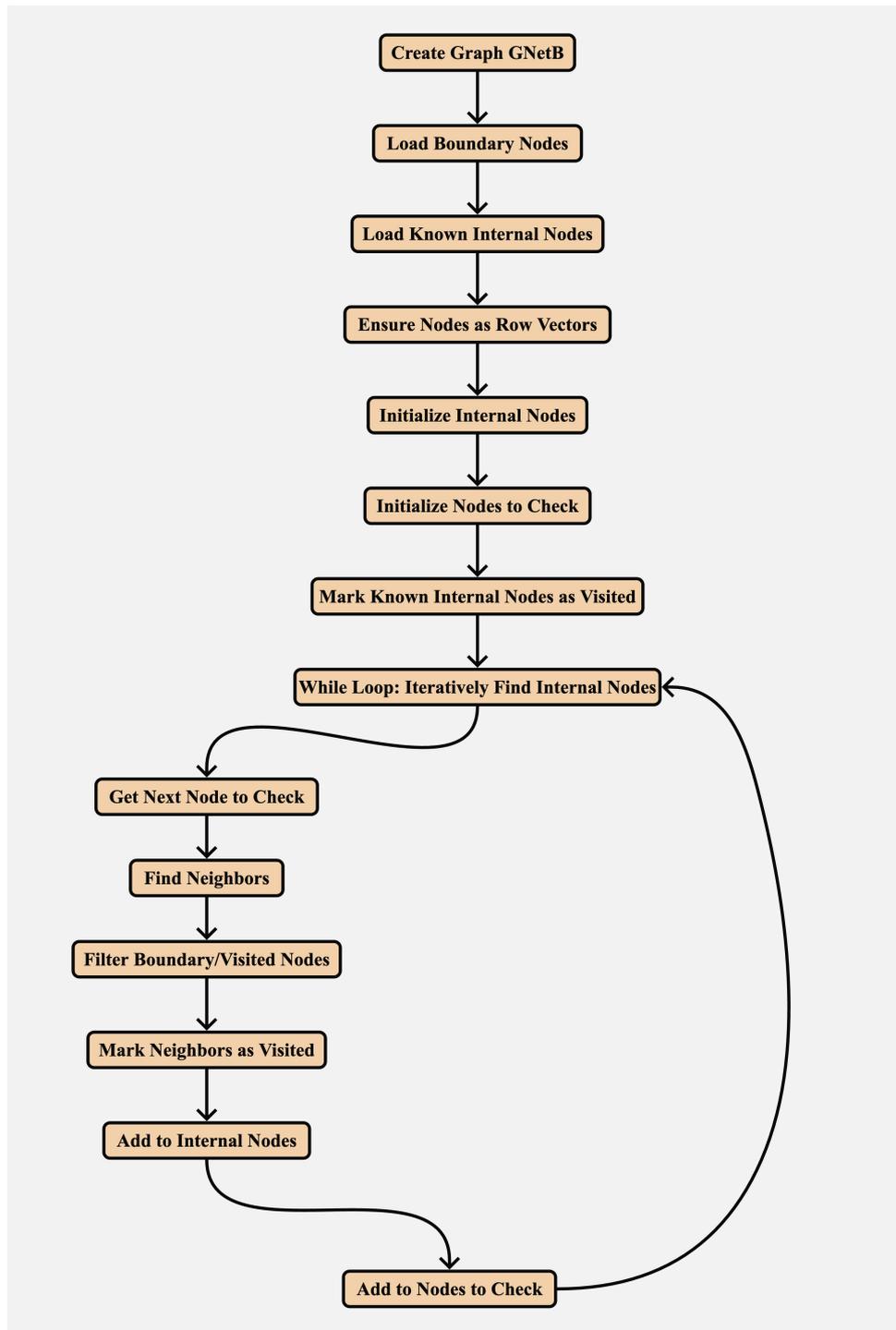


Figure 3.2: Flowchart for the Breadth-First Search-like Approach to Node Identification

- **Get Next Node to Check:** The algorithm retrieves the next node from the queue of nodes to check.
- **Find Neighbors:** It finds the neighboring nodes connected to the current node.
- **Filter Boundary/Visited Nodes:** Any neighboring nodes that are boundary nodes or have already been visited are filtered out.
- **Mark Neighbors as Visited:** Unvisited neighboring nodes are marked as visited.
- **Add to Internal Nodes:** These neighboring nodes are added to the list of internal nodes.
- **Add to Nodes to Check:** The newly identified internal nodes are added to the queue of nodes to check in the next iteration.

This iterative process continues until all internal nodes are identified, ensuring that all relevant nodes within the Full Network are classified properly.

### 3.2.4. Visualization of Node Classification

To facilitate understanding and verification of the node classification, the algorithm provides a visual representation of the network. Each category of node—internal, boundary, and external—is color-coded for easy interpretation.

- **Internal nodes** are highlighted in **green**.
- **Boundary nodes** are highlighted in **blue**.
- **External nodes** are highlighted in **red**.

This visual representation helps verify that the classification accurately reflects the critical structure of the network.

## 3.3. Impact of Node Classification on Network Reduction

The node classification process is vital for the network reduction algorithm. By distinguishing between internal, boundary, and external nodes, the algorithm ensures the visualisation to ease further reduction process. Removing external nodes, which have

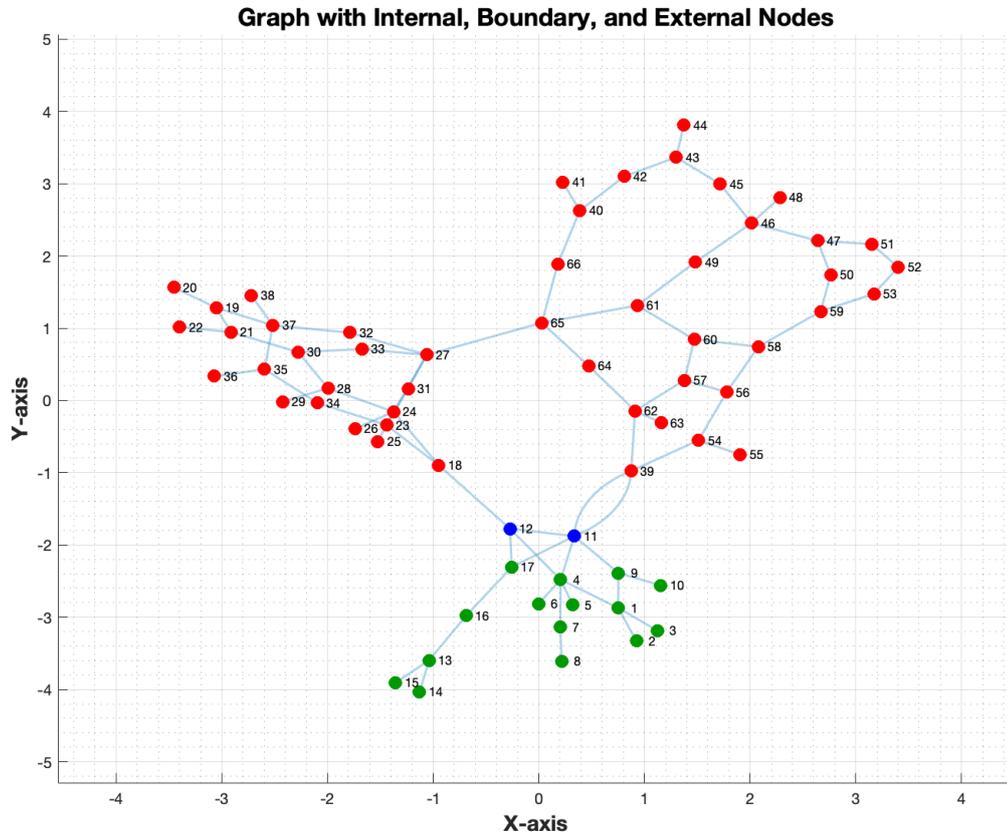


Figure 3.3: Full Network Graph with Internal, Boundary, and External Nodes

minimal impact on the internal dynamics, simplifies the network without compromising its accuracy.

### 3.4. Admittance Matrix Partitioning and Ward's Reduction (Stage 3)

With the nodes classified into internal, boundary, and external categories, the next crucial step is to apply **Ward's reduction** to simplify the network by eliminating external nodes. This process involves partitioning the **bus admittance matrix ( $Y_{bus}$ )** and applying Ward's reduction to maintain the key electrical interactions between internal and boundary nodes while excluding external nodes from the analysis. This step reduces the size of the system, ensuring that the computational model remains manageable without losing accuracy in representing the critical behavior of the full grid.

### 3.4.1. Matrix Partitioning

The first step in Ward's reduction is the partitioning of the **Ybus matrix**, which encapsulates the admittance relationships between all nodes in the network. By reordering and dividing this matrix, the interactions between required nodes (internal and boundary nodes) and external nodes are separated. The partitioning results in four key submatrices:

- **Y<sub>rr</sub>**: Represents the interactions among the required nodes (internal and boundary nodes).
- **Y<sub>re</sub>** and **Y<sub>er</sub>**: Represent the interactions between required nodes and external nodes.
- **Y<sub>ee</sub>**: Represents the interactions solely among the external nodes.

This partitioning is crucial for Ward's reduction, as it isolates the external nodes and allows their effects to be folded into the remaining system structure.

### 3.4.2. Ward's Reduction

Ward's reduction simplifies the network by mathematically "folding" the external nodes into the internal system. This method eliminates external nodes while maintaining their influence on the internal and boundary nodes. The process is performed in two key steps:

1. **Reduced Admittance Matrix Calculation:** The new reduced admittance matrix is calculated by removing the external nodes from the system. This step ensures that the electrical properties of the system are preserved by accounting for the influence of external nodes.

The equation is given by:

$$Y_{rr\_new} = Y_{rr} - (Y_{re} \cdot Y_{ee}^{-1} \cdot Y_{er})$$

Where:

- $Y_{rr\_new}$  is the updated admittance matrix for the retained buses(Required Nodes),
- $Y_{rr}$  is the initial admittance matrix for the retained buses(Required Nodes),
- $Y_{re}$  is the admittance matrix between the retained (Required)and eliminated buses,

- $Y_{ee}^{-1}$  is the inverse of the admittance matrix for the eliminated buses,
  - $Y_{er}$  is the admittance matrix between the eliminated and retained buses(Required Nodes).
2. **Adjusted Current Injection:** The reduced current injection vector is similarly adjusted by accounting for the impact of the eliminated external nodes. This ensures that the system's power balance and current flow dynamics remain consistent with the original system.

The equation is given by:

$$I_{r\_new} = I_r - (Y_{re} \cdot Y_{ee}^{-1} \cdot I_e)$$

Where:

- $I_{r\_new}$  is the updated current at retained buses(Required Nodes),
- $I_r$  is the initial current at retained buses(Required Nodes),
- $Y_{re}$  is the admittance matrix between the retained(Required) and eliminated buses,
- $Y_{ee}^{-1}$  is the inverse of the admittance matrix for eliminated buses,
- $I_e$  is the current at eliminated buses.

### 3.4.3. Partitioning the Reduced Admittance Matrix

After Ward's reduction, the newly formed admittance matrix ( $Y_{rr\_new}$ ) is further partitioned to separately represent the interactions between **internal** and **boundary** nodes. The partitioning results in:

- **Ybb\_new:** Represents the interactions between boundary nodes.
- **Yib\_new** and **Ybi\_new:** Represent the interactions between internal and boundary nodes.
- **Yii\_new:** Represents the interactions among internal nodes.

This partitioning enables more focused analysis of the internal grid and its interfaces with the external system through boundary nodes, ensuring that the simplified model still accurately represents dynamics at critical interfaces.

### 3.4.4. Flowchart: Ward's Reduction Process

Figure 3.4 illustrates the detailed steps of Ward's reduction process applied to the admittance matrix.

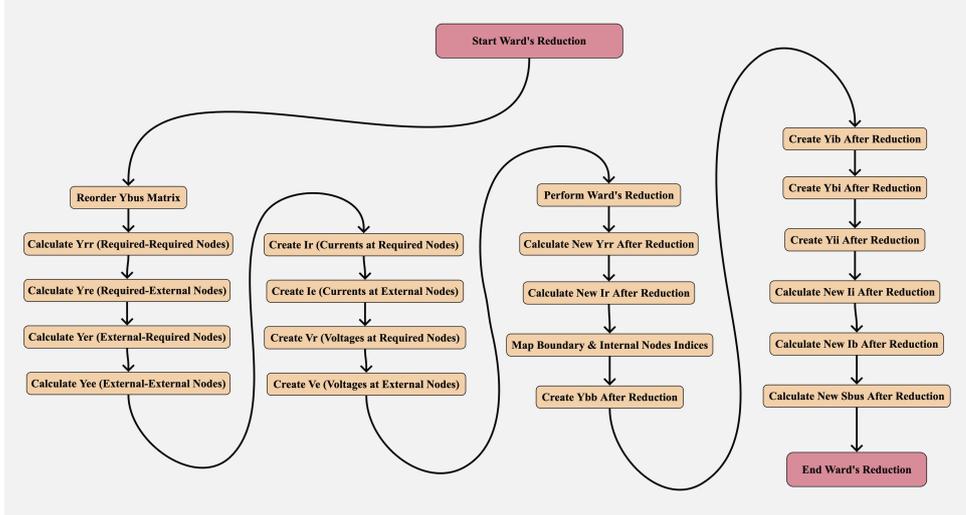


Figure 3.4: Flowchart for Ward's Reduction

The Ward's reduction process can be described as follows:

1. **Start Ward's Reduction:** The process begins by initializing the reduction process.
2. **Reorder Ybus Matrix:** The bus admittance matrix  $Y_{\text{bus}}$  is reordered to separate required nodes (internal and boundary nodes) from external nodes.
3. **Matrix Partitioning:** The reordered  $Y_{\text{bus}}$  is partitioned into four submatrices:
  - $Y_{rr}$ : Represents the interactions among required nodes (internal and boundary nodes).
  - $Y_{re}$  and  $Y_{er}$ : Represent the interactions between required nodes and external nodes.
  - $Y_{ee}$ : Represents the interactions solely among external nodes.
4. **Current and Voltage Calculations:** Currents and voltages at the required nodes ( $I_r, V_r$ ) and external nodes ( $I_e, V_e$ ) are computed.
5. **Perform Ward's Reduction:** The reduction process is applied:
  - Compute the new reduced admittance matrix  $Y_{rr\_new}$  by eliminating the external nodes using  $Y_{re}, Y_{ee}, Y_{er}$ .
  - Adjust the current injections ( $I_{r\_new}$ ) to reflect the influence of external nodes.

6. **Boundary and Internal Node Mapping:** The algorithm maps the newly reduced  $Y_{rr\_new}$  matrix to account for interactions between boundary and internal nodes.
7. **Partition the Reduced Matrix:** The reduced matrix is partitioned into:
  - $Y_{bb\_new}$ : Boundary-boundary interactions.
  - $Y_{ib\_new}$  and  $Y_{bi\_new}$ : Internal-boundary interactions.
  - $Y_{ii\_new}$ : Internal-internal interactions.
8. **End Ward's Reduction:** The process concludes with the calculation of the reduced current injections and the new bus power flows for power flow analysis.

### 3.4.5. Why Ward's Reduction Matters

Ward's reduction is a powerful tool in power system analysis because it enables the simplification of complex, large-scale networks without sacrificing the accuracy of key system dynamics. By eliminating external nodes and reducing the size of the network, the algorithm achieves the following benefits:

- **Computational Efficiency:** The reduction in network size allows for faster simulations, particularly when performing contingency or scenario analysis.
- **Preservation of Critical Interactions:** Despite the reduction in size, interactions between the internal grid and boundary nodes are preserved, ensuring the reduced model faithfully represents the dynamics of the full network.
- **Focus on Internal Grid Dynamics:** By removing external nodes that minimally impact the internal grid, the reduced model focuses on the core system, facilitating more targeted analysis of its behavior.

Hence, Ward's reduction simplifies the **Full network model** into a manageable size while maintaining key interactions with surrounding regions. This step ensures that the reduced network is computationally efficient, yet still robust enough to model the essential characteristics of the full system.

## 3.5. Updating the Branch Matrix and Electrical Parameters (Stage 4)

After performing the network reduction using Ward's method, it becomes crucial to update the **branch matrix** to reflect the new network structure. This stage ensures that the connectivity between internal and boundary nodes is preserved and that the electrical properties of the network, such as resistance ( $R$ ) and reactance ( $X$ ), are accurately recalculated based on the reduced admittance matrix. The goal is to maintain the physical representation of the network consistent with the reduced system topology, thereby enabling accurate further analysis.

### 3.5.1. Filtering and Retaining Relevant Branches

The first task in this stage is to identify and retain only the branches that connect **internal and boundary nodes**, as the external nodes have been eliminated during the reduction process. The updated branch matrix includes only the branches that contribute to the reduced system, ensuring that the core electrical relationships between the retained nodes are maintained.

### 3.5.2. Recalculating Boundary Connections

In the reduced system, the interactions between boundary nodes become critical since these nodes represent the interface between the internal grid and external regions. To ensure that these interactions are captured accurately, the algorithm checks for any missing or newly required connections between boundary nodes, based on the reduced admittance matrix ( $Y_{bb\_new}$ ). This step ensures that the reduced system continues to accurately represent electrical behavior at the boundaries of the internal grid.

### 3.5.3. Flowchart: Updating the Branch Matrix

Figure 3.5 outlines the process of updating the branch matrix. This flowchart provides a step-by-step overview of filtering the branches, checking for connections between boundary nodes, and recalculating branch properties.

The key steps in this process are:

1. **Start Branch Updating:** Initiate the branch updating process.
2. **Filter Branches Connecting Internal and Boundary Nodes:** Identify and

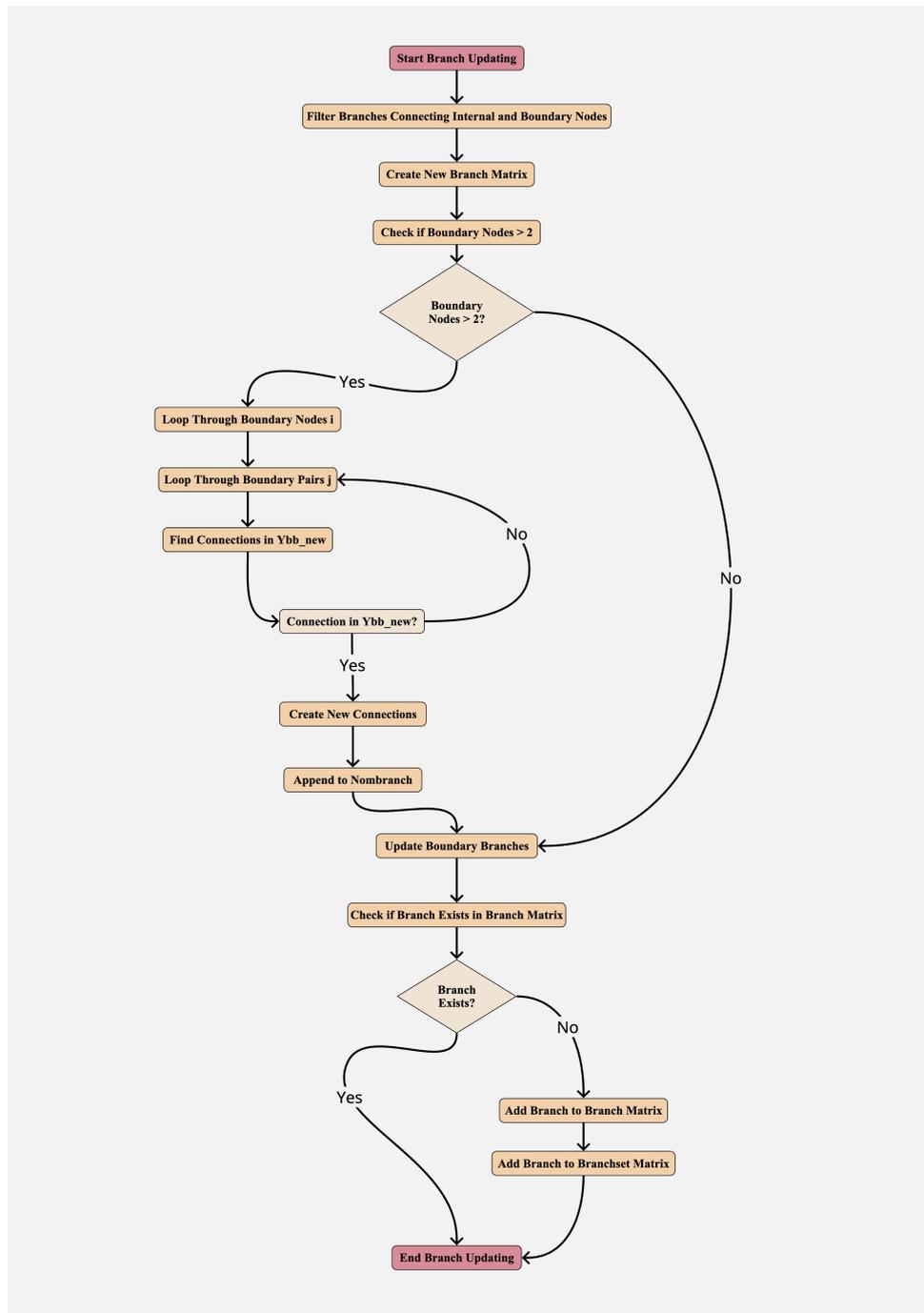


Figure 3.5: Flowchart for Updating the Branch Matrix after Reduction

retain only the branches that connect internal and boundary nodes.

3. **Create New Branch Matrix:** Form the new branch matrix based on the filtered branches.
4. **Check for More Than Two Boundary Nodes:** If more than two boundary nodes are present, check for necessary connections between them.
5. **Loop Through Boundary Nodes and Node Pairs:** Iterate through each pair of boundary nodes to check for connections in the reduced admittance matrix ( $Y_{bb\_new}$ ).
6. **Check for Connections:** Determine whether a connection exists between the boundary nodes in  $Y_{bb\_new}$ .
7. **Create New Connections:** If a connection is found as per  $Y_{bb\_new}$ , create the new connections and update the new branch matrix accordingly.
8. **Update Boundary Branches:** Ensure that the list of boundary branches is updated with any new connections.
9. **Check if Branch Exists in the Branch Matrix:** Verify whether the newly identified branch already exists in the matrix.
10. **Add Missing Branches:** If a branch is missing, add it to the matrix to maintain the correct topology.
11. **End Branch Updating:** Conclude the branch updating process.

#### 3.5.4. Updating the Branch Matrix with New Parameters

Once the boundary branches are recalculated, the algorithm proceeds to update the branch matrix by inserting the newly calculated values of resistance ( $R$ ) and reactance ( $X$ ) for boundary branches. This step ensures that the physical properties of the lines connecting the boundary nodes in the reduced network accurately reflect the electrical characteristics of the original network.

#### 3.5.5. Flowchart: Updating Resistance and Reactance ( $R$ and $X$ )

Figure 3.6 shows the flowchart for recalculating and updating the resistance ( $R$ ) and reactance ( $X$ ) values for the boundary branches in the reduced network. These recalculations

are essential to ensure that the branch matrix reflects the correct electrical properties of the reduced network.

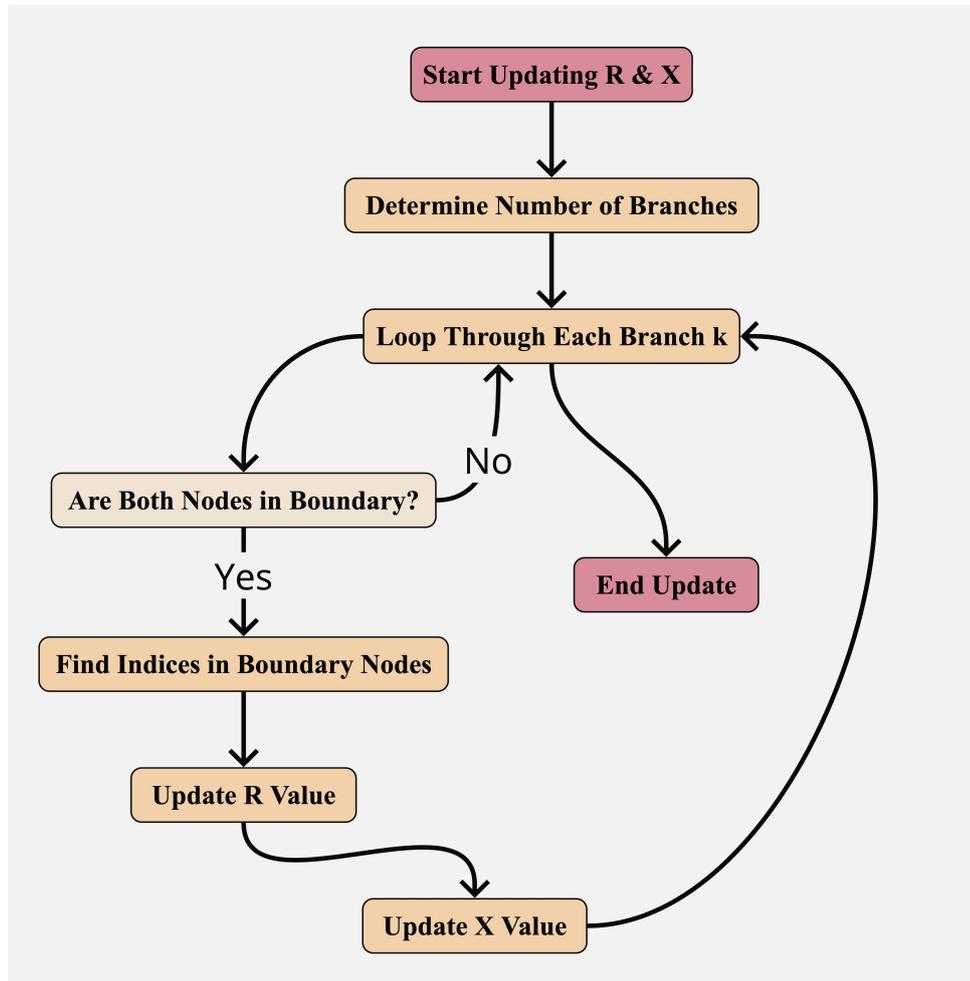


Figure 3.6: Flowchart for Updating R and X values of Boundary Branches after Reduction

This process involves the following steps:

1. **Start Updating R & X:** Begin the process of recalculating the resistance and reactance values.
2. **Determine the Number of Branches:** Identify the total number of branches in the reduced network.
3. **Loop Through Each Branch:** Iterate through each branch to check whether both nodes are boundary nodes.
4. **Check if Both Nodes are Boundary Nodes:** If both nodes of a branch are boundary nodes, continue with updating the branch's electrical parameters.

5. **Find Indices in Boundary Nodes:** Locate the corresponding indices of the boundary nodes in the reduced admittance matrix ( $Y_{bb\_new}$ ).
6. **Update R Value:** Calculate and update the resistance value ( $R_{ij}$ ) between the boundary nodes.
7. **Update X Value:** Calculate and update the reactance value ( $X_{ij}$ ) between the boundary nodes.
8. **End Update:** Conclude the update process after all branches have been processed.

This stage is vital for preserving the **physical integrity** of the reduced network. By updating the branch matrix and recalculating the resistance and reactance values for the remaining branches, the algorithm guarantees that the electrical characteristics of the reduced system remain faithful to the original network. Furthermore, this process ensures that the reduced system is prepared for accurate power flow simulations, stability assessments, and contingency analyses, while also facilitating efficient and reliable analysis of the internal grid and its interactions with external regions.

### 3.6. Recalculating Power Injections, Voltages, and Shunt Admittances (Stage 5)

Following the reduction and the update of the branch matrix, it becomes essential to recalculate the key electrical parameters for the buses in the reduced network. This stage ensures that the system's power flows, voltage profiles, and shunt admittances are consistent with the simplified network structure. These updates guarantee that the **bus matrix** accurately reflects the behavior of the internal and boundary nodes, thereby preserving the integrity of the reduced power system.

#### 3.6.1. Recomputing Real and Reactive Power Demand

The recalculation of real and reactive power demand at each bus is based on the updated voltages and current injections in the reduced system. The real power ( $P_d$ ) represents the actual energy demand, while the reactive power ( $Q_d$ ) manages voltage levels across the network. This step ensures that the power demands are recalculated accurately for the internal and boundary nodes.

### 3.6.2. Updating Power Demand for Internal and Boundary Nodes

After recalculating the power demands, the algorithm differentiates between boundary and internal nodes to assign the appropriate real and reactive power values. For boundary nodes, the algorithm ensures that the power demand reflects the interactions with external regions, while for internal nodes, power demand is adjusted to reflect either internal consumption or generation.

### 3.6.3. Flowchart: Updating Power Demands

The process of updating real and reactive power demands is shown in Figure 3.7. This flowchart visually represents the steps necessary to ensure that the reduced system's power demands are consistent with the updated network structure.

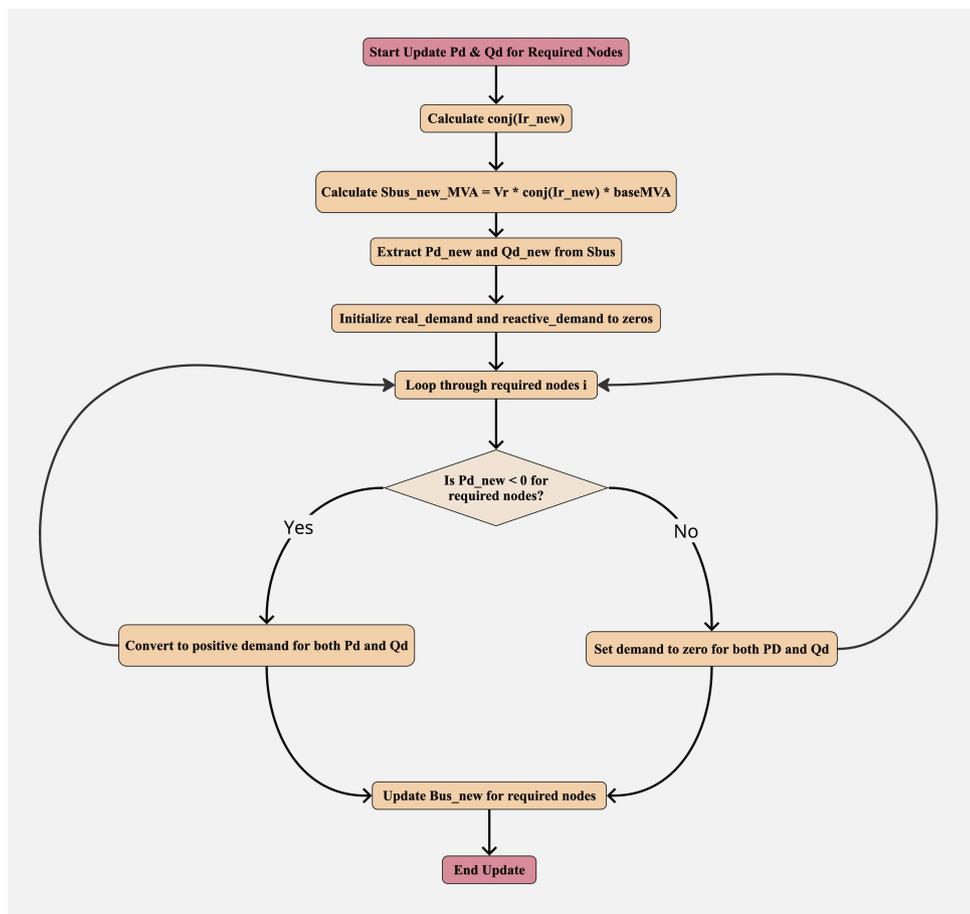


Figure 3.7: Flowchart for Updating Power Demands after Reduction

The process follows these steps:

1. **Start Update Pd & Qd for Required Nodes:** Initialize the process to update

real and reactive power demands.

2. **Calculate  $\text{conj}(I_{r\_new})$ :** Compute the conjugate of the current injection for the required nodes.
3. **Calculate  $S_{bus\_new\_MVA}$ :** Compute the new complex power based on voltage and current values, scaled by the baseMVA.
4. **Extract  $Pd\_new$  and  $Qd\_new$ :** Extract the recalculated real and reactive power demands.
5. **Initialize Real and Reactive Demands to Zeros:** Set initial demand values to zero before updating.
6. **Loop Through Required Nodes:** Iterate through each required node in the reduced network.
7. **Check if  $Pd\_new < 0$ :** For each required node, check if the recalculated real power demand is negative.
8. **Convert to Positive Demand for Both  $Pd$  and  $Qd$ :** If negative, convert the power demand values to positive values.
9. **Set Demand to Zero for Both  $Pd$  and  $Qd$ :** If the demand is non-negative, set the real and reactive power demand to zero.
10. **Update  $Bus\_new$  for Required Nodes:** Update the bus matrix with the recalculated power demand values.
11. **End Update:** Finalize the process of updating power demands.

### 3.6.4. Recalculating and Updating Bus Voltage Profiles

Voltage magnitudes and phase angles are vital for power flow calculations. By recalculating the magnitude and angle of the voltage at each bus, the algorithm ensures that the voltage profile remains consistent with the new network's behavior. These voltage profiles are then updated in the **bus matrix** to reflect the new system state.

### 3.6.5. Calculating and Updating Shunt Admittances

Shunt admittances represent the conductance and susceptance at each bus and are essential for modeling the influence of capacitors and inductors on the system. The algorithm recalculates the shunt admittance values based on the reduced admittance matrix. The

updated shunt conductance ( $G_s$ ) and susceptance ( $B_s$ ) values ensure that the reduced system's reactive power characteristics remain accurate.

For boundary nodes, this recalculation of shunt admittances is particularly important to maintain accurate voltage control at the interface between the internal network and external regions.

Finally, the recalculation of power demands, voltage profiles, and shunt admittances guarantees that the **bus matrix** aligns with the reduced network structure. By performing these updates, the algorithm ensures that the reduced system is consistent with the original network, maintaining accurate representation of the system's key characteristics.

### 3.7. Updating Generator Information (Stage 6)

Generators play a crucial role in ensuring the stability of the power system by supplying both real and reactive power. Following the network reduction process, it is necessary to update the generator data to align with the simplified grid structure. This stage includes:

- **Filtering and retaining relevant generators** that are located at internal and boundary nodes.
- **Recalculating real and reactive power outputs** using the new power flow results from the reduced system.
- **Assigning a slack bus** if one is not already present to maintain system balance and provide a voltage reference point.

The goal is to ensure that the generator matrix in the reduced network accurately reflects the system's generation capabilities and that the slack bus is properly defined for power flow calculations.

#### 3.7.1. Filtering Generators in the Reduced Network

After the reduction process, not all generators remain part of the simplified network. Therefore, the first step involves filtering out generators located at external nodes, retaining only those connected to internal and boundary nodes. This process creates a new generator matrix that is consistent with the reduced network. The retained generators are critical for ensuring that the generation capacity within the internal grid and at its boundaries is preserved.

### 3.7.2. Recalculating Real and Reactive Power Outputs

Once the relevant generators have been identified, the real and reactive power outputs for each generator are recalculated based on the reduced system's power flow results. This recalculation ensures that the generators' output values are accurate and reflect the new operating conditions of the simplified grid. The real power represents the actual energy generated, while the reactive power is responsible for maintaining voltage stability in the system.

The recalculated outputs are then updated in the generator matrix to ensure consistency with the new power flow scenario.

### 3.7.3. Flowchart: Updating Generator Matrix

The process of filtering and updating the generator matrix is shown in Figure 3.8. This flowchart outlines the steps involved in ensuring that the reduced network's generation data remains accurate and functional.

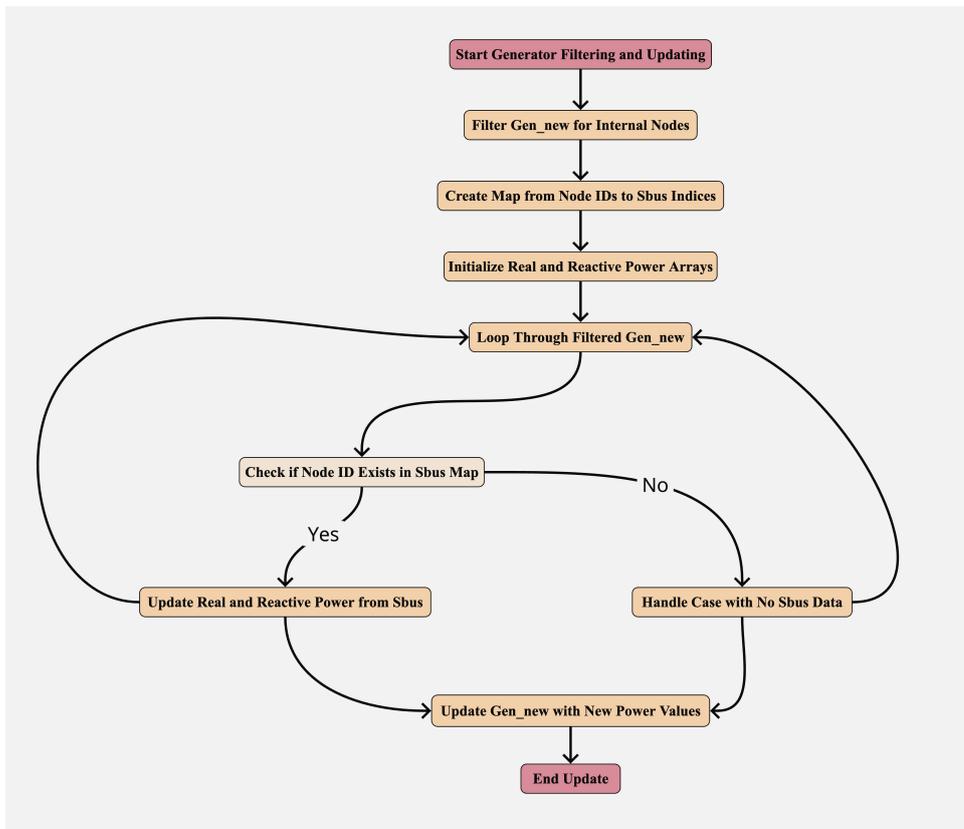


Figure 3.8: Flowchart for Updating Generator Matrix after Reduction

The steps of the flowchart include:

1. **Start Generator Filtering and Updating:** Initiate the process of updating the generator matrix.
2. **Filter Generators for Required Nodes:** Only retain generators connected to internal and boundary nodes.
3. **Create Map from Node IDs to Sbus Indices:** Prepare a mapping between the generator node IDs and the corresponding power flow results.
4. **Initialize Real and Reactive Power Arrays:** Prepare to update the real and reactive power outputs.
5. **Loop Through Filtered Generators:** For each remaining generator, check its corresponding node ID.
6. **Check if Node ID Exists in Sbus Map:** Determine if the generator is part of the power flow calculation.
7. **Update Real and Reactive Power:** If the generator is included, update its output values.
8. **Handle Case with No Sbus Data:** If the generator is not part of the power flow results, handle it accordingly by assigning default values.
9. **End Update:** Complete the process of updating the generator matrix.

#### 3.7.4. Assigning a Slack Bus

A slack bus serves as a reference point for voltage across the power system, absorbing any power imbalances during the power flow process. If, after the reduction, a slack bus is no longer present, it becomes essential to assign a new one. The slack bus is chosen among the generator buses, typically selecting the one with the highest capacity ( $P_{\max}$ ) to act as the new reference.

Assigning a slack bus ensures that the reduced system has a voltage reference point, maintaining power flow consistency and system balance.

#### 3.7.5. Flowchart: Checking for Slack Bus

Figure 3.9 illustrates the process for checking whether a slack bus is present in the reduced system and, if not, how a new slack bus is selected.

The flowchart follows these steps:

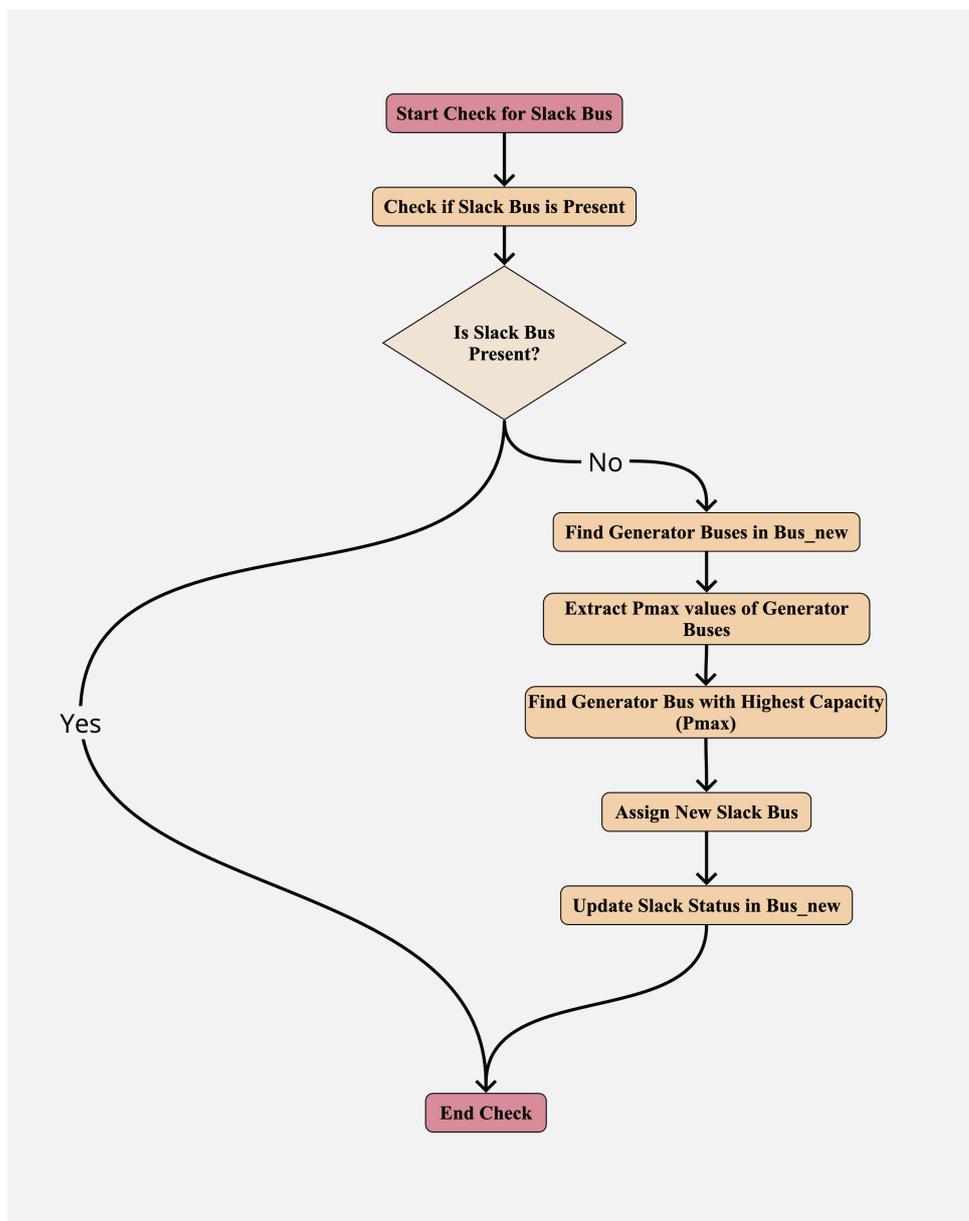


Figure 3.9: Flowchart for Checking the Slack Bus after Reduction

1. **Start Check for Slack Bus:** Begin the process of verifying whether a slack bus is present.
2. **Check if Slack Bus is Present:** Determine if a slack bus exists in the reduced system.
3. **Is Slack Bus Present?** If a slack bus is found, proceed with the current configuration; otherwise, a new slack bus needs to be assigned.
4. **Find Generator Buses in Bus\_new:** Identify all remaining generator buses in the reduced network.
5. **Extract P\_max limits of Generator Buses:** Retrieve the P\_max limits of the generator buses.
6. **Find Generator Bus with Highest P\_max limits:** Select the generator bus with the highest P\_max limits to be the new slack bus.
7. **Assign New Slack Bus:** Designate the selected bus as the new slack bus.
8. **Update Slack Status in Bus\_new:** Update the bus matrix to reflect the new slack bus assignment.
9. **End Check:** Conclude the slack bus assignment process.

Once the generator outputs are recalculated and a slack bus is assigned, the generator matrix and bus matrix are updated accordingly. These updates ensure that the reduced network remains consistent and balanced, allowing for accurate power flow simulations, stability assessments, and contingency analyses. By focusing on internal and boundary nodes and properly managing the slack bus, the algorithm guarantees that the reduced network continues to function correctly, maintaining the essential characteristics of the full system.

### 3.8. Finalizing the Reduced Network Data (Stage 7)

After updating the bus, branch, and generator matrices, the algorithm completes the reduction process by ensuring that all additional network data is aligned with the simplified network structure. This step focuses on finalizing key components such as phase shifters, network constraints, generator bids, and node information. The process guarantees that the reduced network is comprehensive and ready for further analysis or simulations, maintaining the critical operational characteristics of the original system.

### 3.8.1. Filtering Phase Shifters and Network Constraints

Phase shifters, which adjust the phase angle difference between connected nodes, play an important role in controlling power flows. The algorithm ensures that only phase shifters relevant to the reduced network (those connecting internal and boundary nodes) are retained. This filtering ensures that the reduced network retains accurate phase shifter data, which is critical for controlling power flows between internal and boundary nodes.

Additionally, the algorithm updates network constraints such as **N-1 constraints** (which ensure the system remains stable even after the loss of a single component) and unconstrained branches. By retaining relevant constraints in the reduced network, the algorithm ensures that critical operational limits are respected, enabling accurate and reliable analysis of the reduced system.

### 3.8.2. Filtering and Updating Generator Bid Data

The generator bid data, which reflects the cost and operational characteristics of generators, is filtered to include only the generators that remain in the reduced network. This ensures that the **generator bid matrix** accurately reflects the reduced generator set, allowing the system to retain the relevant cost and operational data for future economic dispatch or market analyses.

### 3.8.3. Updating Node and Branch Information

To maintain consistency in the reduced system, key matrices such as **nomnod** (node labels) and **branchset** (branch data) are updated to reflect the structure of the simplified network. By updating node information and branch data, the algorithm ensures that the reduced network maintains the correct topology and operational details, aligning with the original system's structure.

### 3.8.4. Ensuring Consistency in Labels and Other Operational Data

The algorithm also updates additional operational data such as **node information** (**nodInfo**) and **branch labels** (**nombranch**) to ensure consistency across the reduced network. This ensures that node identifiers and branch labels remain consistent, providing a seamless connection between the reduced and full networks. By updating these elements, the algorithm guarantees that all network components are properly aligned, allowing for coherent and accurate simulations.

### 3.8.5. Flowchart: Finalizing the Reduced Network

Figure 3.10 outlines the process of finalizing the reduced network. This includes steps for filtering relevant data, updating branch and node information, and ensuring consistency in all operational parameters.

The key steps involved in finalizing the reduced network are:

1. **Start Modifications for Reduced Network:** Initialize the process of finalizing the reduced network data.
2. **Filter Branches Between Required Nodes:** Filter branches to retain only those between required nodes.
3. **Initialize New Phase Shifters Matrix:** Set up a new matrix to store phase shifters in the reduced network.
4. **Check for External Branch without Phase Shifter:** Exclude any external branches that do not have phase shifters.
5. **Copy Phase Shifter Data:** For the remaining branches, copy the relevant phase shifter data.
6. **Update Constrained Status for Matching Branches:** Update the network constraints for branches that are still present in the reduced network.
7. **Filter GAMS Generation and Supply Bid Data:** Ensure that only the generation and supply bid data relevant to the reduced network is retained.
8. **Update Nominal Node Data:** Update node labels and identifiers to reflect the reduced system.
9. **Loop Through Branchset to Update Branchset\_new:** Update the branchset matrix to align with the reduced system.
10. **End Modifications:** Conclude the process once all modifications have been completed.

### 3.8.6. Ensuring Complete Finalization of the Reduced Network

Finalizing the reduced network data ensures that all components—such as phase shifters, constraints, generator bids, node information, and branch labels—are accurately updated to reflect the reduced network structure. This comprehensive approach guarantees that the reduced network is not only smaller but also retains all critical operational details from

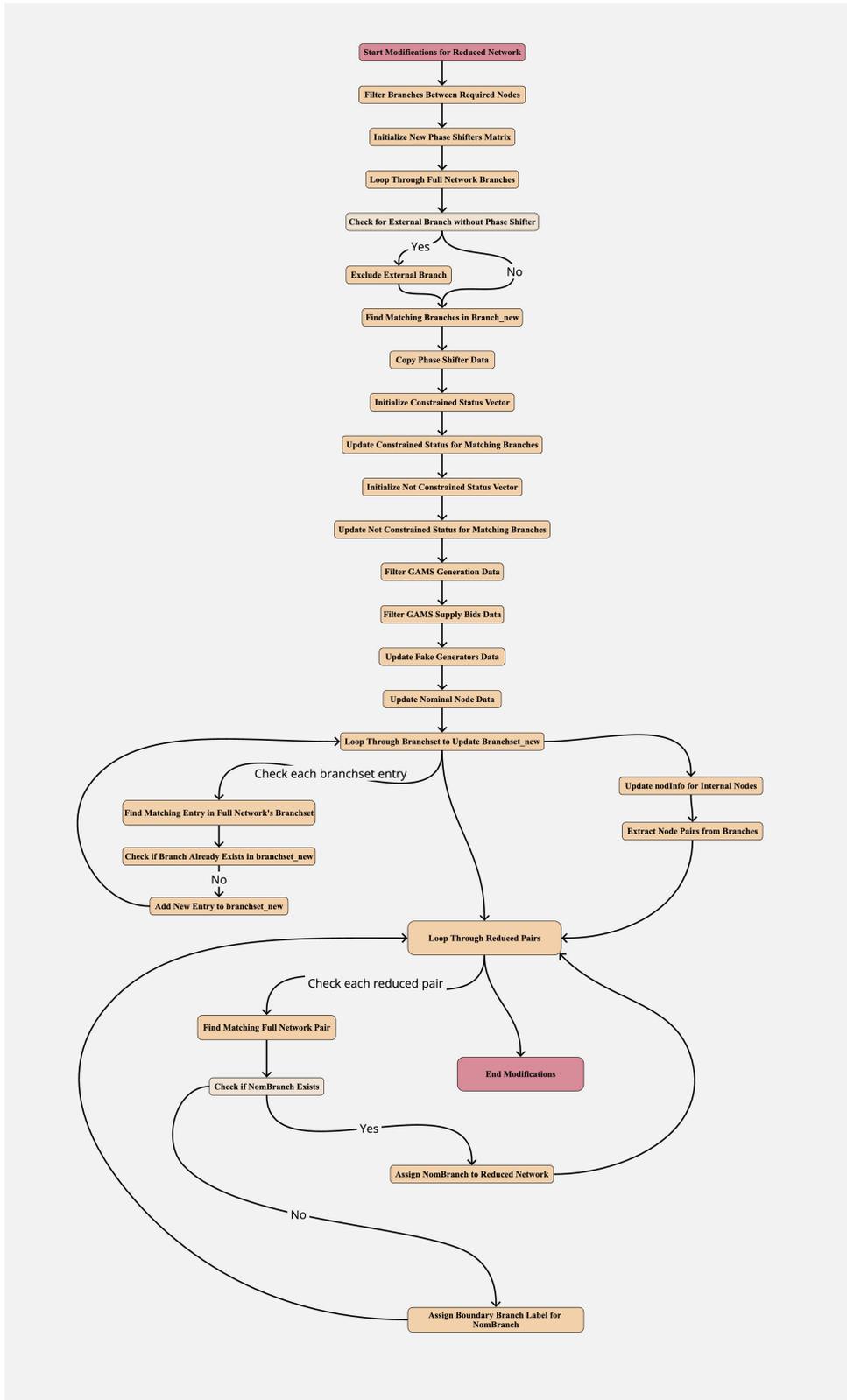


Figure 3.10: Flowchart for Remaining Modifications of the Reduced Network

the original system. By finalizing the data in this manner, the algorithm ensures that the reduced model is fully prepared for future power flow analyses, market simulations, or stability assessments.

### 3.9. Renumbering and Saving the Reduced Network (Stage 8)

The final stage of the algorithm focuses on renumbering the buses and branches of the reduced network and saving the network data in a consistent and manageable format. This step involves creating a sequential numbering system for buses and branches to simplify the network representation, ensuring that the reduced model is well-structured for future simulations or analyses. Additionally, the original-to-new bus number mappings are saved, providing a reference for maintaining traceability between the original and reduced systems.

#### 3.9.1. Renumbering Buses for Consistency and Simplification

To streamline the representation of the reduced network, the original bus numbers are replaced with a **sequential numbering system**. This simplifies the handling of the network during simulations and makes the reduced model easier to interpret. The algorithm generates a mapping between the **original bus numbers** and the **new sequential numbers**, which are easier to handle in future analyses. This mapping also ensures that the integrity of the system is maintained by keeping a reference to the original numbering scheme.

#### 3.9.2. Renumbering Branches for Compatibility with New Bus Numbers

Similarly, the branches are renumbered to reflect the new bus numbering system. This step ensures that all **branch connections** remain consistent with the renumbered buses. By updating the "From" and "To" bus numbers for each branch, the reduced network maintains its proper connectivity while aligning with the new bus numbering system. This ensures consistency in the system's topology, allowing for accurate future simulations.

### 3.9.3. Renumbering Generators and Branchset

The generators are also renumbered to align with the updated bus matrix. This step ensures that the generator data remains consistent with the updated bus numbers, preserving the correct generator locations in the reduced network. Additionally, the **branchset matrix**, which tracks active branches, is renumbered to reflect the new bus numbering. This ensures that the active branches are correctly updated to align with the renumbered buses, maintaining the integrity of the branch connections in the reduced network.

### 3.9.4. Saving the Final Reduced Network

Once the renumbering is complete, the reduced network data is saved into a series of output files for future use. This step saves the renumbered bus, branch, generator, and related network data into a final output file. This file contains all the necessary information for conducting simulations on the reduced network.

### 3.9.5. Flowchart: Renumbering and Mapping the Reduced Network

Figure 3.11 outlines the renumbering and mapping process for the reduced network. This flowchart highlights the steps necessary to renumber the buses, branches, and generator matrices, and to save the final reduced network.

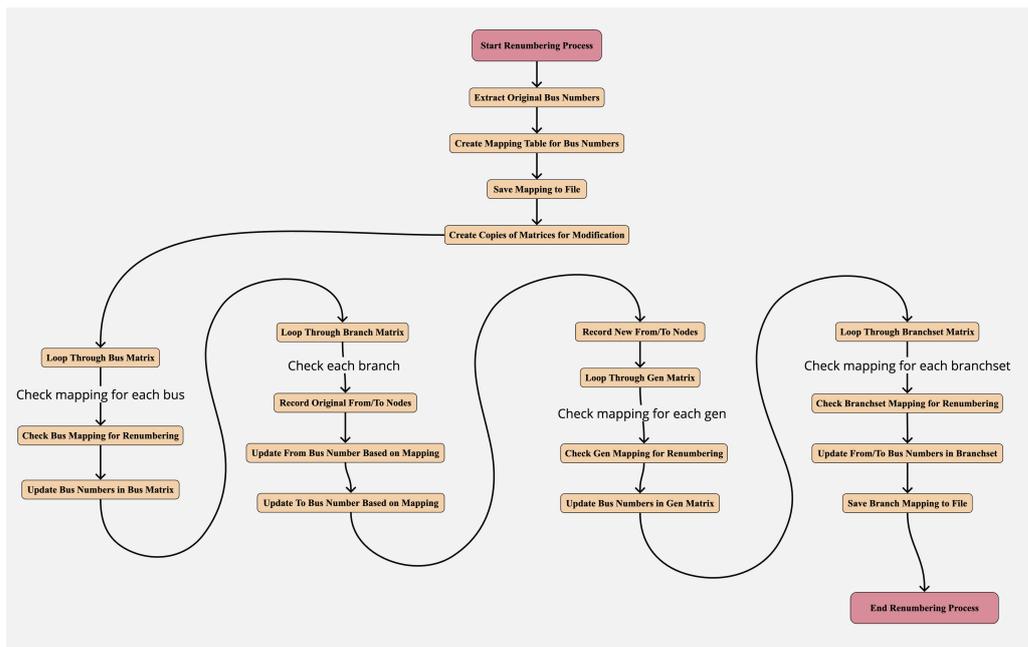


Figure 3.11: Flowchart for Renumbering and Mapping the Reduced Network

The key steps in this process are:

1. **Start Renumbering Process:** Begin the process of renumbering buses and branches.
2. **Extract Original Bus Numbers:** Extract the original bus numbers from the bus matrix.
3. **Create Mapping Table for Bus Numbers:** Generate a mapping table to link the original bus numbers to the new sequential numbers.
4. **Update Bus Numbers:** Apply the new numbering to the bus matrix and update the "From" and "To" bus numbers in the branch matrix.
5. **Update Generators and Branchset:** Renumber the generator matrix and branchset matrix to maintain consistency with the updated bus numbers.
6. **Save Final Reduced Network:** Save the renumbered bus, branch, generator, and related matrices to a final output file.
7. **Save Bus and Branch Mappings:** Save the mappings between original and renumbered buses and branches to a file for future reference.
8. **End Renumbering Process:** Conclude the renumbering process.

### 3.9.6. Ensuring Consistency and Usability of the Reduced Network

Renumbering the buses and branches is crucial for simplifying the network structure and ensuring consistency across all components. The sequential numbering system not only makes the reduced network easier to manage but also improves computational efficiency for large-scale simulations. Saving the reduced network data and maintaining mappings between original and renumbered buses ensures that the essential characteristics of the original system are preserved, while the reduced model remains comprehensible and ready for future analysis.

## 3.10. Investigating the Efficacy of the Reduction and Future Applications

The algorithm presented in this chapter (MATLAB codes are shown in Appendix A) delivers a highly effective and systematic approach to **network reduction**, making it particularly well-suited for studying future scenarios related to the 2030 energy transi-

tion. By utilizing techniques such as **Ward's reduction** and incorporating thorough updates to network matrices, this methodology ensures that the reduced network preserves key electrical characteristics while significantly reducing the system's size. This reduction provides a computationally efficient model that retains the internal and boundary node dynamics of the Reduced Network, allowing for in-depth analysis of various future operational scenarios.

The robustness of the algorithm lies in its ability to balance **model simplification** with **accuracy**. Through careful filtering, renumbering, and recalculation of power flows, generator outputs, and branch data, the algorithm ensures that the reduced model remains an accurate representation of the original, larger system. This enables it to handle detailed studies such as **contingency analysis**, **stability assessments**, and simulations aimed at understanding the evolving challenges posed by the energy transition, while maintaining a level of computational efficiency necessary for large-scale simulations.

### 3.10.1. Investigating the Efficacy of the Reduction

While the algorithm achieves a substantial reduction in network complexity, the next step is to rigorously evaluate its **efficacy and accuracy**. This can be done by comparing the results of key analyses, such as power flows and voltage profiles, between the full network and the reduced model. Metrics such as the deviation in **bus voltages**, **branch power flows**, and **reactive power injections** will be critical in determining how well the reduced model preserves the operational dynamics of the original network.

In particular, evaluating the reduced model's performance under normal and **contingency scenarios** will reveal how well it maintains system stability under various fault conditions. Furthermore, conducting **stability assessments** on the reduced grid will help assess its responsiveness to potential fluctuations in power generation and demand, which are expected to increase in variability as the grid integrates more renewable energy sources by 2030.

By systematically testing and validating the accuracy of the reduced model against the full system, the efficacy of the network reduction can be confirmed. This investigation will also provide insights into potential adjustments needed to refine the reduction process, ensuring that the simplified model remains a reliable tool for strategic planning and scenario analysis in the context of the National grid's future energy challenges.

### 3.10.2. Future Applications of the Reduced Model

The reduced model created through this algorithm can serve as the foundation for various future applications, including:

- **Long-term Planning:** The simplified model can be used to conduct future grid studies that account for the integration of renewables, electric vehicles, and other emerging technologies in the Reduced grid.
- **Real-time Simulations:** The model can be deployed for real-time simulations that monitor grid stability and test corrective actions in response to potential faults or outages.
- **Market and Economic Analysis:** With accurate generation data and operational constraints, the reduced model can support market simulations, helping stakeholders evaluate generation bids, economic dispatch strategies, and price formation in future electricity markets.

The combination of these future applications with the reduced model enables grid operators and planners to address the complex challenges posed by the energy transition while maintaining a simplified, yet accurate, representation of the Full Network. The ability to efficiently simulate and analyze scenarios in the reduced network will ultimately support informed decision-making, ensuring grid reliability and resilience as the energy landscape evolves.



# 4 | Observation on the Reduced Network and Ward's Reduction Accuracy Analysis

The Ward reduction method is a powerful tool for simplifying large-scale power systems by reducing the external part of the network while maintaining the internal system's key characteristics. By replacing the external network with an equivalent model, the method significantly reduces computational complexity while still providing an accurate representation of the internal system. In this chapter, the behavior of the reduced network is analyzed and compared with the full network to assess the accuracy and efficacy of the Ward reduction.

The focus of this analysis is twofold:

1. To evaluate how the reduction impacts key electrical parameters such as voltage, current, and power at boundary and internal buses.
2. To assess the behavior of the reduced network during common contingency scenarios.

The outcomes of this analysis provide a basis for understanding the strengths and potential limitations of Ward's reduction in power system studies.

## 4.1. A Comparative Assessment of Power Flow Results between Full and Reduced Networks under Normal Conditions.

Buses serve as critical junctions within power systems, facilitating electrical energy flow, maintaining voltage stability, and regulating interactions among various network components. In the context of Ward's network reduction, boundary buses, which interconnect the internal and external segments of a grid, exhibit the most pronounced impacts. These buses reflect the alterations imposed by the modified external system, making them piv-

otal in understanding how well the reduction process preserves system behavior.

#### 4.1.1. Understanding Boundary and Internal Buses

Boundary buses, located at the interface between the internal and external systems, are most affected by the reduction process. In this case, bus number 11 and 12 are serving as the boundary buses for the reduction process. The internal buses, by contrast, remain largely unaffected due to their insulation from the reduced external model. As a result, we expect Bus Current magnitude changes at boundary buses, whereas internal buses should maintain their original values. Moreover, Voltage magnitude and phase angle should be remain almost unaltered.

#### 4.1.2. Voltage Magnitude and Phase Angle Differences

To check the efficacy of the reduced network under normal condition, the Power flow analysis of both Reduced and Full networks are carried out and voltage differences are evaluated. The **Voltage Magnitude Differences** graph (top panel) and the **Voltage Phase Differences** graph (middle panel) in Figure 4.1 reveals very minimal deviations across few buses and no deviations for most of the buses. The minimal rise in magnitude differences (reaching approximately  $2.5 \times 10^{-16}$  pu) and phase differences (reaching approximately  $2 \times 10^{-15}$  pu) in few buses indicates that the Ward reduction had minimal impact on Bus voltages, which implies the accuracy of the Reduction process.

#### 4.1.3. Bus Current Differences

The **Current Differences** graph (bottom panel) in Figure 4.1 displays significant current discrepancies at bus 11 (peaking around 22 pu) and bus 12 as expected. This underscores the fact that current injections at boundary buses are particularly susceptible to changes in the external system's representation. The large shifts in current indicate a redistribution of power flows, which can be attributed to the altered impedance from the external network. In contrast, internal buses demonstrate minimal to no current changes, reinforcing the effectiveness of Ward's reduction in preserving the internal network's integrity.

Figure 4.1 shows the comparison of power flow results under normal conditions for the full and reduced networks.

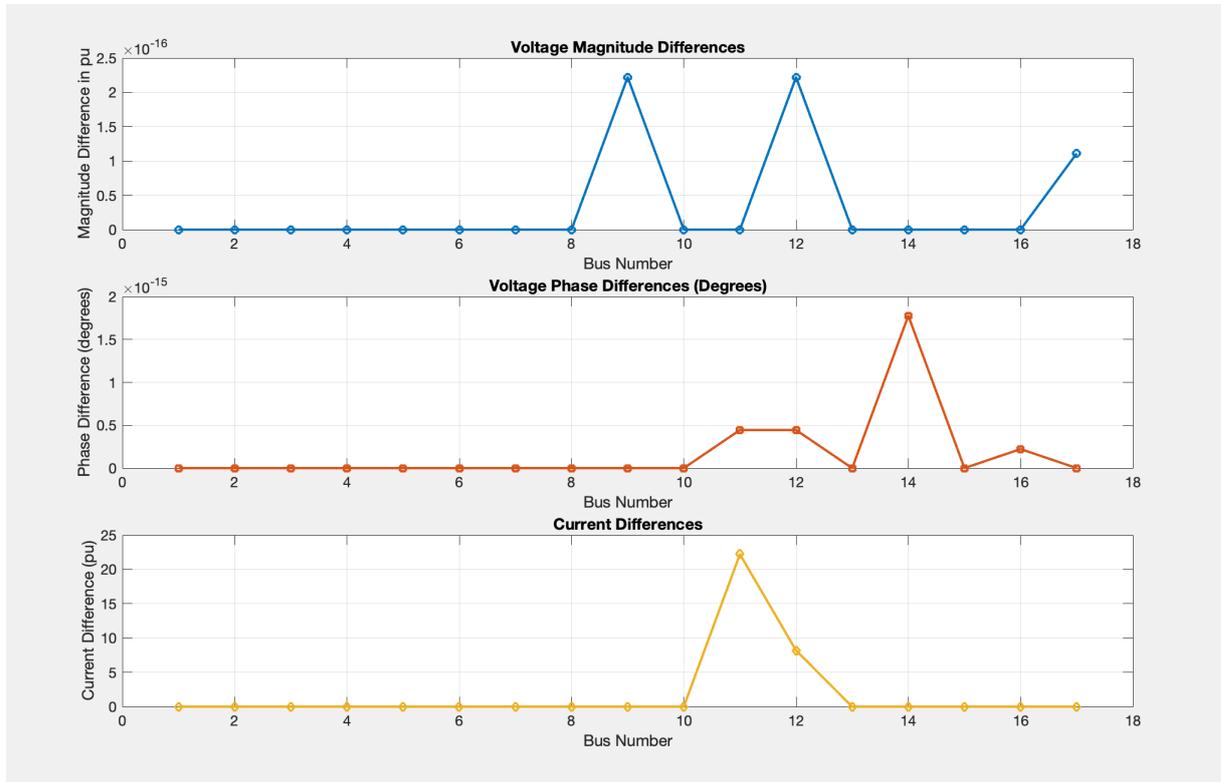


Figure 4.1: Comparison of Power Flow Results between Full and Reduced Networks under Normal Conditions.

#### 4.1.4. Interpretation of Results

The pronounced deviations in bus current at boundary buses 11 and 12 align with expectations: these buses, being the direct points of interaction with the simplified external system, exhibit the most significant impacts from the Ward reduction. The minimal deviations at internal buses provide a reassuring confirmation that the reduction process has preserved the essential characteristics of the internal grid, ensuring that its behavior remains stable and consistent with the full network model.

By quantifying these deviations in terms of voltage, phase angle, and current, this analysis provides critical insights into the strengths and limitations of Ward's reduction for power system analysis. The results confirm that, while boundary buses are inevitably impacted, the internal network's behavior remains intact, ensuring that the reduced network serves as a reliable proxy for the full system under normal operating conditions.

## 4.2. Contingency Analysis of the Reduced Network

The next step in evaluating the efficacy of Ward's reduction is to observe how the reduced network behaves during common contingency scenarios, such as line outages, generator tripping, or bus faults. Contingency analysis is essential in power system studies to ensure the network's stability and reliability under unexpected conditions. By comparing the behavior of the full and reduced networks under identical contingencies, the accuracy of the reduction can be assessed.

A set of common contingencies affecting both the full and reduced networks must be selected for comparison. These include:

- **Single-line outages:** Where transmission lines are disconnected due to faults or maintenance.
- **Generator tripping:** The sudden loss of a generator.
- **Bus faults:** Short-circuit or voltage violations at specific buses.

In the first phase of the analysis, contingency scenarios are simulated on the full network. The resulting changes in bus voltages, power flows, and current injections are recorded as a baseline for comparison with the reduced network. These parameters are particularly important at the boundary buses, where the external network's influence is most prominent.

Afterward, the required contingency scenarios are simulated on the reduced network. Since Ward's reduction simplifies the external network, the internal system's response should ideally remain consistent between the full and reduced networks. Key parameters, such as bus voltages and power flows, are monitored and compared to the results from the full network.

A direct comparison is made between the full and reduced networks to evaluate any deviations in system behavior during contingencies. The focus is placed on:

- **Voltage Profiles:** The voltage magnitudes and angles at key buses, particularly buses near to the boundary buses, should remain consistent between the full and reduced networks. Significant deviations suggest a loss of accuracy in the reduced network.
- **Power Flow Deviations:** Power flow patterns should be similar in both networks. Differences in power flow distribution indicate that the reduced network may not fully capture the external network's influence.

- **Current Injections:** Current injections at boundary buses must be compared, as these represent the interaction between the internal system and the reduced external system.

If the reduced network exhibits minimal deviations in bus voltages, power flows, and current injections compared to the full network, it can be concluded that Ward's reduction maintains the internal system's integrity during contingencies. The reduced network accurately represents the external system's influence, ensuring that the internal system behaves similarly to the full network under fault conditions.

However, if significant deviations are observed, this would imply that key aspects of the external system's behavior are not fully captured by the reduced model, necessitating further refinement of the reduction process.



# 5 | N-1 Contingency Simulation and Power Flow Results for Full and Reduced Networks

This chapter presents the methodology and results for performing **N-1 contingency analysis** on both the **full network** and the **reduced network**. The analysis examines the impact of branch failures on power flow, voltage, and overall system behavior. The full network represents the original system, while the reduced network is a simplified version that retains critical elements for computational efficiency.

The N-1 contingency analysis involves simulating the failure of each branch (one at a time) and analyzing the system's response to these failures, including power redistribution, voltage fluctuations, and potential overloading of other branches. Common contingency scenarios are identified for both networks, ensuring a consistent comparison of results. After the identification of these common contingencies, the simulation processes are adapted for the full and reduced networks, highlighting the differences in their respective behaviors.

## 5.1. Methodology for N-1 Contingency Simulation

The process for performing the N-1 contingency analysis involves several key steps, applicable to both the full and reduced networks:

1. **Identifying non-radial branches** to be considered for contingency simulations.
2. **Simulating branch failure scenarios**, where one branch is removed at a time from the network, and the system's ability to maintain power flow is analyzed.
3. **Conducting power flow analysis** after each branch failure to observe changes in voltage, line current, and power injection at each bus.
4. **Saving the simulation results** for further comparison between the full and reduced networks.

While the general process is consistent, adjustments are made in how branch removal and power flow calculations are handled due to the structural differences between the full and reduced networks. These differences are highlighted in the following sections.

## 5.2. Graph Representation and Branch Removal

In both the full and reduced networks, the system is represented as a graph where nodes correspond to buses and edges to transmission lines (branches). The process of removing branches is handled similarly, but the network size and structure impact the results.

### 5.2.1. Network Representation

The **full network** is the original, unaltered system that includes all nodes and branches. The **reduced network** is a simplified version of the full network, retaining only critical elements to reduce computational complexity. Despite this reduction, the process for identifying non-radial branches remains similar. Each branch is evaluated to determine whether it is critical for connectivity:

- **Radial branches:** Removing these branches would disconnect parts of the network, making them critical for network integrity.
- **Non-radial branches:** These branches do not disconnect the network when removed but may still impact power flow and system stability.

### 5.2.2. Graphical Representation of Radial and Non-Radial Branches

The graphical representation of the full and reduced networks with radial and non-radial branches highlighted is shown in the Figure 5.1 and Figure 5.2 below. In the full network, the branches and connections between buses are shown in both radial and non-radial formats, with the radial branches indicated in red and non-radial branches in black. The reduced network represents a smaller version of the original, focusing on critical nodes and connections while retaining the same branch classification.

As highlighted in the figures, the process for identifying radial and non-radial branches is consistent between the full and reduced networks. The reduced network shows renumbered nodes, which were mapped during the network reduction process to simplify calculations. For accurate comparisons, this mapping helps retrieve the original bus numbers.

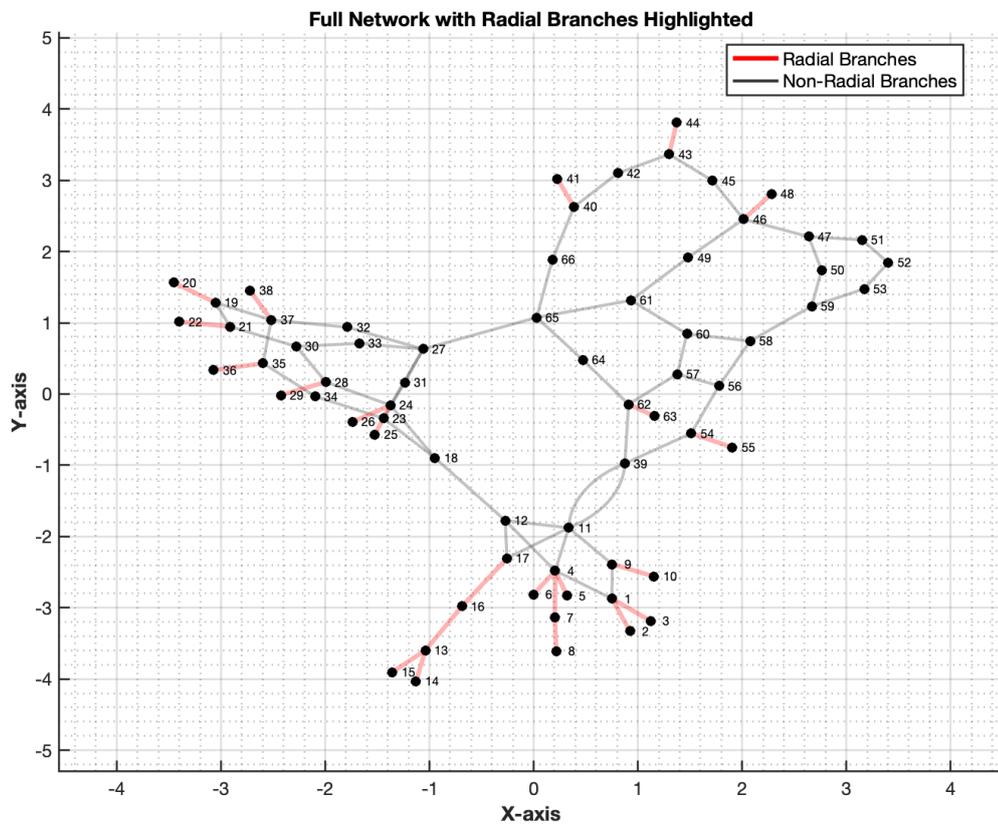


Figure 5.1: Full Network with Radial Branches Highlighted. Radial branches are shown in red, while non-radial branches are shown in black.

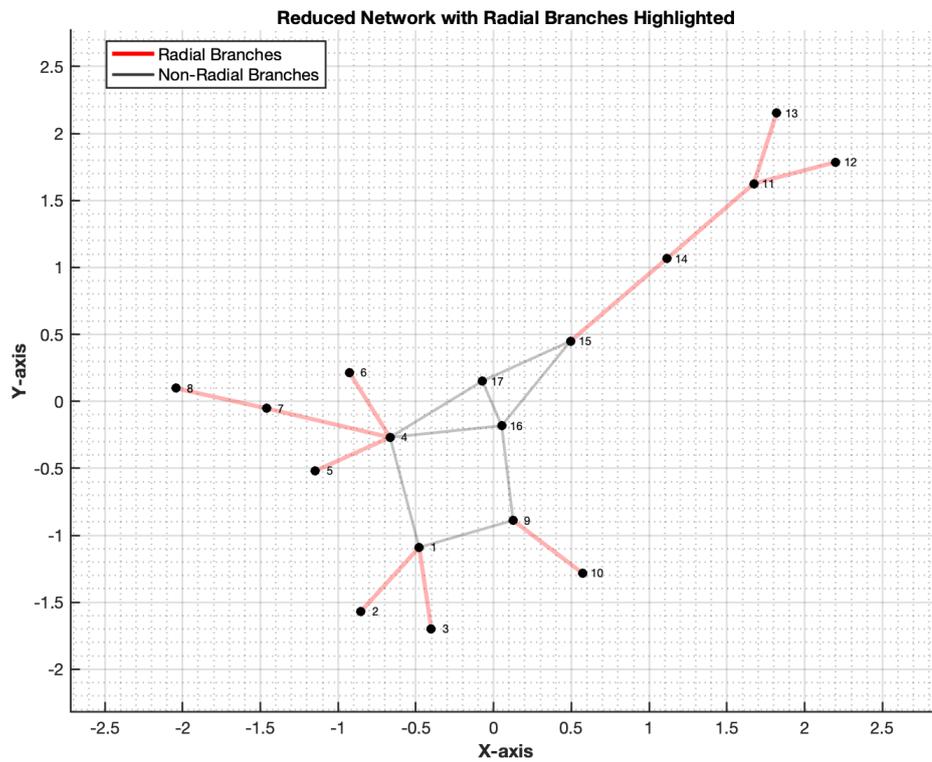


Figure 5.2: Reduced Network with Radial Branches Highlighted. Radial branches are shown in red, while non-radial branches are shown in black. The bus numbers in the reduced network have been renumbered during the network reduction process, but a mapping exists to recover the original bus numbers for comparison with the full network.

### 5.2.3. Flowchart Description: Radial and Non-Radial Branch Detection

The process of identifying and classifying radial and non-radial branches involves several key steps. Figure 5.3 provides a flowchart representation of this process:

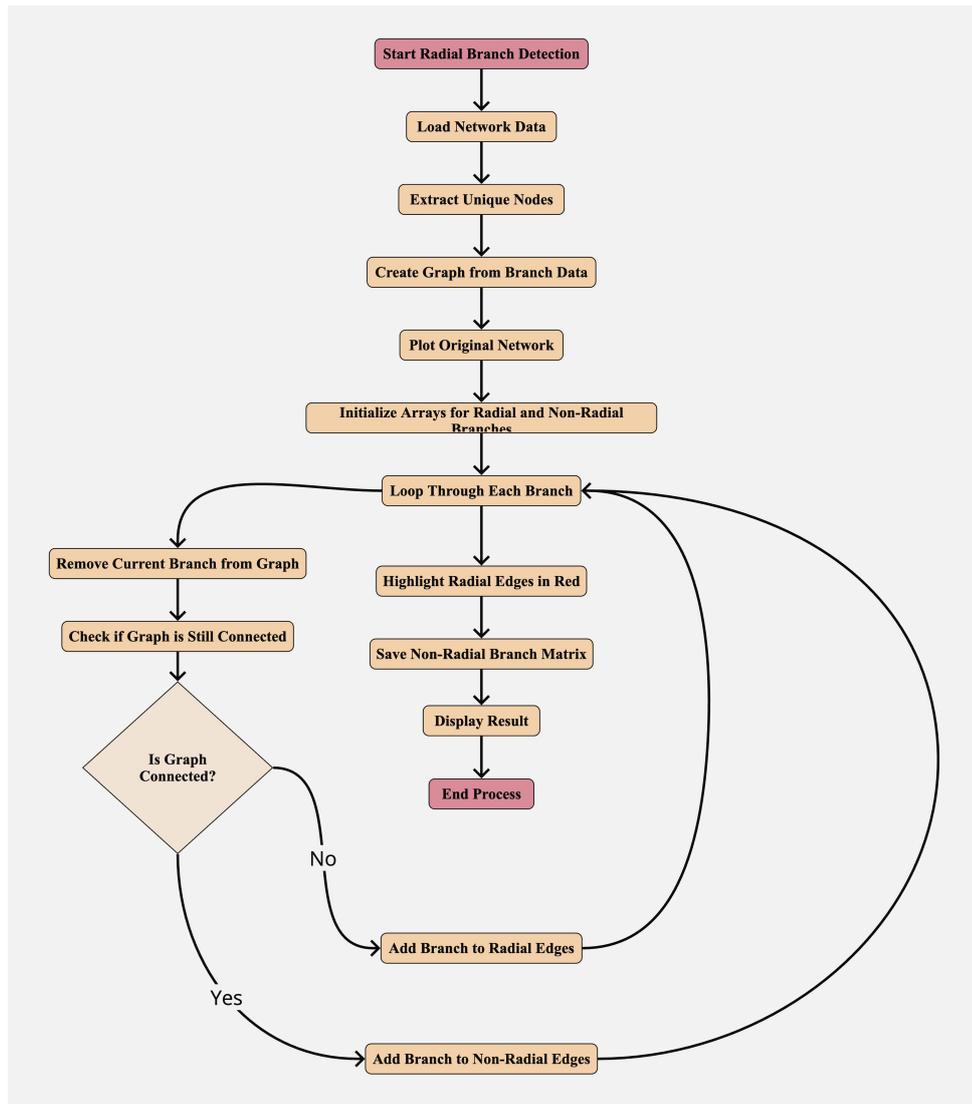


Figure 5.3: Flowchart for Non-Radial Branch Detection

The major steps in detecting non-radial branches are as follows:

1. **Start Radial Branch Detection:** The process begins by initializing the branch detection mechanism.
2. **Load Network Data:** The network data containing node and branch information is loaded to initialize the graph representation.

3. **Extract Unique Nodes:** Extract the unique nodes from the network data, which form the vertices of the graph.
4. **Create Graph from Branch Data:** Using the branch data, a graph is constructed with nodes representing buses and edges representing branches.
5. **Plot Original Network:** The original network is plotted for visualization, with the nodes and branches clearly represented.
6. **Initialize Arrays for Radial and Non-Radial Branches:** Two separate arrays are created to store the radial and non-radial branches identified during the process.
7. **Loop Through Each Branch:** The algorithm iterates through each branch to evaluate its importance for maintaining network connectivity.
8. **Remove Current Branch from Graph:** The current branch being evaluated is temporarily removed from the graph.
9. **Check if the Graph is Still Connected:** After removing the branch, the algorithm checks whether the network remains connected. If the removal of the branch disconnects the network, it is classified as a radial branch.
10. **Add Branch to Radial or Non-Radial Edges:** Based on the outcome of the connectivity check, the branch is added to either the radial or non-radial branch arrays.
11. **Highlight Radial Edges in Red:** For visualization purposes, radial branches are highlighted in red, emphasizing their critical role in maintaining network connectivity.
12. **Save Non-Radial Branch Matrix:** The matrix containing the non-radial branches is saved for further analysis.
13. **Display Results:** The final result, including the classification of branches as radial or non-radial, is displayed.
14. **End Process:** The process concludes with the classification complete, and the radial and non-radial branches identified for future use.

The above flowchart represents a systematic approach to identifying and classifying radial and non-radial branches in both full and reduced networks, providing clarity on the structural importance of each branch.

### 5.3. Contingency Mapping Before N-1 Contingencies Simulation

Before performing N-1 contingency simulations, the system goes through a process of mapping out the branches that will be subjected to these simulations. This step is essential to ensure that only non-radial branches are tested, as their removal affects power flow but doesn't disconnect the network.

The process, illustrated in Figure 5.4, follows these steps:

1. **Initialize Contingency Mapping:** A matrix is created to store the results of each contingency scenario.
2. **Loop Through Non-Radial Branches:** The algorithm iterates over each non-radial branch in the network.
3. **Remove Current Branch:** The selected branch is temporarily removed from the network to simulate a contingency.
4. **Perform Power Flow Analysis:** The system's response is evaluated through a power flow analysis, recalculating key metrics like bus voltages and line currents.
5. **Update and Save Contingency Mapping:** The contingency results are recorded and saved for further analysis.

The flowchart in Figure 5.4 provides a visual representation of this process:

This process ensures that each non-radial branch is systematically tested, and its impact on the network is captured and saved for further comparison between the full and reduced networks.

### 5.4. Power Flow Analysis for Contingencies

In both the full and reduced networks, N-1 contingency simulations were conducted to assess the impact of branch failures on system stability. However, the contingencies differed between the two networks due to their structural differences. The following process, illustrated in Figure 5.5, was applied to each scenario.

The key steps in the power flow analysis were:

1. **Perform Power Flow Analysis:** Recalculate bus voltages and currents for each contingency.

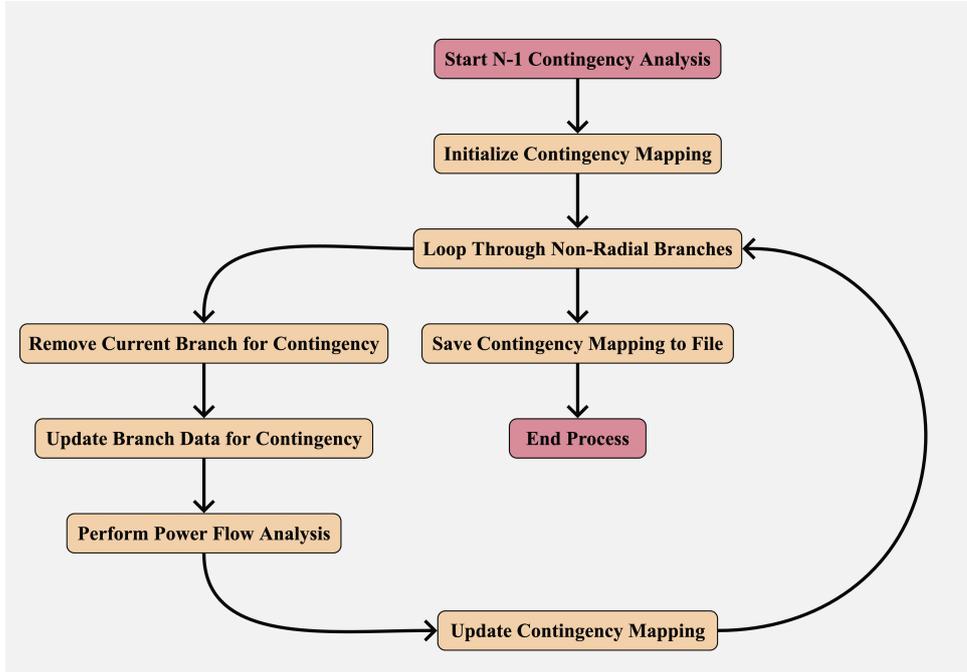


Figure 5.4: Flowchart for Contingency Mapping Process

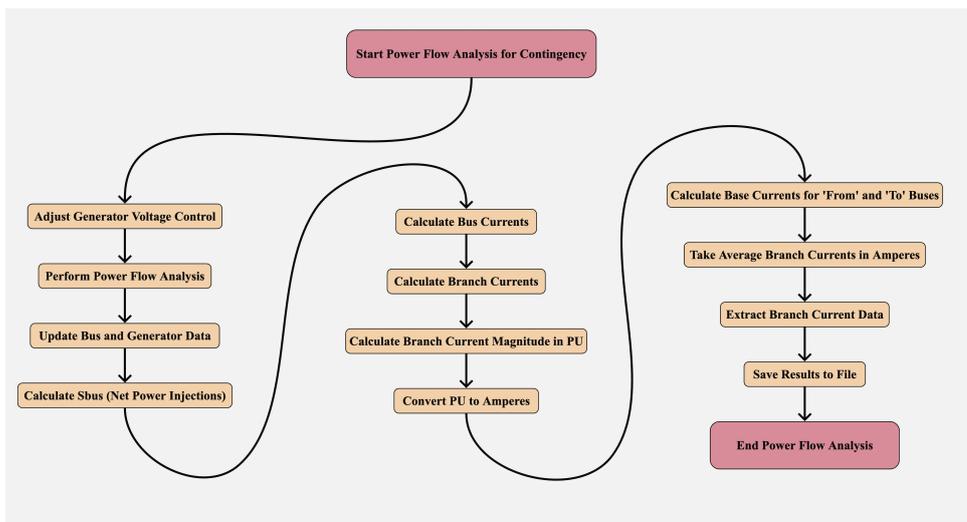


Figure 5.5: Flowchart for Power Flow Analysis during Contingency Simulation

2. **Update Bus and Generator Data:** Reflect changes in the network after each branch removal.
3. **Calculate Net Power Injections:** Compute  $S_{\text{bus}}$ , representing net power injections at each bus.
4. **Calculate and Convert Branch Currents:** Calculate branch currents in per unit (PU) and convert them to amperes for easier interpretation.
5. **Save Results:** Store all key metrics such as bus voltages, power injections, and current magnitudes for further comparison.

### 5.4.1. Full Network Simulation

For the full network, each non-radial branch was removed one by one, and power flow analysis was performed to assess the impact of these failures on the system. Key outputs included bus voltage profiles, power injections, and line currents. This process ensures the full system's response to contingencies is captured.

### 5.4.2. Reduced Network Simulation

The reduced network followed a similar power flow analysis process, but with a distinct set of contingencies. As the reduced network simplifies certain elements, the impact of branch removals differs. This allows the reduced network to maintain computational efficiency while preserving critical behaviors.

## 5.5. Recovering Original Indices and Updating Contingency Results for the Reduced Network

During the network reduction process, the reduced network was renumbered to simplify computations. To enable a meaningful comparison between the full and reduced networks, it is necessary to recover the original bus numbers using a predefined mapping. This step ensures the identification of common contingencies between the two networks and allows for updating the saved results of the reduced network with the original bus numbers.

The process consists of two main stages, as illustrated in Figures 5.6 and 5.7:

### 5.5.1. Recovering Original Indices for the Reduced Network

1. **Load Contingency, Branch, and Bus Mapping Data:** The algorithm starts by loading the contingency and branch mappings for the reduced network, alongside the bus mapping that links the renumbered buses to their original identifiers.
2. **Initialize Result Array:** A result array is initialized to store the combined data, which will include the original bus numbers, contingency data, and branch mappings.
3. **Recover Original Bus Numbers:** For each contingency, the original bus numbers are recovered using the mapping. This ensures that the reduced network's results can be correctly compared to the full network.
4. **Match Contingency and Branch Data:** The algorithm loops through both the contingency and branch mappings, checking for matching rows based on specific conditions. When a match is found, the contingency and branch data, along with the recovered original bus numbers, are appended to the result array.
5. **Update Saved Results:** The saved results for each contingency of the reduced network are updated with the original bus numbers, ensuring consistency when comparing the results of the full and reduced networks.

### 5.5.2. Updating Contingency Results with Original Bus Numbers

After recovering the original bus numbers, the next step is to update the saved results of each contingency file in the reduced network to ensure accurate comparisons with the full network. The process follows these steps:

1. **Load Mapping, Branch Mapping, and Contingency Data:** The algorithm begins by loading the bus mapping, branch mapping, and contingency files. This includes the original bus-to-renumbered bus mappings, branch connections, and contingency results.
2. **Extract Number of Contingencies:** The number of unique contingencies is extracted from the loaded contingency data to loop through and update each contingency file.
3. **Loop Through Each Contingency File:** For each contingency, the corresponding file is loaded, and the branch, bus, and generator data are prepared for updating.

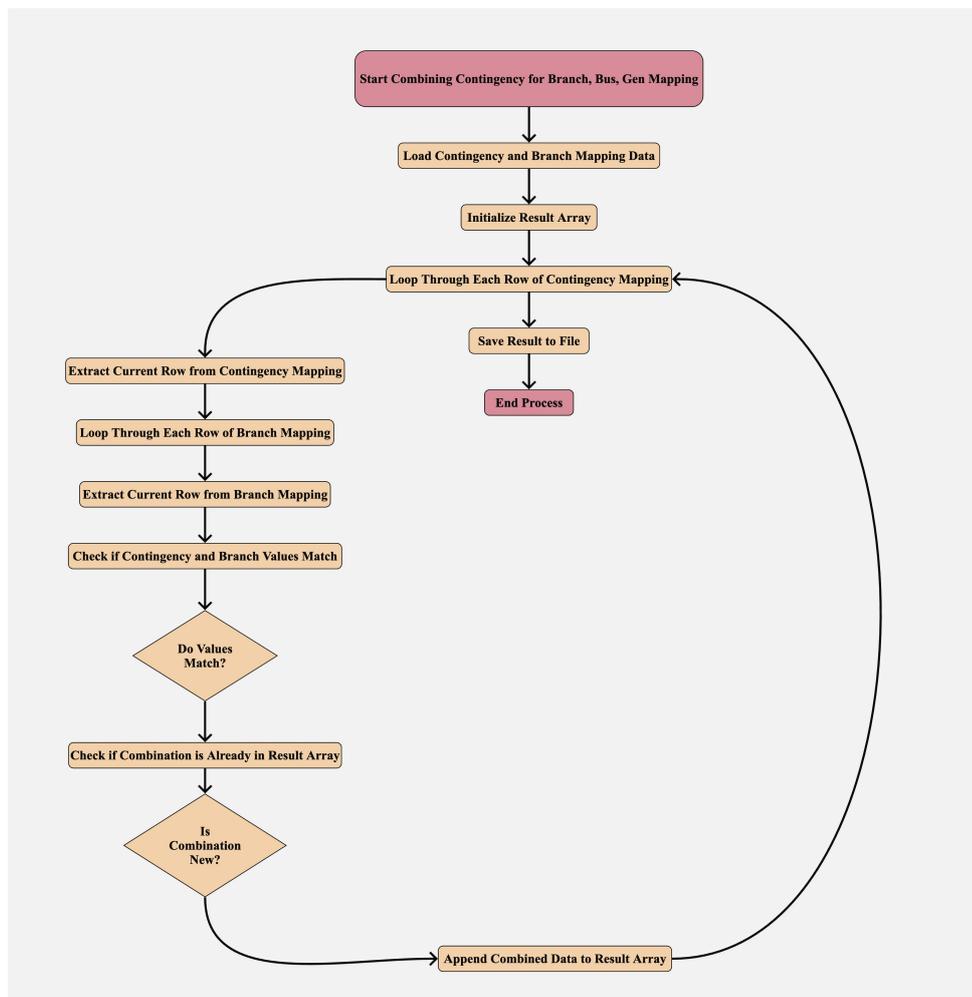


Figure 5.6: Flowchart for Recovering Original Indices for the Reduced Network

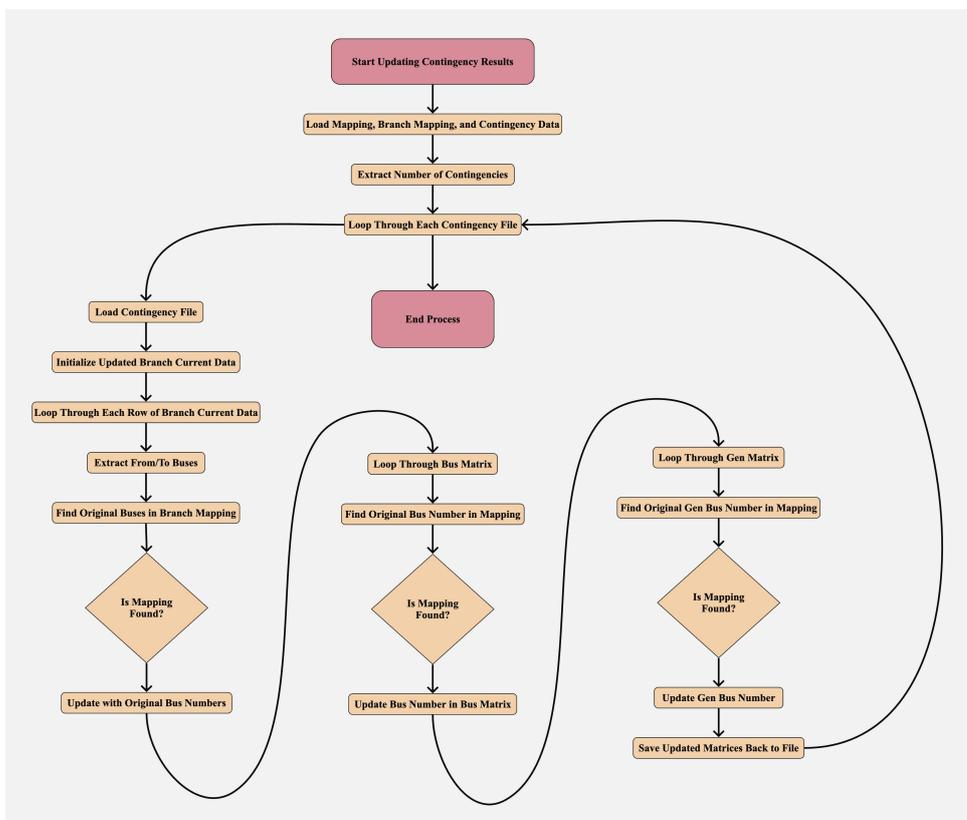


Figure 5.7: Flowchart for Updating Contingency Results with Original Bus Numbers

4. **Update Branch Data:** For each branch in the contingency file, the original "from" and "to" buses are recovered from the branch mapping. If a mapping is found, the original bus numbers replace the renumbered buses in the branch data.
5. **Update Bus Matrix:** The bus matrix is updated by looping through each bus entry and finding the corresponding original bus number in the mapping. The original bus numbers replace the renumbered ones in the bus matrix.
6. **Update Generator Data:** Similarly, the generator matrix is updated by mapping the renumbered bus numbers back to the original bus numbers. This ensures that generator buses are also correctly linked.
7. **Save Updated Results:** After updating the branch, bus, and generator matrices, the updated data is saved back into the contingency file. This ensures that future comparisons between the full and reduced networks use consistent bus numbering.

By completing these two stages, the reduced network's contingency results are accurately updated with the original bus numbers, allowing for a consistent and meaningful comparison with the full network during N-1 contingency analysis.

### 5.6. Comparative Analysis of Contingencies in Full and Reduced Power Networks

This section details the comparison between the full and reduced network models by analyzing common contingency events. The objective is to evaluate the effectiveness of the reduced network in replicating the full network under the same conditions, specifically by comparing critical electrical parameters such as voltage (V), apparent power (S<sub>bus</sub>), and current (I). The analysis is instrumental in assessing how well the reduced network maintains the reliability of power system performance during critical events.

#### 5.6.1. Identification of Common Contingencies

To perform a meaningful comparison between the full and reduced networks, it is essential to identify the common contingencies that occur in both systems. Contingencies refer to critical events such as transmission line failures that test the system's resilience and ability to remain stable.

## Methodology

The methodology for identifying common contingencies involves cross-referencing the contingency data from both networks to find matching events that affect the same branches. Once the branches are matched between the two networks, a set of common contingency scenarios is established for further analysis.

The flowchart in Figure 5.8 provides a detailed visual representation of the process for finding common contingencies between the full and reduced networks.

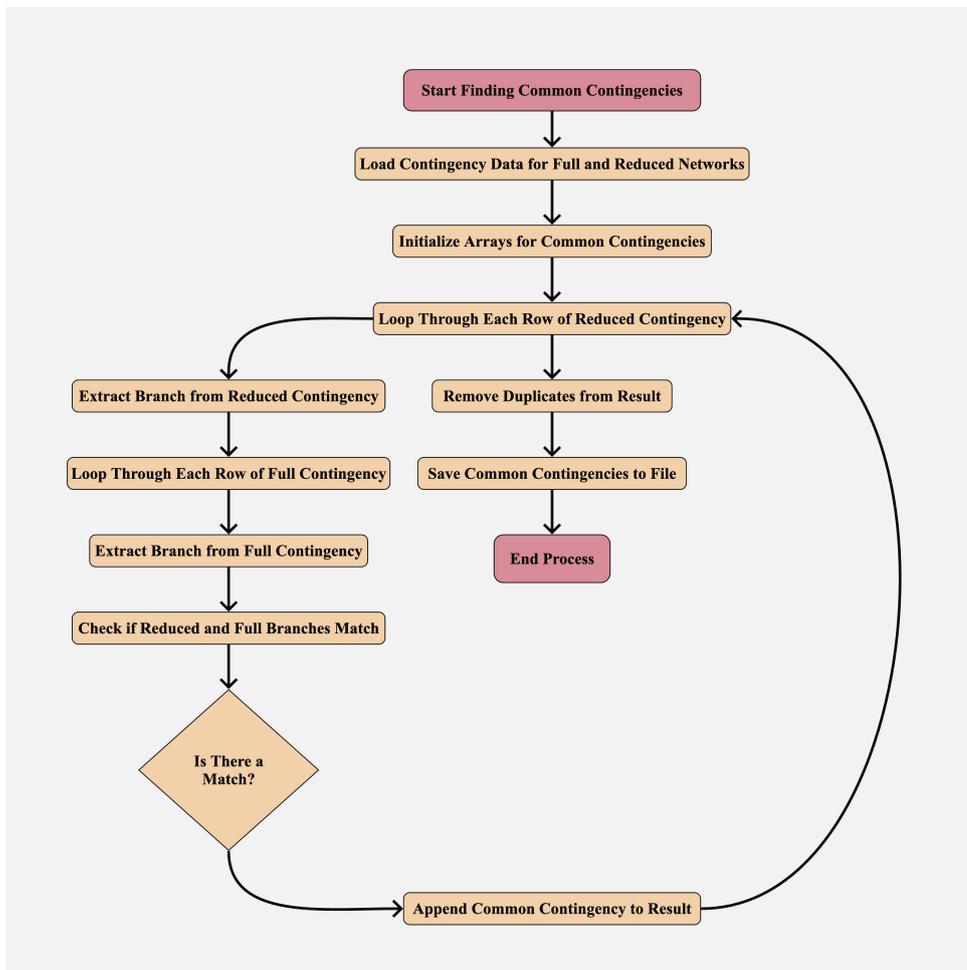


Figure 5.8: Flowchart for Finding Common Contingencies between Full and Reduced Networks

The key steps for finding common contingencies are as follows:

1. **Load Contingency Data:** Data for contingency events in both the full and reduced networks is loaded.
2. **Initialize Arrays for Common Contingencies:** Arrays are created to store

matching contingency events between the two networks.

3. **Loop Through Reduced Contingencies:** The algorithm iterates through each contingency event from the reduced network.
4. **Match with Full Network Contingencies:** Each reduced contingency is cross-referenced with the full network's data to find matches in branch information.
5. **Remove Duplicates:** Any duplicate contingency events are removed, and the results are saved for further comparison.

By establishing a set of common contingencies, we ensure that the events being analyzed have similar effects on both networks, allowing for an accurate comparative analysis.

### 5.6.2. Comparative Evaluation of Electrical Parameters

Once common contingencies have been identified, the next step is to evaluate and compare the electrical parameters of both networks during these events. The comparison focuses on voltage levels, apparent power ( $S_{bus}$ ), and current to determine how closely the reduced network mirrors the full network's behavior under stress.

### Data Preparation

For each common contingency event, power flow data is loaded for both networks, including voltage, apparent power, and branch current. Only buses common to both networks are considered for this analysis, ensuring a consistent basis for comparison.

The flowchart in Figure 5.9 shows the step-by-step process used to compare power flows between the full and reduced networks.

The comparative analysis process is broken down into these steps:

1. **Load Power Flow Data:** For each common contingency, power flow data for both the full and reduced networks is loaded, including voltage, apparent power, and current.
2. **Loop through Common Contingencies:** The algorithm iterates through each contingency to compare power flow results.
3. **Check for Power Flow Success:** The power flow results are checked for convergence in both networks. If both networks succeed, the differences in electrical parameters are calculated. If either network fails, the contingency is logged for further investigation.

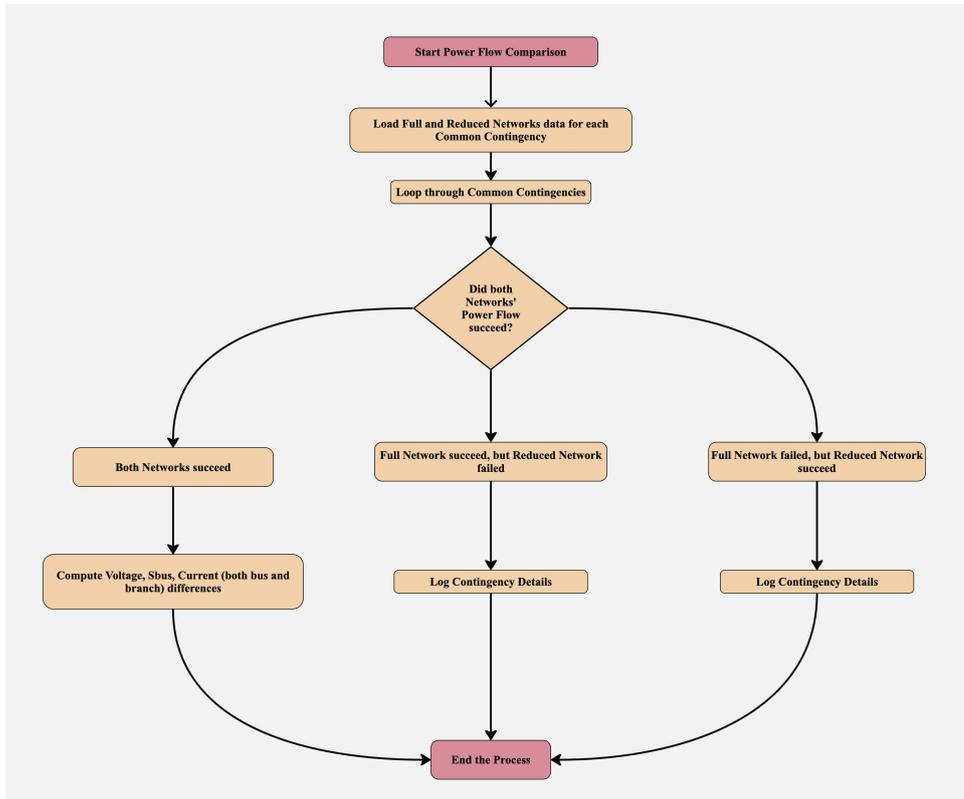


Figure 5.9: Flowchart for Power Flow Comparison between Full and Reduced Networks

4. **Calculate Differences:** If the power flow converges in both networks, the differences in voltage, apparent power, and current are computed for each common bus and branch.
5. **Log Results:** The differences or any failure to converge are logged for each contingency event.

### Handling Non-Converging Cases

Not all contingency scenarios may result in successful power flow convergence. In cases where the power flow does not converge in either the full or reduced network, the contingency event is logged but excluded from further detailed analysis. This ensures that only valid results are used in the final comparison.

#### 5.6.3. Significance of the Analysis

This comparative analysis provides critical insights into the behavior of the reduced network relative to the full network under contingency conditions. By focusing on common contingencies, we can assess how well the reduced network replicates key aspects such as

voltage stability, power flow redistribution, and current changes. Ultimately, the analysis reveals the strengths and limitations of the reduced network model in simulating critical power system events, offering a foundation for future improvements.

This process of comparison ensures that the reduced network remains a valid and efficient substitute for the full network, enabling accurate contingency and stability analyses without the computational burden of running simulations on the entire network.

## 5.7. Results interpretation and Discussion

The analysis of voltage differences, reactive power differences, branch current differences and generator bus status changes between the full and reduced networks during contingencies, highlighting the impact of network reduction are discussed below:

### 5.7.1. Voltage Difference and Relative Error Analysis

To assess how closely the reduced network replicates voltage behavior, we calculate the maximum, average, and minimum voltage differences between the full and reduced networks for each common contingency. The relative error for voltage is defined as:

$$\text{Relative Error (V)} = \frac{\Delta V}{V_{\text{full}}}$$

Where:

- $\Delta V$  is the difference in voltage between the full and reduced networks.
- $V_{\text{full}}$  is the reference voltage from the full network (maximum, average, or minimum, depending on the comparison).

### Graphical Representation

The graphical representation of the relative voltage error ( $\Delta V/V$ ) obtained for each common contingency is shown in Figure 5.10. The graph highlights the maximum, average, and minimum relative errors for each contingency, allowing for a clear comparison of how well the reduced network matches the full network in terms of voltage behavior.

### Discussion of Results

From Figure 5.10, we can observe the following key trends:

## 5| N-1 Contingency Simulation and Power Flow Results for Full and Reduced Networks

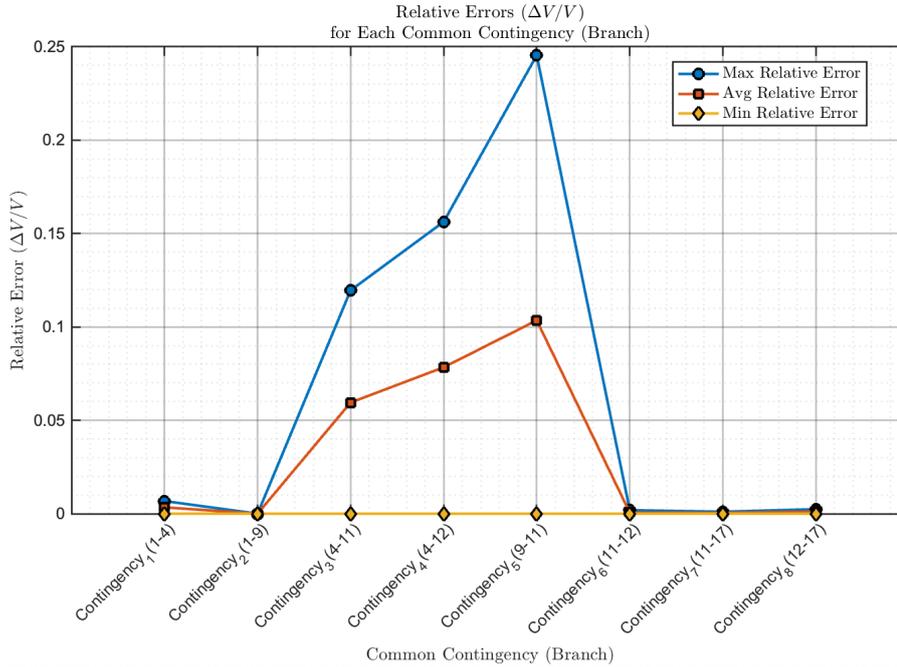


Figure 5.10: Relative Errors ( $\Delta V/V$ ) for Each Common Contingency (Branch).

- **Max Relative Error:** The maximum relative error is highest for Contingency 5 (9-11), reaching a value of approximately 0.24. This indicates a significant voltage difference between the full and reduced networks in this case.
- **Avg Relative Error:** The average relative error remains below 0.1 for most contingencies and highest for Contingency 5 (9-11).
- **Min Relative Error:** The minimum relative error is close to zero for all contingencies, which shows that for some branches, the voltage difference between the full and reduced networks is negligible.

The overall pattern suggests that while the reduced network replicates the voltage behavior of the full network closely in most cases, there are specific contingencies (such as Contingency 5) where the voltage behavior differs significantly. This may be due to the specific configuration of branches in the reduced network that are more sensitive to changes in load or generation. As the boundary buses were 11 and 12 in this case (as can be seen from Figure 3.3), though we already have a branch between 11 and 12, the impact is more near the boundary buses.

### 5.7.2. Voltage Differences and Significance

In the Figure 5.11, the color-coded branches are identified based on how the reduced network performs in response to the **contingency event of that specific branch**. For each common contingency between the full and reduced networks, the process follows these steps:

1. **Contingency Simulation:** A specific branch that serves as a common contingency in both the full and reduced networks is selected, and power flow analysis is performed under this contingency condition.
2. **Voltage Difference Calculation:** The maximum voltage difference across all buses in the reduced network is calculated by comparing the corresponding values in the full network.
3. **Branch Highlighting:** Depending on the maximum voltage difference for each common contingency, the branches are highlighted in different colors:
  - **Red:** If the voltage difference exceeds 0.1 p.u., indicating a large discrepancy.
  - **Yellow:** If the voltage difference is between 0.01 and 0.1 p.u., showing moderate deviation.
  - **Green:** If the voltage difference is less than 0.01 p.u., indicating close agreement.
  - **Black:** If the power flow succeeds in one network but fails in the other, showing inconsistency.

### Significance

- **Precise Performance Assessment:** This visualization enables a focused examination of how the reduced network responds to specific branch contingencies, offering a clear understanding of areas where the reduced model either replicates or diverges from the full network's behavior.
- **Voltage Stability Insights:** By identifying the maximum voltage differences between the full and reduced networks, this method pinpoints critical areas where voltage discrepancies are the highest, guiding necessary adjustments in the reduced network to enhance accuracy.
- **Efficient Visual Analysis:** The color-coded branches serve as a straightforward visual tool for system planners, highlighting sections of the network where improve-

## 5| N-1 Contingency Simulation and Power Flow Results for Full and Reduced Networks

80

ments are needed to better match the reduced network's performance to that of the full network under contingency conditions. This aids in optimizing the reduced model for reliable analysis.

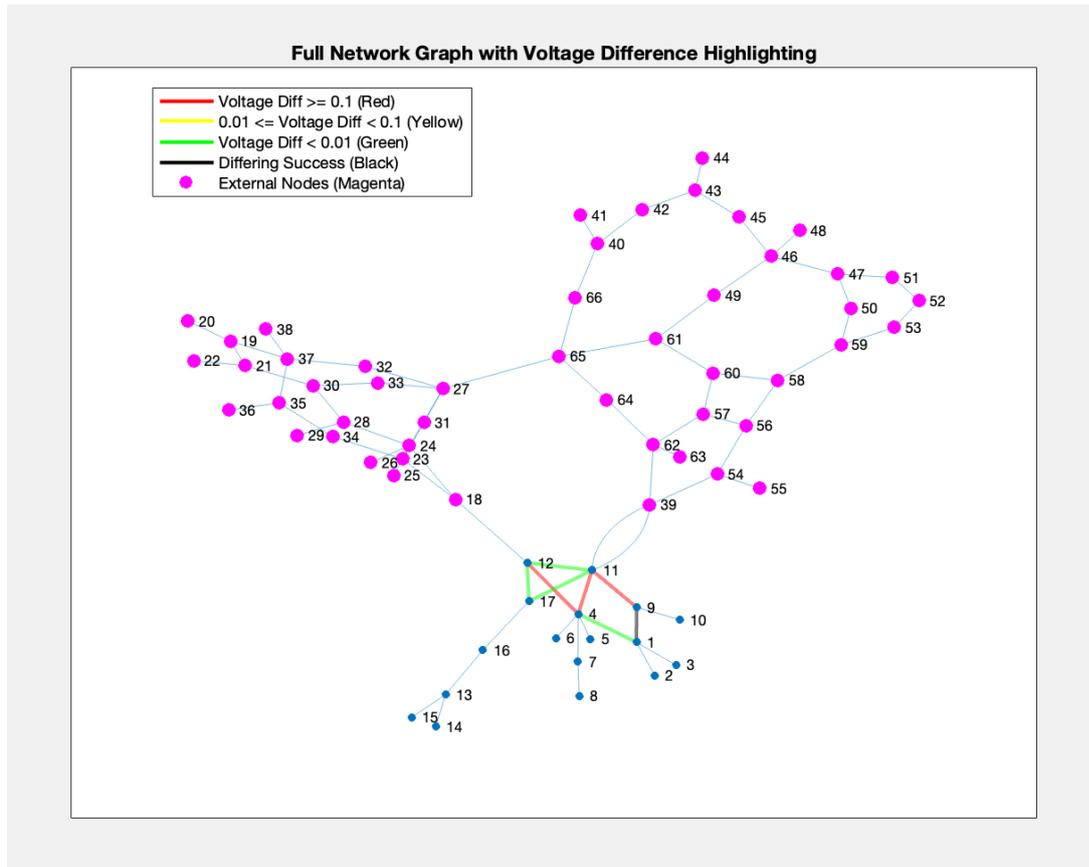


Figure 5.11: Full Network Graph with Voltage Difference Highlighting. The branches are color-coded to reflect voltage discrepancies between the full and reduced networks for each common contingency.

### Discussion of Results

Figure 5.11 demonstrates that while the reduced network performs well in most areas, there are specific branches (highlighted in red) where significant voltage discrepancies occur. Those are the same branches which showed highest Relative error (V) in Figure 5.10. These discrepancies highlight areas where the reduced model needs further refinement to more accurately simulate the full network's behavior under contingency conditions.

### 5.7.3. Generator Bus Status Change from PV-to-PQ

This graph tracks generator buses that change classification from PV (voltage control) in the full network to PQ (no voltage control) in the reduced network:

- **PV-to-PQ Status Changes:** Red markers indicate generator buses that switch from PV to PQ during contingencies in Reduced Network.
- **Annotated Bus Labels:** Each marker is labeled with the generator bus number undergoing this change for each contingency.

### Methodology used for Checking Generator Bus Status Shift

To assess shifts in generator bus status from PV to PQ during contingencies, we applied a structured approach using voltage and reactive power tolerances. This process helps to identify any status changes between the full and reduced networks.

### Tolerance Levels

Two critical tolerance levels are defined:

- **Voltage Tolerance ( $\Delta V$ ):** Set at  $10^{-5}$  p.u., representing the acceptable deviation from the voltage setpoint.
- **Reactive Power Tolerance ( $\Delta Q$ ):** Set at  $10^{-3}$  p.u., determining how close the reactive power is to its upper or lower limits.

### Classification Process

For each common contingency in both networks, the following steps are performed:

- **Reactive Power Deviation ( $\Delta Q$ ):** The deviations of the actual reactive power from its limits ( $Q_{\min}, Q_{\max}$ ) are calculated:

$$\Delta Q_{\min} = |Q - Q_{\min}|, \quad \Delta Q_{\max} = |Q_{\max} - Q|$$

- **Voltage Deviation ( $\Delta V$ ):** The deviation of the actual voltage from the setpoint is determined:

$$\Delta V = |V_{\text{bus}} - V_{\text{setpoint}}|$$

- **Bus Classification:**

## 5| N-1 Contingency Simulation and Power Flow Results for Full and Reduced Networks

82

- **PQ Bus (Type 1)**: If the reactive power is near its limits ( $Q_{\min}$  or  $Q_{\max}$ ) and the voltage deviation exceeds the tolerance ( $\Delta V$ ).
- **PV Bus (Type 2)**: If the reactive power remains within limits and the voltage deviation is below the tolerance.

### Identifying Status Shifts

For each common generator bus in both networks, the classification is compared. A **status shift** is recorded if a bus is classified as PV (Type 2) in the full network and PQ (Type 1) in the reduced network. These shifts indicate potential differences in voltage control behavior between the two network models during contingencies.

By comparing these classifications, we can evaluate how well the reduced network approximates the full network's behavior in maintaining voltage and reactive power control under different contingencies.

### Graphical Representation

The graphical representation of the PV-to-PQ status change for generator buses obtained is shown in Figure 5.12. It highlights the generator bus that switches from PV to PQ for each contingency, offering insight into how the reduced network impacts voltage control.

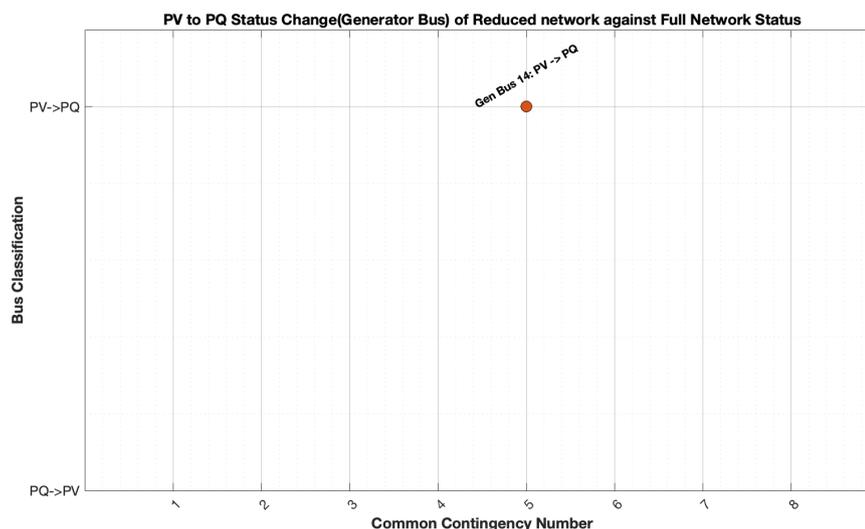


Figure 5.12: PV to PQ Status Change of Generator Bus in the Reduced Network against Full Network Status. The graph tracks generator buses that change classification from PV (voltage control) to PQ (no voltage control) during contingencies.

### Significance of the Graph

- **Voltage Control Insights:** The graph illustrates where generator buses lose voltage control (PV to PQ), showing potential impacts of network reduction on system stability. This shift can lead to voltage regulation challenges in the reduced network, as these buses no longer control voltage.
- **System Stability:** The shift from PV to PQ can have serious implications for system stability, particularly in cases where voltage control is essential for maintaining grid reliability. The loss of voltage control at critical buses may cause voltage instability or degrade the performance of the reduced network during contingencies.
- **Contingency-Specific Effects:** The annotation of the generator bus number for each contingency allows system planners to pinpoint specific contingencies that cause this classification shift, helping to identify areas where the reduced network requires adjustment.

### Discussion of Results

The results presented in Figure 5.12 reveal a significant PV-to-PQ status change at **Generator Bus 14** during Contingency 5. This shift indicates a loss of voltage control at the generator, underscoring a critical vulnerability in maintaining system stability under stressed conditions. Such a status change is crucial, as it demonstrates how generator bus classification adjustments can profoundly affect the network's ability to sustain voltage regulation, particularly when reactive power limits are exceeded.

In the case of Contingency 5 (9-11), **Generator Bus 14** is pushed beyond its reactive power capability, as seen in Figure 5.13, exacerbating the situation in the affected area (highlighted). The contingency's impact becomes more severe due to the unavailability of the reactive power needed to balance the system, especially in the nearby buses. Consequently, **Bus 14** experiences the largest voltage difference, as illustrated in Figure 5.14, emphasizing the cascading effect of the generator's inability to regulate voltage under stressed conditions.

In conclusion, these results highlight the need for proactive contingency planning and reactive power management, particularly at boundary nodes. Ensuring that boundary buses have sufficient reactive power reserves is not only crucial for supporting local areas but also for preserving overall grid stability in the event of unexpected contingencies.

5| N-1 Contingency Simulation and Power Flow Results for Full and Reduced Networks

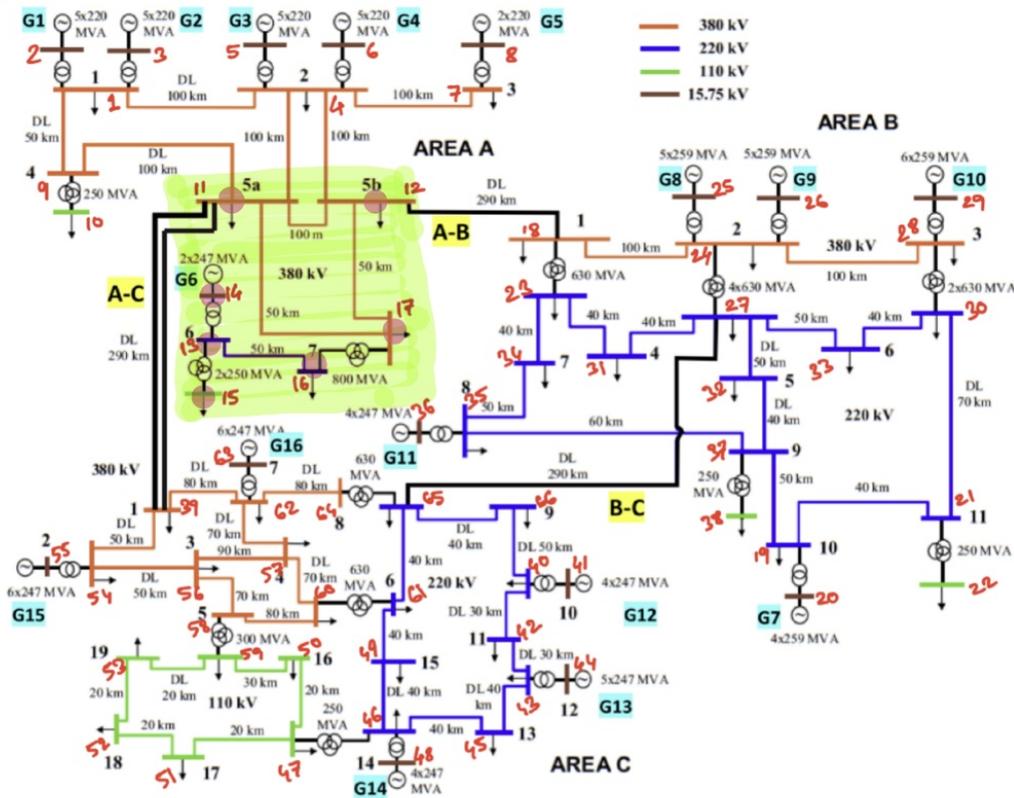


Figure 5.13: Single-line diagram of the 66-bus *Rueda\_network\_3\_1* with impacted area highlighted for contingencies 5, 4, and 3.

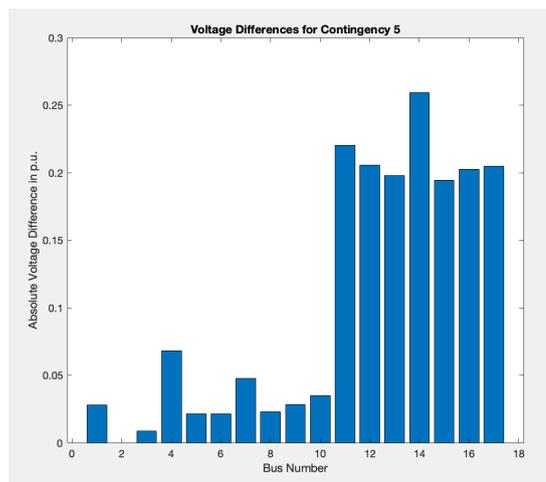


Figure 5.14: Voltage differences for Contingency 5.

5.7.4. Relative Error Analysis for Reactive Power (Q)

For reactive power ( $Q$ ), differences between the full and reduced networks are calculated for generator buses, which are critical for reactive power support and voltage regulation.

The relative error for reactive power is calculated as:

$$\text{Relative Error (Q)} = \frac{\Delta Q}{Q_{\text{ref}}}$$

Where:

- $\Delta Q$  is the difference in reactive power at generator buses between the full and reduced networks.
- $Q_{\text{ref}}$  is the maximum reactive power in the full network.

This analysis helps to evaluate how accurately the reduced network maintains reactive power at generator buses, which is essential for system stability. Large relative errors in  $Q$  may indicate an inability of the reduced network to replicate reactive power control seen in the full network, leading to voltage instabilities.

### Graphical Representation

The graphical representation of the relative error for reactive power ( $\Delta Q/Q$ ) obtained is shown in Figure 5.15. It highlights the maximum, average, and minimum relative errors for each common contingency involving generator buses. The graph provides insights into how well the reduced network matches the full network in terms of reactive power control.

### Discussion of Results

From Figure 5.15, the following observations can be made:

- **Max Relative Error:** The maximum relative error is highest for Contingency 5 (9-11), reaching a value of approximately 0.011.
- **Avg Relative Error:** The average relative error is generally below 0.006 for all contingencies.
- **Min Relative Error:** The minimum relative error remains close to zero for all contingencies.

The overall pattern indicates that while the reduced network performs well in most cases, there are specific contingencies where reactive power discrepancies arise, particularly for Contingency 5 (9-11). These differences may result in voltage regulation issues in the reduced network as discussed in the previous section, emphasizing the need to refine the model to handle reactive power control more accurately under these conditions.

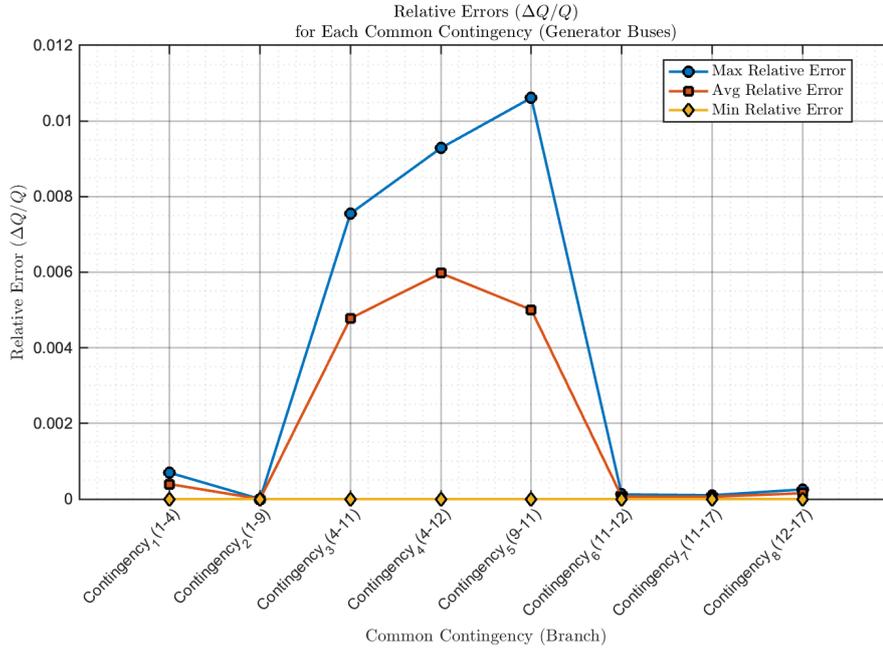


Figure 5.15: Relative Errors ( $\Delta Q/Q$ ) for Each Common Contingency (Generator Buses). The graph compares the maximum, average, and minimum relative reactive power errors between the full and reduced networks for each common contingency.

### 5.7.5. Relative Error Analysis for Branch Current (I) Difference

Branch current differences between the full and reduced networks are analyzed to assess current flow changes during contingencies. The relative error for branch current is defined as:

$$\text{Relative Error (I)} = \frac{\Delta I}{I_{\text{full}}}$$

Where:

- $\Delta I$  is the difference in current between the full and reduced networks.
- $I_{\text{full}}$  is the maximum branch current in the full network, used as the reference for comparison.

### Graphical Representation

The graphical representation of the relative error for branch current ( $\Delta I/I$ ) obtained is shown in Figure 5.16. It highlights the maximum, average, and minimum relative errors for each common contingency. This visualization helps identify where the reduced network

deviates most from the full network in terms of current flow.

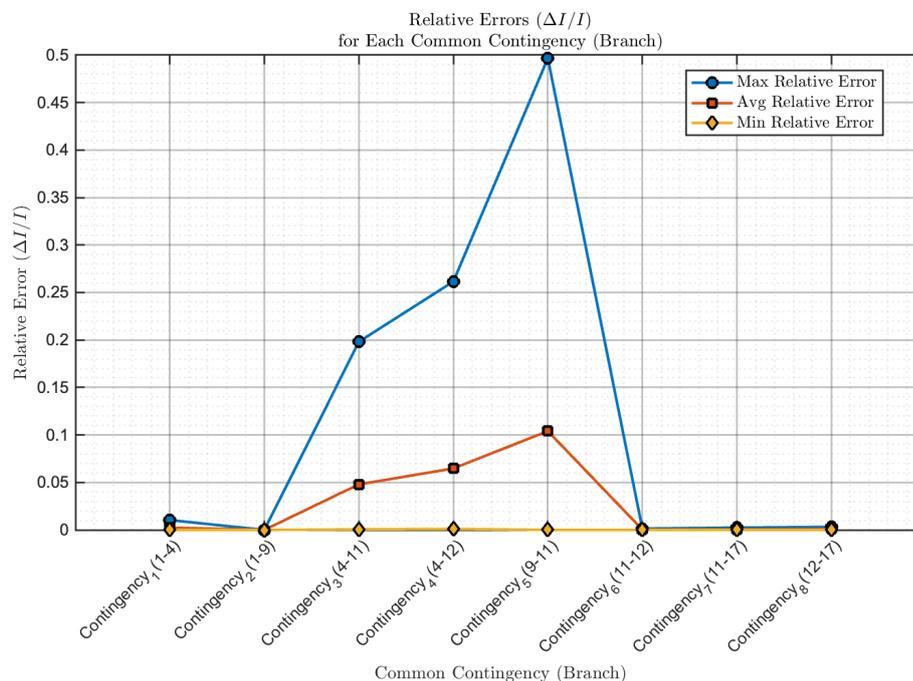


Figure 5.16: Relative Errors ( $\Delta I/I$ ) for Each Common Contingency (Branch). The graph compares the maximum, average, and minimum relative current errors between the full and reduced networks for each common contingency.

## Discussion of Results

From Figure 5.16, we observe the following trends:

- Max Relative Error:** The highest relative error occurs for Contingency 5 (9-11), where the maximum relative error around 0.49. This indicates a substantial difference in current flow between the full and reduced networks under this contingency, pointing to a new re-distribution of branch currents due to the changes in voltage and reactive power scenarios.
- Avg Relative Error:** The average relative error for most contingencies remains below 0.1, suggesting that while there are some deviations in current flow, the reduced network generally performs adequately in replicating current behavior. However, the higher values for Contingencies 4 (4-11) and 5 (9-11) suggest areas where the reduced model may require further tuning.
- Min Relative Error:** The minimum relative error remains very close to zero across all contingencies.

Overall, the reduced network performs well for most contingencies, but there are specific areas, such as Contingency 5 (9-11), where current flow discrepancies are significant. This may be due to the specific configuration of branches in the reduced network that are more sensitive to changes in load or generation. As the boundary buses were 11 and 12 in this case (as can be seen from Figure 3.3), though we already have a branch between 11 and 12, the impact is more near the boundary buses.

## 5.8. Experimenting with Boundary Nodes of the Reduced Network

In the previous analysis, the regions near the boundary nodes showed significant **voltage discrepancies** between the full and reduced networks. The discrepancies were particularly concerning, as they led to relative voltage errors ( $\Delta V/V$ ) exceeding **0.1** in some critical contingencies.

To address this issue, initially an experiment was conducted by **shifting the boundary nodes** one layer towards the external network i.e., boundary nodes as 18 and 39, but some of the contingencies near the boundary nodes were not converged. This could be explained by the fact that the impedance between the studied area and equivalent network is increased with transmission lines A-C and A-B as could be seen from Figure 5.17. As the equivalent is a passive network (with no reactive support) that may cause the divergence for some contingencies near the boundary.

Therefore, another attempt of **boundary shifting** is done with boundary nodes 18, 39 and 62 so that **Generator bus 63** could support the Reactive Power demands for the boundary region. The new boundary (B3) could be visualised from the Figure 5.17.

The Figure 5.18 visualizes the classification of nodes after the boundary shift (B3), where the **internal (as green)**, **boundary (as blue)**, and **external (as red) nodes** are redefined based on this new approach. By shifting the boundary nodes, the internal network was expanded, allowing for better replication of reduced network behavior.

The results after shifting the boundary nodes demonstrated a notable reduction in voltage discrepancies. Below are the key figures that summarize the **relative voltage error** and **voltage difference behavior** after the boundary shift.

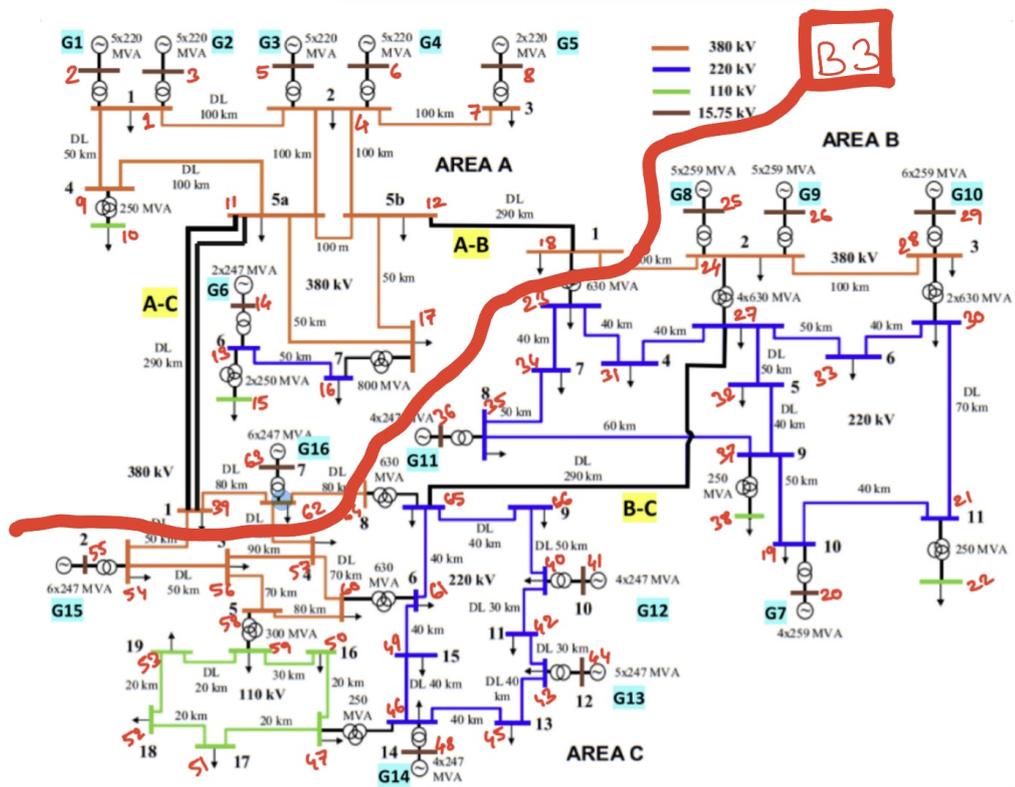


Figure 5.17: Network with Boundary nodes 18, 39 and 62

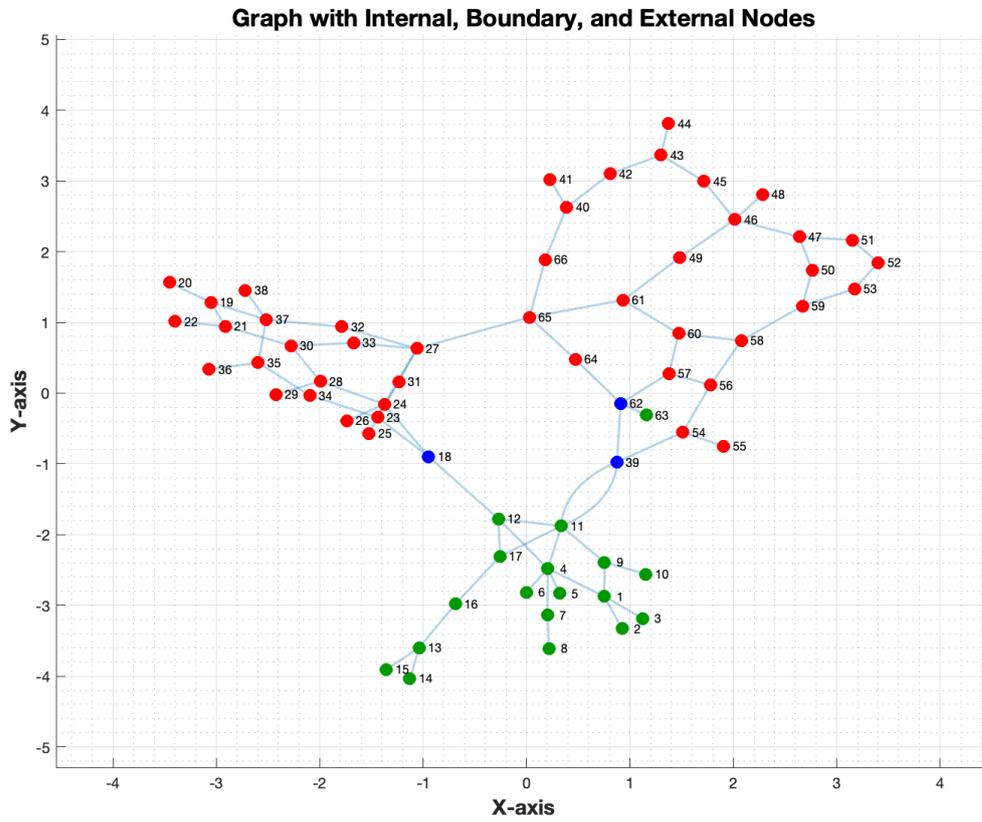


Figure 5.18: Graph with Internal(green), Boundary(blue), and External(red) Nodes After Boundary Shift.

### 5.8.1. Relative Voltage Error ( $\Delta V/V$ )

Before the boundary shift, the maximum relative voltage error reached **0.24** for *Contingency 5 (9-11)*. After shifting the boundary nodes, the maximum error was significantly reduced to **0.034**, reflecting a **85% improvement** as shown in the Figure 5.19. This is particularly significant as it brings the voltage discrepancies within an acceptable range for most contingencies.

- **Before:**

- Max Relative Error (Contingency 5):  $\sim 0.24$
- Avg Relative Error (Contingency 5):  $\sim 0.10$

- **After:**

- Max Relative Error (Contingency 5):  $\sim 0.034$
- Avg Relative Error (Contingency 5):  $\sim 0.02$

This reduction shows that the strategic boundary node shift had a major impact on improving the reduced network’s performance in replicating voltage behavior.

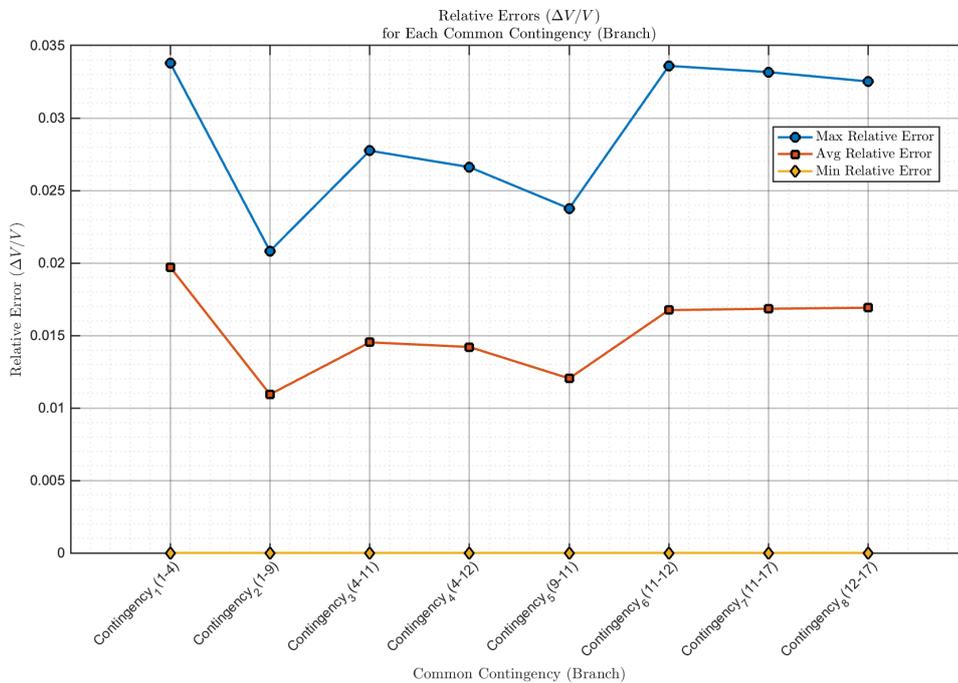


Figure 5.19: Relative Voltage Errors ( $\Delta V/V$ ) for Each Common Contingency After Boundary Shift.

### 5.8.2. Voltage Difference Plot

In addition to the relative errors, a **voltage difference plot** was generated to visually represent the differences between the full and reduced networks. The plot uses a *color-coded* approach to highlight the branches based on the magnitude of the maximum voltage difference.

After the strategic boundary shift, as shown in the Figure 5.20 no branches exhibited **red** colors, and fewer branches exhibited **yellow** colors, indicating that the voltage differences between the full and reduced networks were reduced.

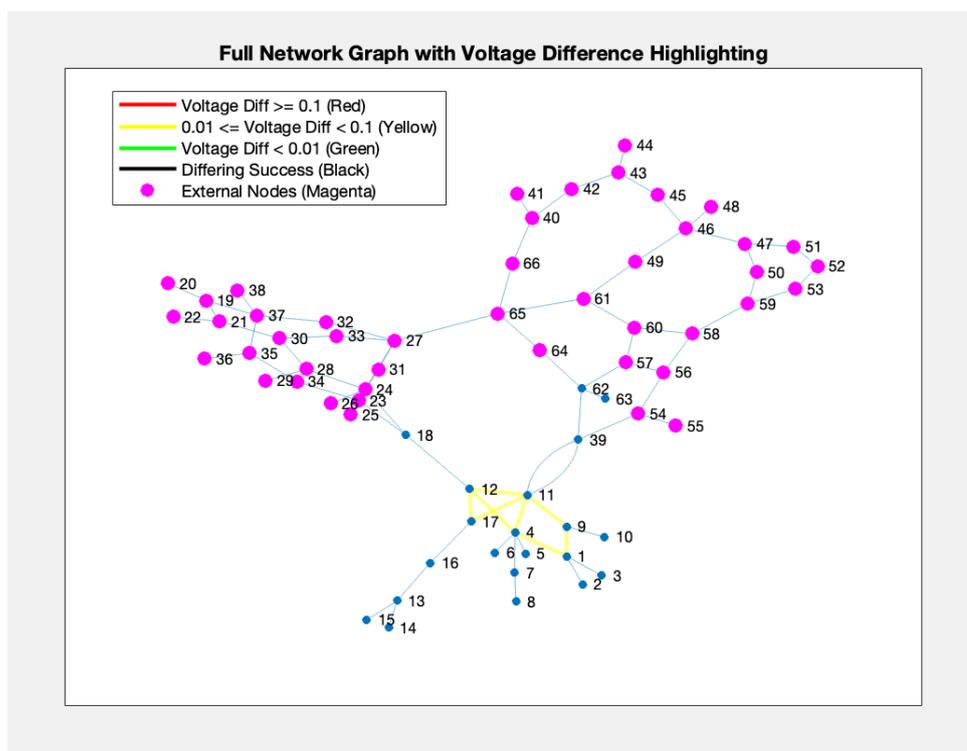


Figure 5.20: Full Network Graph with Voltage Difference Highlighting After Boundary Shift.

### 5.8.3. Generator Bus Status Shift (PV to PQ)

In the reduced network, one generator bus shifted from **PV (voltage control)** to **PQ (load bus)**, indicating a loss of voltage control.

In the *case before*, **Generator Bus 14** experienced a **PV-to-PQ status shift** during *Contingency 5 (9-11)*, which could have significant implications for system stability. After the boundary shift, **no PV-to-PQ shifts** were observed, and **Generator Bus 14** retained its **PV** classification, reflecting an improvement in voltage control for the generator

buses.

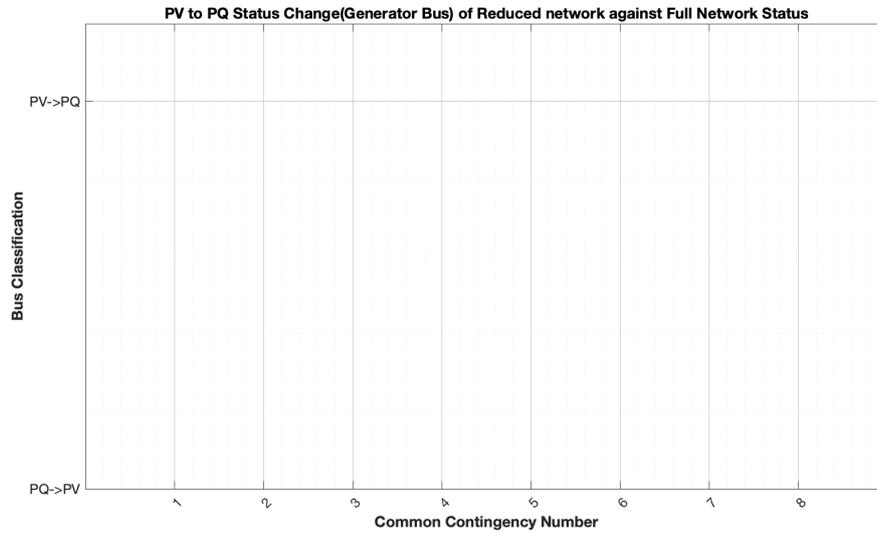


Figure 5.21: PV to PQ Status Change for Generator Buses After Boundary Shift.

#### 5.8.4. Relative Branch Current Error ( $\Delta I/I$ )

Before the boundary shift, the maximum relative branch current error reached **0.49** for *Contingency 5 (9-11)*. After shifting the boundary nodes, the maximum error was significantly reduced to **0.012**, reflecting a **97% improvement** as shown in the Figure 5.22. This is particularly significant as it brings the branch current discrepancies within an acceptable range for most contingencies.

- **Before:**
  - Max Relative Error (Contingency 5):  $\sim 0.49$
  - Avg Relative Error (Contingency 5):  $\sim 0.1$
- **After:**
  - Max Relative Error (Contingency 5):  $\sim 0.012$
  - Avg Relative Error (Contingency 5):  $\sim 0.004$

#### 5.8.5. Relative Reactive Power Error ( $\Delta Q/Q$ )

Before the boundary shift, the maximum relative Reactive Power error reached **0.011**. After shifting the boundary nodes, the maximum error was significantly reduced to **0.0018**, reflecting a **84% improvement** as shown in the Figure 5.23.

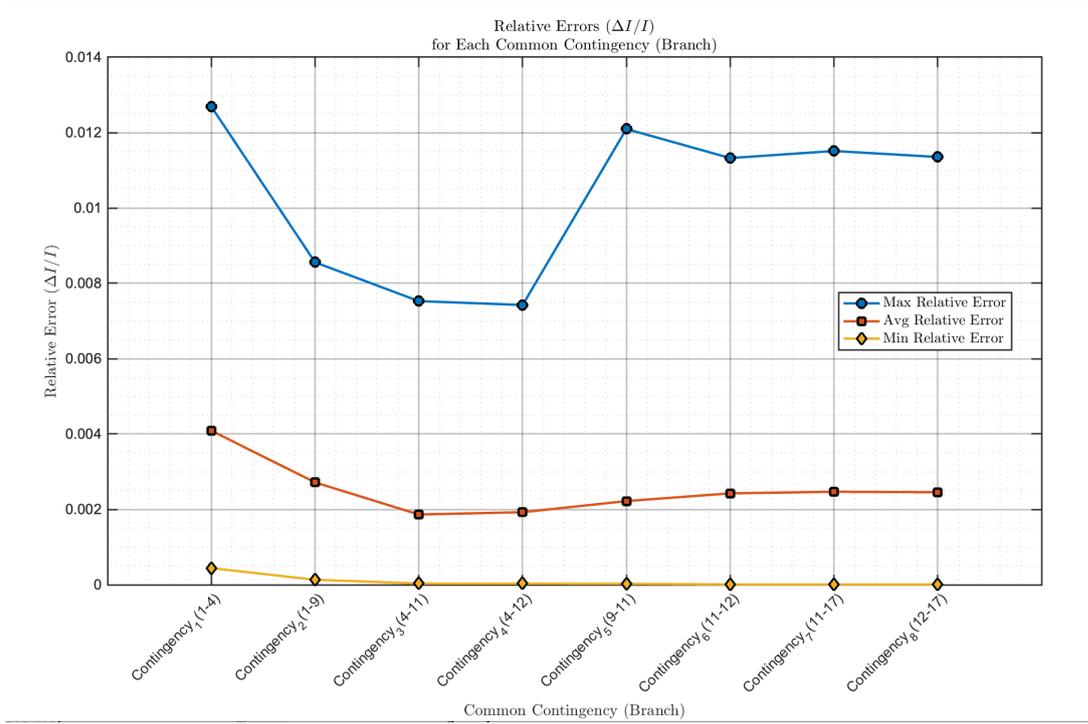


Figure 5.22: Relative Branch Current Errors ( $\Delta I/I$ ) for Each Common Contingency After Boundary Shift.

- **Before:**
  - Max Relative Error (Contingency 5):  $\sim 0.011$
  - Avg Relative Error (Contingency 5):  $\sim 0.006$
- **After:**
  - Max Relative Error (Contingency 5):  $\sim 0.0018$
  - Avg Relative Error (Contingency 5):  $\sim 0.0012$

Hence, the results clearly show that the strategic boundary node selection has significantly improved the reduced network's (the region where we were concentrating) performance, particularly in reducing the maximum relative error for voltage, branch current and Reactive Power. The voltage difference plot indicates that no branches exhibit large voltage deviations, and most branches now demonstrate minimal discrepancies. Furthermore, the generator bus status analysis shows that the boundary shift helped maintain voltage control at critical generator buses, improving the network's stability under contingency conditions.

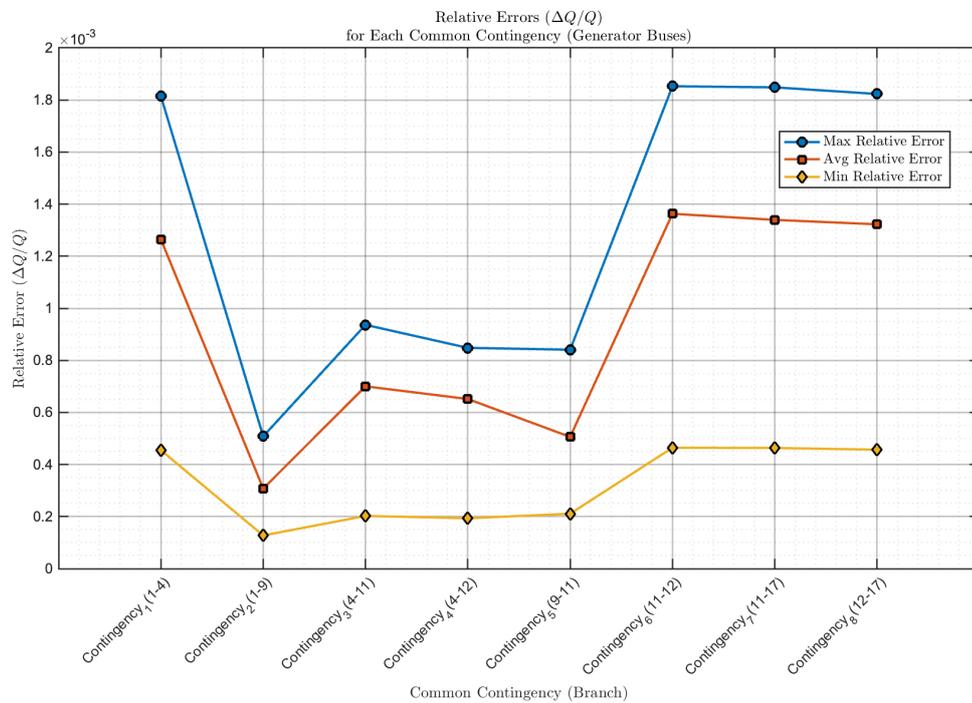


Figure 5.23: Relative Reactive Power Errors ( $\Delta Q/Q$ ) for Each Common Contingency After Boundary Shift.



# 6 | Conclusion

This thesis has delved into the intricate balance between network reduction techniques and their effects on both accuracy and computational efficiency in power system analysis, with a particular emphasis on **Ward's Equivalent**. The findings underscore that network reduction, when applied thoughtfully, can significantly streamline the analysis of complex power grids by reducing external system complexity while maintaining the core integrity of the internal network. However, the research also uncovered important limitations, particularly in managing boundary nodes, where voltage and reactive power discrepancies can arise under certain contingencies.

Under normal operating conditions, Ward's network reduction method effectively preserves the stability and performance of the internal network. While boundary buses exhibit noticeable current fluctuations due to interactions with the reduced external system, the deviations in voltage and phase angles remain negligible. This highlights the technique's ability to handle steady-state scenarios reliably.

Contingency simulations revealed critical insights: voltage differences exceeding acceptable thresholds occurred in some cases, underscoring the importance of **strategic boundary node selection** and reactive power support. A key solution identified in this research was the adjustment of boundary nodes—by strategic shifting them one layer towards the external network, voltage discrepancies were markedly reduced along with the relative error for branch current and Reactive Power. This finding reinforces the idea that boundary node placement is pivotal in ensuring that the reduced network accurately reflects the full system's behavior during stressed conditions.

The study also emphasized the challenges posed by generator bus status changes, particularly the shift from PV to PQ under stressed conditions, which can lead to substantial voltage imbalances and pose a threat to system stability. These findings suggest the need for ongoing refinement of network reduction techniques, especially in scenarios where voltage control is crucial.

The comparison between the full and reduced networks illuminated the inherent trade-offs of network reduction. While the reduced network offers substantial computational

advantages, the research identified areas for improvement, particularly in refining the method to better manage discrepancies in reactive power and current flow. As a result, this thesis concludes that Ward's Equivalent remains a highly valuable tool for large-scale power system analysis. However, its effectiveness depends on a nuanced understanding of its limitations, especially regarding voltage stability and reactive power management.

Looking ahead, the application of Ward's Equivalent can extend to real-time simulations and market analyses, offering substantial benefits in managing complex scenarios with reduced computational demands. Nonetheless, continued refinement of the technique is essential to ensure that reduced networks remain accurate and reliable representations of the full system, especially in critical contingency situations. For those situations, network equivalents should be upgraded to an "active" equivalent model so that it can provide required Reactive Power support. Finally, This thesis paves the way for future advancements in network reduction techniques and reaffirms their growing importance in the evolving landscape of modern power system management.

# Bibliography

- [1] S. M. Ashraf, B. Rathore, and S. Chakrabarti. Performance analysis of static network reduction methods commonly used in power systems. In *IEEE Power Energy Society General Meeting*, pages 1–5. IEEE, IEEE, 2014. doi: 10.1109/PESGM.2014.6939895.
- [2] E. Cremona. Breaking borders: Doubling electricity interconnection in europe. <https://ember-climate.org/insights/research/in-it-together-cee-power-system/>, June 2023. Accessed: 2024-09-13.
- [3] P. Dimo. *Nodal analysis of power systems*. Abacus Press, Kent, England, 1975.
- [4] J. F. Dopazo, G. Irisarri, and A. M. Sasson. Real-time external system equivalent for online contingency analysis. *IEEE Transactions on Power Apparatus and Systems*, PAS-98(6):498–508, 1980.
- [5] European Council and Council of the European Union. Fit for 55. <https://www.consilium.europa.eu/en/policies/green-deal/fit-for-55/>, n.d. Accessed: Sep. 13, 2024.
- [6] E. C. Housos, G. Irisarri, R. M. Porter, and A. M. Sasson. Steady-state network equivalents for power system planning applications. *IEEE Transactions on Power Apparatus and Systems*, PAS-99(6):2301–2310, 1980.
- [7] G. Kron. Tensor analysis of networks. *Transactions of the American Institute of Electrical Engineers*, 57(12):1057–1064, 1939.
- [8] K. Lo, L. Peng, J. Macqueen, A. Ekwue, and D. Cheng. An extended ward equivalent approach for power system security assessment. *Electric Power Systems Research*, 42:181–188, 1997. doi: S0378-7796(96)01203-5. URL <https://www.sciencedirect.com/science/article/pii/S0378779696012035>.
- [9] A. Monticelli, S. Deckmann, A. Garcia, and B. Stott. Real-time external equivalents for static security analysis. *IEEE Transactions on Power Apparatus and Systems*, PAS-98(2):498–508, 1979.

- [10] H. Oh. A new network reduction methodology for power system planning studies. *IEEE Transactions on Power Systems*, 25(2):117–124, 2010.
- [11] I. Pavić, Z. Hebel, and M. Delimar. Power system equivalent based on an artificial neural network. In *Proceedings of the 23rd International Conference on Information Technology Interfaces (ITI)*, pages 359–365, Pula, Croatia, 2001. University of Zagreb, Faculty of Electrical Engineering and Computing, IEEE.
- [12] M. Ramirez-Gonzalez, M. Bossio, F. R. S. Sevilla, and P. Korba. Evaluation of static network equivalent models for n-1 line contingency analysis. In *2022 4th Global Power, Energy and Communication Conference (GPECOM)*, pages 328–333. IEEE, 2022. doi: 10.1109/GPECOM55404.2022.9815713.
- [13] A. Shapovalov, C. Spieker, and C. Rehtanz. Network reduction algorithm for smart grid applications. In *23rd Australasian Universities Power Engineering Conference*, 2013.
- [14] A. Shapovalov, C. Spieker, and C. Rehtanz. Forecast-based network reconfiguration algorithm, 2013. Retrieved from <https://ieeexplore.ieee.org/document/6522365>.
- [15] Statista. Length of electricity transmission lines in europe as of 2023, by country. <https://www.statista.com/statistics/1459893/electricity-transmission-lines-length-europe-by-country/>, 2024. Accessed: 2024-09-13.
- [16] Statista. Number of electricity substations in italy from 2010 to 2017. <https://www.statista.com/statistics/884845/electricity-substations-in-italy/>, 2024. Accessed: 2024-09-13.
- [17] J. B. Ward. Equivalent circuits for power flow studies. *AIEE Transactions on Power Apparatus and Systems*, 68:373–382, 1949.
- [18] Wikipedia. Electricity sector in italy. [https://en.wikipedia.org/wiki/Electricity\\_sector\\_in\\_Italy](https://en.wikipedia.org/wiki/Electricity_sector_in_Italy), 2024. Accessed: 2024-09-13.
- [19] Wikipedia. European network of transmission system operators for electricity. [https://en.wikipedia.org/wiki/European\\_Network\\_of\\_Transmission\\_System\\_Operators\\_for\\_Electricity](https://en.wikipedia.org/wiki/European_Network_of_Transmission_System_Operators_for_Electricity), 2024. Accessed: 2024-09-13.
- [20] Wikipedia. Synchronous grid of continental europe. [https://en.wikipedia.org/wiki/Synchronous\\_grid\\_of\\_Continental\\_Europe](https://en.wikipedia.org/wiki/Synchronous_grid_of_Continental_Europe), 2024. Accessed: 2024-09-13.

# A | Appendix A

The codes that were used for this Thesis work are shown below:

- Full MATLAB Code for Network Reduction:

```

1 function launch_PF
2
3 maindir = cd;
4
5 netFile = 'Rueda_network_3_1.mat';
6 load(netFile, 'baseMVA', 'bus', 'branch', 'gen', 'nomgen', '
   nomnod', 'nombranch', ...
7     'br_ph_shifters', 'branch_N1_constrained', '
   branch_notN_constrained', ...
8     'GAMS_data_gen_bid', 'GAMS_data_Supply_bids', 'gens_fake'
   , 'branchset', 'nodInfo');
9
10 genVctrl = gen(:, end);
11
12 % adapt bustype to generator AV mode and produced power
13 % mode -> PF - Power Flow and ORPF mode OPF - OPF
14
15 [bus, gen] = adjust_genQctrl(bus, gen, genVctrl);
16
17 %-Check PF
18 gauss      = 0;
19 limitato   = 1;
20 disp       = 0;
21
22 addpath("/Users/parthasmacbook/Desktop/Thesis/Final/1st_case/
   Load_flow");
23 %caso base sicurezza n

```

```

24 [busPF, genPF, ~ ] = gen_Vconst_cut(bus, gen, nomgen);
25
26 [bus1, gen1, success, Yf, Yt, Y, V]=...
27     load_flow(baseMVA, busPF, genPF, branch(:, [1:11 14]),
28         nomnod, ...
29         gauss, limitato, disp, []); %#ok<ASGLU>
30
31 bus(:,8:9) = bus1(:,8:9);
32 gen(gen(:, end) == 1, 2:3) = gen1(:, 2:3);
33
34 % Calculate Sbus
35 Pd = bus1(:, 3); % Real power demand (load) in MW
36 Qd = bus1(:, 4); % Reactive power demand (load) in MVar
37
38 Pg = zeros(size(bus1, 1), 1); % Initialize real power
39     generation
40 Qg = zeros(size(bus1, 1), 1); % Initialize reactive power
41     generation
42
43 % Map generator real and reactive power to the
44     corresponding buses
45 genBuses = gen1(:, 1); % Bus numbers where generators are
46     located
47 Pg(genBuses) = gen1(:, 2); % Real power generation in MW
48 Qg(genBuses) = gen1(:, 3); % Reactive power generation in
49     MVar
50
51 % Calculate net power injections
52 Pinj = Pg - Pd; % Net real power injection in MW
53 Qinj = Qg - Qd; % Net reactive power injection in MVar
54
55 % Form the complex power injection matrix Sbus
56 Sbus = (Pinj + 1i * Qinj)/baseMVA;
57
58 % Ensure Y is full
59 Ybus_full = full(Y);
60

```

```
55     % Save Sbus, V, and Y to a .mat file
56     save('power_flow_results.mat', 'Sbus', 'V', 'Ybus_full');
57
58     % I_calculation
59
60     function [Sbus, V, Y] = load_power_flow_results
61     % Load Sbus, V, and Y from the saved .mat file
62     load('power_flow_results.mat', 'Sbus', 'V', 'Ybus_full');
63     end
64
65     % Calculate the conjugate of Sbus
66     conj_Sbus = conj(Sbus);
67     % Calculate the conjugate of V
68     conj_V = conj(V);
69     % Perform element-wise division for current calculation
70     I = conj_Sbus ./ conj_V;
71
72
73
74     % Create the graph
75     GNetB = graph(branch(:, 1), branch(:, 2));
76
77     % Load the boundary nodes
78     load('boundary_nodes.mat');
79
80     % Load the known internal nodes adjacent to boundary nodes
81     load('known_internal_nodes.mat');
82
83     % Ensure boundary nodes and known internal nodes are row
84     % vectors
85     boundary_nodes = boundary_nodes(:)';
86     known_internal_nodes = known_internal_nodes(:)';
87
88     % Initialize the list of internal nodes with the known
89     % internal nodes
90     internal_nodes = known_internal_nodes;
```

```
90 % Initialize the list of nodes to check (starting with the
    known internal nodes)
91 nodes_to_check = known_internal_nodes;
92
93 % Use a set to keep track of visited nodes to avoid
    reprocessing
94 visited_nodes = false(numnodes(GNetB), 1);
95
96 % Mark the known internal nodes as visited
97 visited_nodes(known_internal_nodes) = true;
98
99 % Iteratively find all internal nodes without crossing
    boundary nodes
100 while ~isempty(nodes_to_check)
101     % Get the next node to check
102     current_node = nodes_to_check(1);
103     nodes_to_check(1) = []; % Remove it from the list
104
105     % Find neighbors of the current node
106     neighbors_current = neighbors(GNetB, current_node);
107
108     % Filter out boundary nodes and already identified
        internal nodes
109     for neighbor = neighbors_current'
110         if ~ismember(neighbor, boundary_nodes) && ~
            visited_nodes(neighbor)
111             % Mark this neighbor as visited
112             visited_nodes(neighbor) = true;
113
114             % Add to the list of internal nodes
115             internal_nodes = [internal_nodes, neighbor]; %
                Add as row vector
116
117             % Add to the list of nodes to check
118             nodes_to_check = [nodes_to_check, neighbor]; %
                Add as row vector
119         end
    end
```

```
120     end
121 end
122
123 % Reorder nodes based on their appearance in the bus matrix
124 % Create logical indices for the bus matrix
125 internal_mask = ismember(bus(:, 1), internal_nodes);
126 boundary_mask = ismember(bus(:, 1), boundary_nodes);
127
128 % Extract nodes in the order they appear in the bus matrix
129 internal_nodes = bus(internal_mask, 1)';
130 boundary_nodes = bus(boundary_mask, 1)';
131
132 % Combine internal and boundary nodes into required nodes
133 required_nodes = [internal_nodes, boundary_nodes];
134
135 % Identify external nodes as those nodes that are not in
136 % required nodes
137 all_nodes = 1:numnodes(GNetB);
138 external_nodes = setdiff(all_nodes, required_nodes);
139
140 % Ensure external nodes are row vectors
141 external_nodes = external_nodes(:)';
142
143 % Reorder external nodes based on their appearance in the bus
144 % matrix
145 external_mask = ismember(bus(:, 1), external_nodes);
146 external_nodes = bus(external_mask, 1)';
147
148 % Save internal, boundary, and external nodes to workspace
149 % variables or files if needed
150 save('internal_nodes.mat', 'internal_nodes');
151 save('boundary_nodes.mat', 'boundary_nodes');
152 save('external_nodes.mat', 'external_nodes');
153
154 % Visualize the graph with different colors for internal and
155 % external nodes
156 figure; % Create a new figure window
```

```
153 p = plot(GNetB, 'Layout', 'force'); % 'force' layout
154
155 % Customize node appearance
156 highlight(p, internal_nodes, 'NodeColor', 'green'); %
    Internal nodes in green
157 highlight(p, boundary_nodes, 'NodeColor', 'blue'); % Boundary
    nodes in blue
158 highlight(p, external_nodes, 'NodeColor', 'red'); % External
    nodes in red
159
160 title('Graph with Internal, Boundary, and External Nodes');
161 xlabel('X-axis');
162 ylabel('Y-axis');
163
164 % load the internal and boundary nodes
165 load('internal_nodes.mat');
166 load('boundary_nodes.mat');
167
168 % Create the new order: required nodes first, then external
    nodes
169 new_order = [required_nodes, external_nodes];
170
171 % Reorder the Ybus matrix
172 Ybus_reordered = Ybus_full(new_order, new_order);
173
174 % Number of required nodes
175 num_required = length(required_nodes);
176
177 % Create the Yrr matrix (required-required nodes)
178 Yrr = Ybus_reordered(1:num_required, 1:num_required);
179
180 % Create the Yre matrix (required-external nodes)
181 Yre = Ybus_reordered(1:num_required, num_required+1:end);
182
183 % Create the Yer matrix (external-required nodes)
184 Yer = Ybus_reordered(num_required+1:end, 1:num_required);
185
```

```
186 % Create the Yee matrix (external-external nodes)
187 Yee = Ybus_reordered(num_required+1:end, num_required+1:end);
188
189 % Create the Ir matrix (currents at required nodes)
190 Ir = I(required_nodes, 1);
191
192 % Create the Ie matrix (currents at external nodes)
193 Ie = I(external_nodes, 1);
194
195 % Create the Vr matrix (voltages at required nodes)
196 Vr = V(required_nodes, 1);
197
198 % Create the Ve matrix (voltages at external nodes)
199 Ve = V(external_nodes, 1);
200
201 % Perform Ward's reduction
202
203 % Create the new Yrr matrix after reduction
204 Yrr_new = Yrr - (Yre * inv(Yee) * Yer);
205
206 % Create the new Ir matrix after reduction
207 Ir_new = Ir - (Yre * inv(Yee) * Ie);
208
209 % Map the boundary and internal nodes indices to new Yrr_new
    matrix
210 [~, idx_boundary] = ismember(boundary_nodes, required_nodes);
211 [~, idx_internal] = ismember(internal_nodes, required_nodes);
212
213 % Create Ybb_new matrix after the reduction
214 Ybb_new = Yrr_new(idx_boundary, idx_boundary);
215
216 % Create Yib_new matrix after the reduction
217 Yib_new = Yrr_new(idx_internal, idx_boundary);
218
219 % Create Ybi_new matrix after the reduction
220 Ybi_new = Yrr_new(idx_boundary, idx_internal);
221
```

```

222 % Create Yii_new matrix after the reduction
223 Yii_new = Yrr_new(idx_internal, idx_internal);
224
225 % Create Ii_new matrix after the reduction
226 Ii_new = Ir_new(idx_internal);
227
228 % Create Ib_new matrix after the reduction
229 Ib_new = Ir_new(idx_boundary);
230
231 conj_Ir_new = conj(Ir_new);
232
233 Sbus_r_new = Vr .* conj_Ir_new;
234
235 % Building and updating branch_new matrix
236
237 % Logical index for branches that only connect internal and
    % boundary nodes
238 keep_branches = ismember(branch(:, 1), required_nodes) & ...
239                 ismember(branch(:, 2), required_nodes);
240
241 % Create a new_branch matrix with only internal and boundary
    % nodes
242 branch_new = branch(keep_branches, :);
243
244 % Check if the boundary nodes are more than two
245 if length(boundary_nodes) > 2
246     % Find all possible boundary branches
247     boundary_branches = [];
248     for i = 1:length(boundary_nodes)
249         for j = i+1:length(boundary_nodes)
250             % Check if the connection exists in Ybb_new
251             if abs(Ybb_new(i, j)) > 0
252                 boundary_branches = [boundary_branches;
253                                     boundary_nodes(i), boundary_nodes(j)];
254             end
255         end
256     end

```

```
256     % Define the new connection string for boundary
        branches
257     new_connections = strcat('Boundary_branch_', string(
        boundary_branches(:, 1)), '-', string(
        boundary_branches(:, 2)));
258
259     % Append the new connections to the nombranch cell array
260     nombranch = [nombranch; cellstr(new_connections)];
261     end
262 else
263     % Directly use the provided boundary nodes
264     boundary_branches = [boundary_nodes];
265 end
266     % Define the new connection string for boundary branches
267     new_connections = strcat('Boundary_branch_', string(
        boundary_branches(:, 1)), '-', string(
        boundary_branches(:, 2)));
268
269
270
271 % Update the branch and branchset matrices
272 for i = 1:size(boundary_branches, 1)
273     node1 = boundary_branches(i, 1);
274     node2 = boundary_branches(i, 2);
275
276     % Check if branch exists in the branch matrix
277     branch_exists = any((branch_new(:, 1) == node1 &
        branch_new(:, 2) == node2) | ...
278         (branch_new(:, 1) == node2 &
        branch_new(:, 2) == node1));
279
280     % If the branch does not exist, add it to both branch and
        branchset matrices
281     if ~branch_exists
282         % Add to branch matrix
283         new_branch = [node1, node2, 0, 0, 0, 0, 9999, 9999,
            9999, 0, 0, 0, 0, 1];
```

```
284     branch_new = [branch_new; new_branch];
285
286     % Add to branchset matrix
287     new_branchset = [node1, node2, 1];
288     branchset = [branchset; new_branchset];
289     end
290 end
291
292 % Calculating R & X for boundary branches
293
294 % Get the size of the Ybb_new matrix
295 n = size(Ybb_new, 1);
296
297 % Initialize matrices to store R and X
298 R_matrix = zeros(n, n);
299 X_matrix = zeros(n, n);
300
301 for i = 1:n
302     for j = 1:n
303         if i ~= j
304             Yij = Ybb_new(i, j);
305             Gij = real(Yij);
306             Bij = imag(Yij);
307
308             denominator = Gij^2 + Bij^2;
309             Rij = -Gij / denominator;
310             Xij = Bij / denominator;
311
312             R_matrix(i, j) = Rij;
313             X_matrix(i, j) = Xij;
314         end
315     end
316 end
317
318 % Updating R & X values in the boundary branches
319
320 % Number of branches in the new branch matrix
```

```
321 num_branches = size(branch_new, 1);
322
323 % Update the R and X values in the new_branch matrix for
    boundary branches
324 for k = 1:num_branches
325     from_node = branch_new(k, 1);
326     to_node = branch_new(k, 2);
327
328     if ismember(from_node, boundary_nodes) && ismember(
        to_node, boundary_nodes)
329         % Find the indices in boundary nodes
330         from_index = find(boundary_nodes == from_node);
331         to_index = find(boundary_nodes == to_node);
332
333         % Update R value
334         branch_new(k, 3) = R_matrix(from_index, to_index);
335
336         % Update X value
337         branch_new(k, 4) = X_matrix(from_index, to_index);
338     end
339 end
340
341
342 % Update the Bus matrix
343 Bus_new = bus(required_nodes, :);
344
345 % Update Column 3 & 4 of Bus_new with updated Pd_new & Qd_new
346 conj_Ir_new = conj(Ir_new);
347
348 Sbus_new_MVA = Vr .* conj_Ir_new * baseMVA;
349
350 Pd_new = real(Sbus_new_MVA);
351 Qd_new = imag(Sbus_new_MVA);
352
353 % Initialize demand arrays
354 real_demand = zeros(size(Pd_new));
355 reactive_demand = zeros(size(Qd_new));
```

```
356
357 % Calculate demand for nodes with negative real power or
      boundary nodes
358 for i = 1:length(Pd_new)
359     node_idx = required_nodes(i); % Map i to the actual node
      index
360     if ismember(node_idx, boundary_nodes) % Check if it is a
      boundary node
361         if Pd_new(i) >= 0
362             real_demand(i) = - Pd_new(i); % Update to
      negative value
363             reactive_demand(i) = - Qd_new(i); % Update to
      negative value
364         else
365             real_demand(i) = - Pd_new(i); % Retain original
      (negative) values
366             reactive_demand(i) = - Qd_new(i); % Retain
      original (negative) values
367         end
368     elseif Pd_new(i) < 0
369         real_demand(i) = -Pd_new(i); % Convert to positive
      demand
370         reactive_demand(i) = -Qd_new(i); % Convert to
      positive demand
371     else
372         real_demand(i) = 0;
373         reactive_demand(i) = 0;
374     end
375 end
376
377 % Update Column 3 & 4 of Bus_new with updated real_demand &
      reactive_demand
378 Bus_new(:,3) = real_demand;
379 Bus_new(:,4) = reactive_demand;
380
381 % Calculate the magnitude of the voltage
382 V_magnitude = abs(Vr);
```

```
383
384 % Calculate the angle of the voltage in radians
385 V_angle_radians = angle(Vr);
386
387 % Convert the angle to degrees
388 V_angle_degrees = rad2deg(V_angle_radians);
389
390 % Update Column 8 & 9 of Bus_new with updated V_magnitude &
    V_angle_degree
391 Bus_new(:,8) = V_magnitude;
392 Bus_new(:,9) = V_angle_degrees;
393
394
395 % Shunt susceptance calculation and updating to Bus_new
    matrix
396
397 % Define the number of buses
398 num_buses = size(Bus_new, 1); % Total number of buses
399
400 % Initialize arrays for self-admittance and connected
    admittance per bus
401 self_admittance = zeros(num_buses, 1);
402 connected_admittance = zeros(num_buses, 1);
403
404 % Initialize arrays for line charging susceptance and
    conductance
405 line_charging_admittance = zeros(num_buses, 1);
406
407 % Calculate total self-admittance and total connected
    admittance per bus
408 for i = 1:num_buses
409     self_admittance(i) = Yrr_new(i, i); % Total admittance at
        the bus (diagonal of Yrr_new)
410     for j = 1:num_buses
411         if i ~= j
412             connected_admittance(i) = connected_admittance(i)
                - Yrr_new(i, j);
```

```

413         end
414     end
415 end
416
417 % Initialize matrix for Yij for connected nodes
418 Yij_connected = zeros(num_buses, num_buses);
419
420 % Process the branch matrix to sum all Bij and Gij values for
421 % connected buses
422 for k = 1:size(branch_new, 1)
423     from_bus = branch_new(k, 1);
424     to_bus = branch_new(k, 2);
425     Bij = branch_new(k, 5); % Line charging susceptance
426     Gij = branch_new(k, 6); % Line charging conductance
427
428     % Line charging admittance (complex)
429     Yij = Gij + 1i * Bij;
430
431     % Find indices of the from and to buses in the Bus_new
432     % matrix
433     from_index = find(Bus_new(:, 1) == from_bus);
434     to_index = find(Bus_new(:, 1) == to_bus);
435
436     % Sum line charging admittance for each connected bus
437     if ~isempty(from_index) && ~isempty(to_index)
438         Yij_connected(from_index, to_index) = Yij_connected(
439             from_index, to_index) + Yij;
440         Yij_connected(to_index, from_index) = Yij_connected(
441             to_index, from_index) + Yij;
442     end
443 end
444
445 % Sum line charging admittance for each bus by summing each
446 % row of Yij_connected
447 line_charging_admittance = sum(Yij_connected, 2) / 2;
448
449 % Calculate shunt admittance for each bus using the given

```

```

    formula
445 shunt_admittance = zeros(num_buses, 1);
446 for i = 1:num_buses
447
448     % Calculate the shunt admittance for the current bus
449     shunt_admittance(i) = self_admittance(i) -
        connected_admittance(i) - line_charging_admittance(i);
450 end
451
452 % Extract the conductance (Gc) and susceptance (Bs) from the
    shunt admittance
453 Gs = real(shunt_admittance) * baseMVA;
454 bs = imag(shunt_admittance) * baseMVA;
455
456 % Initialize an array to hold the updated shunt susceptance
    for Bus_new
457 updated_shunt_susceptance = zeros(num_buses, 1);
458 updated_shunt_conductance = zeros(num_buses, 1);
459
460 % Update susceptance in Bus_new only for boundary nodes
461 for i = 1:length(boundary_nodes)
462     bus = boundary_nodes(i);
463     bus_index = find(Bus_new(:, 1) == bus);
464     if ~isempty(bus_index)
465         updated_shunt_susceptance(bus_index) = bs(bus_index);
466         updated_shunt_conductance(bus_index) = Gs(bus_index);
467     end
468 end
469 % Updating Bus_new matrix with the updated shunt susceptance
    values
470 Bus_new(:, 6) = updated_shunt_susceptance;
471 Bus_new(:, 5) = updated_shunt_conductance;
472
473 % Map the gen matrix to nomgen
474 gen_node_ids = gen(:, 1);
475 nomgen_map = containers.Map('KeyType', 'double', 'ValueType',
    'char');
```

```

476 for i = 1:length(gen_node_ids)
477     if i <= length(nomgen)
478         nomgen_map(gen_node_ids(i)) = nomgen{i};
479     end
480 end
481
482 % Filter Gen_new to include only generators at internal nodes
483 gen_indices = ismember(gen(:, 1), required_nodes);
484 Gen_new = gen(gen_indices, :);
485
486 % Create a map from node IDs to Sbus_new indices for
487     alignment
488 node_to_index_sbus = containers.Map(required_nodes, 1:length(
489     required_nodes));
490
491 % Initialize arrays to hold new generation values
492 real_power = zeros(size(Gen_new, 1), 1);
493 reactive_power = zeros(size(Gen_new, 1), 1);
494
495 % Iterate over the filtered Gen_new to update values
496 for i = 1:size(Gen_new, 1)
497     node_id = Gen_new(i, 1);
498     if node_to_index_sbus.isKey(node_id)
499         sbus_index = node_to_index_sbus(node_id);
500         real_power(i) = real(Sbus_new_MVA(sbus_index));
501         reactive_power(i) = imag(Sbus_new_MVA(sbus_index));
502     else
503         % Handle appropriately if no Sbus data
504         real_power(i) = 0;
505         reactive_power(i) = 0;
506     end
507 end
508
509 % Update Gen_new matrix
510 Gen_new(:, 2) = real_power; % Update real power in column 2
511 Gen_new(:, 3) = reactive_power; % Update reactive power in
512     column 3

```

```
510
511 % Convert keys of node_to_index_sbus to numeric array for
      comparison
512 numeric_keys = cell2mat(keys(node_to_index_sbus));
513
514 % Get the node IDs already in Gen_new
515 existing_gen_node_ids = Gen_new(:, 1);
516
517 % Create nomgen_new to include only entries that match nodes
      in Gen_new
518 nomgen_new = cell(size(Gen_new, 1), 1);
519 for i = 1:size(Gen_new, 1)
520     node_id = Gen_new(i, 1);
521     if nomgen_map.isKey(node_id)
522         nomgen_new{i} = nomgen_map(node_id);
523     end
524 end
525
526 % Check if there is a slack bus in Bus_new matrix
527 slack_bus_present = any(Bus_new(:, 2) == 3);
528
529 % If no slack bus is present, assign a new slack bus
530 if ~slack_bus_present
531     % Criteria for selecting new slack bus (e.g., highest
      capacity)
532
533     % Find the indices of generator buses in Bus_new
534     gen_bus_indices = ismember(Bus_new(:, 1), Gen_new(:, 1));
535
536     % Extract the Pmax of these generator buses
537     gen_bus_Pmax = Gen_new(gen_bus_indices, 9);
538
539     % Find the index of the generator bus with the highest
      Pmax
540     [~, max_Pmax_index] = max(gen_bus_Pmax);
541
542     % Map this index back to the original Bus_new array
```

```

543     gen_bus_numbers = Bus_new(gen_bus_indices, 1); % Get the
           bus numbers of generator buses
544     new_slack_bus = gen_bus_numbers(max_Pmax_index);
545
546     % Update the second column to 3 to indicate slack bus
547     Bus_new(Bus_new(:, 1) == new_slack_bus, 2) = 3;
548 end
549
550 % Required modifications to make the Reduced network mat file
551
552 % Filter br_ph_shifters to include only branches between
           required nodes
553
554 % Initialize the new phase shifters matrix with the same size
           as branch_new
555 br_ph_shifters_new = zeros(size(branch_new, 1), 1);
556
557 % Identify the external branch connected to external nodes
           without a phase shifter
558 exclude_index = [];
559
560 for i = 1:size(branch, 1)
561     if branch(i, 10) == 0 && (ismember(branch(i, 1),
           external_nodes) || ismember(branch(i, 2),
           external_nodes))
562         exclude_index = i;
563         break;
564     end
565 if ~isempty(exclude_index)
566     % Create branch_1 matrix excluding the selected branch
567     branch_1 = branch;
568     branch_1(exclude_index, :) = [];
569 end
570 % Find matching indices in the filtered network for each
           branch in branch_new
571     for i = 1:size(branch_new, 1)
572         for j = 1:size(branch_1, 1)

```

```
573         if isequal(branch_new(i, 1:2), branch_1(j, 1:2))
574             % If a match is found and nodes are required
575             % nodes, copy the phase shifter data
576             br_ph_shifters_new(i, :) = br_ph_shifters(j,
577                 :);
578             break; % Stop searching once a match is found
579         end
580     end
581 end
582 % Initialize the new constrained status vector
583 branch_N1_constrained_new = false(size(branch_new, 1), 1);
584
585 % Find matching indices in the full network for each branch
586 % in Branch_new
587 for i = 1:size(branch_new, 1)
588     for j = 1:size(branch, 1)
589         if isequal(branch_new(i, 1:2), branch(j, 1:2))
590             % If a match is found, copy the constrained
591             % status
592             branch_N1_constrained_new(i) =
593                 branch_N1_constrained(j);
594             break; % Stop searching once a match is found
595         end
596     end
597 end
598
599 % Initialize the new not constrained status vector for
600 % reduced network
601 branch_notN_constrained_new = false(size(branch_new, 1), 1);
602
603 % Find matching indices in the full network for each branch
604 % in Branch_new
605 for i = 1:size(branch_new, 1)
606     for j = 1:size(branch, 1)
607         if isequal(branch_new(i, 1:2), branch(j, 1:2))
```

```

603         % Copy the not constrained status if nodes match
604         branch_notN_constrained_new(i) =
605             branch_notN_constrained(j);
606         break; % Stop searching once a match is found
607     end
608 end
609
610 %GAMS_data_gen_bid_new
611 GAMS_data_gen_bid_new = GAMS_data_gen_bid(ismember(
612     GAMS_data_gen_bid(:, 1), Gen_new(:,1)), :);
613
614 %GAMS_data_Supply_bids_new
615 GAMS_data_Supply_bids_new = GAMS_data_Supply_bids(ismember(
616     GAMS_data_Supply_bids(:, 1), Gen_new(:,1)), :);
617
618 %gens_fake_new
619 gens_fake_new = gens_fake(1:size(Gen_new(:,1)), :);
620
621 CBCO_new = [1 0];
622 CBusCO = [];
623 GAMS_data_pBid_gen = [1];
624
625 % nomnod_new
626 nomnod_new = nomnod(required_nodes, :);
627
628 % Initialize branchset_new
629 branchset_new = [];
630
631 % Loop through each branch in Branch_new
632 for i = 1:size(branch_new, 1)
633     node1 = branch_new(i, 1);
634     node2 = branch_new(i, 2);
635
636     % Find all matches in the full network branchset
637     for j = 1:size(branchset, 1)
638         if branchset(j, 1) == node1 && branchset(j, 2) ==

```

```
node2
637     % Prepare a potential new entry with the same
        structure as branchset_new
638     potential_new_entry = [node1, node2, branchset(j,
        3)];
639
640     % Check if this potential new entry is already in
        branchset_new
641     if isempty(branchset_new) || ~ismember(
        potential_new_entry, branchset_new, 'rows')
642         branchset_new = [branchset_new;
            potential_new_entry];
643     end
644 end
645 end
646 end
647
648 % Extract the first column data from nodInfo as an array
649 columnData = nodInfo(:, 1);
650
651 % Find the rows in nodInfo where the first column is a member
        of internal_nodes
652 isMember = ismember(columnData, required_nodes);
653
654 % Filter nodInfo to keep only rows where the first column
        matches internal_nodes
655 nodInfo_new = nodInfo(isMember, :);
656
657 % Extract node pairs from the original and reduced matrices
658 fullPairs = branch(:, 1:2); % Extracting 'From' and 'To'
        nodes
659 reducedPairs = branch_new(:, 1:2);
660
661 % Initialize a cell array to store nombranch labels for the
        reduced network
662 newNomBranch = cell(size(reducedPairs, 1), 1);
663
```

```

664 % Loop through each pair in the reduced network
665 for i = 1:size(reducedPairs, 1)
666     % Find the index in the full network that matches the
        reduced pair
667     rowIndex = find(ismember(fullPairs, reducedPairs(i, :), '
        rows'), 1, 'first');
668
669     % If a match is found, retrieve the nombranch label
670     if ~isempty(rowIndex)
671         newNomBranch{i} = nombranch{rowIndex};
672     else
673         newNomBranch{i} = new_connections; % Indicate
        unmatched cases with boundary branch
674
675     end
676 end
677
678 % Save the required matrices and arrays
679 outputFileName1 = 'Reduced_Network.mat';
680 outputFileName2 = 'Reduced_NW_Results.mat';
681 outputFileName3 = 'Final_Reduced_network.mat';
682
683 save(outputFileName1, 'baseMVA', 'Bus_new', 'Gen_new', '
        branch_new', 'br_ph_shifters_new', '
        branch_N1_constrained_new', ...
684     'branch_notN_constrained_new', 'GAMS_data_gen_bid_new', '
        GAMS_data_Supply_bids_new', ...
685     'gens_fake_new', 'CBCO_new', 'CBusCO', 'GAMS_data_pBid_gen'
        , 'nomgen_new', 'nomnod_new', 'branchset_new', '
        nodInfo_new', 'newNomBranch');
686
687 save(outputFileName2, 'Yrr_new', 'Ybb_new', 'Yib_new', '
        Ybi_new', 'Yii_new', 'Ii_new', 'Ib_new', 'Vr', 'Sbus_new_MVA
        ', 'Ir_new', 'Ybus_full', 'Sbus', 'V');
688
689 % Rename the matrix by assigning it to a new variable
690 br_ph_shifters = br_ph_shifters_new;

```

```
691 branch_N1_constrained = branch_N1_constrained_new;
692 branch = branch_new;
693 branch_notN_constrained = branch_notN_constrained_new;
694 branchset = branchset_new;
695 bus = Bus_new;
696 CBCO = CBCO_new;
697 GAMS_data_gen_bid = GAMS_data_gen_bid_new;
698 GAMS_data_Supply_bids = GAMS_data_Supply_bids_new;
699 gen = Gen_new;
700 gens_fake = gens_fake_new;
701 nombranch = newNomBranch;
702 nodInfo = nodInfo_new;
703 nomgen = nomgen_new;
704 nomnod = nomnod_new;
705
706 save(outputFileName3, 'baseMVA', 'bus', 'gen', 'branch', '
    br_ph_shifters', 'branch_N1_constrained', ...
707     'branch_notN_constrained', 'GAMS_data_gen_bid', '
    GAMS_data_Supply_bids', ...
708     'gens_fake', 'CBCO', 'CBusCO', 'GAMS_data_pBid_gen', '
    nomgen', 'nomnod', 'branchset', 'nodInfo', 'nombranch'
    );
709
710 % Extract original bus numbers
711 original_bus_numbers = bus(:, 1);
712
713 % Create new bus numbers
714 new_bus_numbers = (1:length(original_bus_numbers))';
715
716 % Create a mapping table
717 mapping = [original_bus_numbers, new_bus_numbers];
718
719 % Save mapping file
720 save('mapping.mat', 'mapping');
721
722 % Create copies of matrices to modify
723 new_bus_matrix = bus;
```

```
724 new_branch_matrix = branch;
725 new_gen_matrix = gen;
726 new_branchset = branchset;
727
728 % Create a mapping table for branch renumbering
729 original_branch_indices = (1:size(branch, 1))';
730 new_branch_indices = zeros(size(branch, 1), 1);
731
732 % Initialize a table to store the mapping of original and new
    'From' and 'To' nodes
733 branch_mapping = zeros(size(branch, 1), 4); % Columns: [
    Original From, Original To, New From, New To]
734
735 % Update bus numbers in the bus matrix
736 for i = 1:size(mapping, 1)
737     original_bus_idx = bus(:, 1) == mapping(i, 1);
738     new_bus_matrix(original_bus_idx, 1) = mapping(i, 2);
739 end
740
741 % Update 'from' and 'to' bus numbers in the branch matrix and
    record the mapping
742 for i = 1:size(branch, 1)
743     % Record original 'From' and 'To' nodes
744     branch_mapping(i, 1:2) = branch(i, 1:2);
745
746     % Update 'From' bus number
747     from_bus_idx = branch(i, 1) == mapping(:, 1);
748     if any(from_bus_idx)
749         new_branch_matrix(i, 1) = mapping(from_bus_idx, 2);
750     end
751
752     % Update 'To' bus number
753     to_bus_idx = branch(i, 2) == mapping(:, 1);
754     if any(to_bus_idx)
755         new_branch_matrix(i, 2) = mapping(to_bus_idx, 2);
756     end
757
```

```
758     % Record new 'From' and 'To' nodes
759     branch_mapping(i, 3:4) = new_branch_matrix(i, 1:2);
760 end
761
762 % Update generator bus numbers in the gen matrix
763 for i = 1:size(mapping, 1)
764     gen_bus_idx = gen(:, 1) == mapping(i, 1);
765     new_gen_matrix(gen_bus_idx, 1) = mapping(i, 2);
766 end
767
768 % Update 'from' and 'to' bus numbers in the branchset matrix
769 for i = 1:size(mapping, 1)
770     from_bus_idx = branchset(:, 1) == mapping(i, 1);
771     to_bus_idx = branchset(:, 2) == mapping(i, 1);
772     new_branchset(from_bus_idx, 1) = mapping(i, 2);
773     new_branchset(to_bus_idx, 2) = mapping(i, 2);
774 end
775
776 % Save the branch mapping table to a .mat file
777 save('branch_mapping.mat', 'branch_mapping');
778
779
780 % Create a table from the array
781 mappingTable = array2table(mapping, 'VariableNames', {'
782     Original', 'New'});
783
784 % Update bus numbers in the nodInfo matrix
785 for i = 1:height(nodInfo)
786     % Find the new bus number from mappingTable corresponding
787     % to the current bus number
788     currentBusNumber = nodInfo{i, 1};
789     mappedIndex = find(mappingTable.Original ==
790         currentBusNumber);
791     if ~isempty(mappedIndex)
792         nodInfo{i, 1} = mappingTable.New(mappedIndex);
793     end
794 end
795 end
```

```

792
793
794 % Rename the updated matrices to original names
795 bus = new_bus_matrix;
796 branch = new_branch_matrix;
797 gen = new_gen_matrix;
798 branchset = new_branchset;
799
800 % Save the updated matrices
801 save(outputFileName3, 'baseMVA', 'bus', 'gen', 'branch', '
      br_ph_shifters', 'branch_N1_constrained', ...
802      'branch_notN_constrained', 'GAMS_data_gen_bid', '
      GAMS_data_Supply_bids', ...
803      'gens_fake', 'CBCO', 'CBusCO', 'GAMS_data_pBid_gen', '
      nomgen', 'nomnod', 'branchset', 'nodInfo', 'nombranch'
      );
804 end

```

Listing A.1: Full MATLAB Code for Network Reduction

• MATLAB code for N-1 Contingencies Simulation of the both Networks:

```

1 % Load the mat file
2 load('Rueda_network_3_1.mat');
3
4 % Extract unique nodes
5 nodes = unique(branch(:, 1:2));
6 numNodes = max(nodes);
7
8 % Create graph from branch data
9 G = graph(branch(:, 1), branch(:, 2));
10
11 % Plot the original graph
12 figure;
13 h = plot(G, 'Layout', 'force');
14 title('Original Network');
15
16 % Function to check if the graph is connected

```

```
17 isConnected = @(G) all(conncomp(G) == 1);
18
19 % Initialize arrays to keep track of radial and non-radial
    branches
20 nonRadialEdges = [];
21 radialEdges = [];
22
23 % Iterate over each branch to test its removal
24 for i = 1:size(branch, 1)
25     % Remove the branch
26     tempG = rmedge(G, branch(i, 1), branch(i, 2));
27
28     % Check if the graph is still connected
29     if ~isConnected(tempG)
30         % If not connected, mark this branch as radial
31         radialEdges = [radialEdges; branch(i, 1), branch(i,
32             2)];
33     else
34         % If connected, mark this branch as non-radial
35         nonRadialEdges = [nonRadialEdges; branch(i, 1),
36             branch(i, 2)];
37     end
38 end
39
40 % Highlight radial branches in red
41 highlight(h, radialEdges(:, 1), radialEdges(:, 2), 'EdgeColor
42     ', 'r', 'LineWidth', 2);
43
44 % Save the non-radial branch matrix
45 nonRadialBranchMatrix = nonRadialEdges;
46
47 % Display the result
48 title('Network with Radial Branches Highlighted');
49
50 % Initialize a matrix to keep track of the contingency
    mapping
51 contingencyMapping = [];
```

```

49
50 % Perform N-1 contingency analysis on non-radial branches
51 for i = 1:size(nonRadialBranchMatrix, 1)
52     % Remove the branch for contingency
53     contingencyBranch = nonRadialBranchMatrix(i, :);
54     tempBranch = branch;
55     idx = find(branch(:, 1) == contingencyBranch(1) & branch
56         (:, 2) == contingencyBranch(2));
57     tempBranch(idx, :) = [];
58
59 % Perform power flow analysis with the modified branch
60     data
61     launch_PF(baseMVA, bus, tempBranch, gen, nomgen, nomnod,
62         i);
63
64 % Update the contingency mapping
65     contingencyMapping = [contingencyMapping; i,
66         contingencyBranch];
67 end
68
69 % Save the contingency mapping to a .mat file
70 save('contingency_mapping_full.mat', 'contingencyMapping');
71
72 % Define the local function to perform power flow and save
73     results
74 function launch_PF(baseMVA, bus, branch, gen, nomgen, nomnod,
75     contingencyIndex)
76     % Adjust generator voltage control
77     genVctrl = gen(:, end);
78     [bus, gen] = adjust_genQctrl(bus, gen, genVctrl);
79
80 % Perform power flow analysis
81     gauss = 0;
82     limitato = 1;
83     disp = 0;
84
85     addpath("/Users/parthasmacbook/Desktop/Thesis/Final/

```

```

    Comparison_1st_case/N-1_simulation_full/Load_flow");
80 [busPF, genPF, ~] = gen_Vconst_cut(bus, gen, nomgen);
81 [bus1, gen1, success, Yf, Yt, Y, V] = load_flow(baseMVA,
    busPF, genPF, branch(:, [1:11 14]), nomnod, gauss,
    limitato, disp, []);
82
83 % Update bus and generator data
84 bus(:, 8:9) = bus1(:, 8:9);
85 gen(gen(:, end) == 1, 2:3) = gen1(:, 2:3);
86
87 % Calculate Sbus
88 Pd = bus1(:, 3); % Real power demand (load) in MW
89 Qd = bus1(:, 4); % Reactive power demand (load) in MVar
90
91 Pg = zeros(size(bus1, 1), 1); % Initialize real power
    generation
92 Qg = zeros(size(bus1, 1), 1); % Initialize reactive power
    generation
93
94 % Map generator real and reactive power to the
    corresponding buses
95 genBuses = gen1(:, 1); % Bus numbers where generators are
    located
96 Pg(genBuses) = gen1(:, 2); % Real power generation in MW
97 Qg(genBuses) = gen1(:, 3); % Reactive power generation in
    MVar
98
99 % Calculate net power injections
100 Pinj = Pg - Pd; % Net real power injection in MW
101 Qinj = Qg - Qd; % Net reactive power injection in MVar
102
103 % Form the complex power injection matrix Sbus
104 Sbus = (Pinj + 1i * Qinj) / baseMVA;
105
106 % Ensure Y is full
107 Ybus_full = full(Y);
108

```

```

109 % Calculate bus currents
110 conj_Sbus = conj(Sbus);
111 conj_V = conj(V);
112 I = conj_Sbus ./ conj_V; % Bus current calculation
113
114 % Calculate branch currents
115 I_f = Yf * V; % Current from the 'from' bus
116 I_t = Yt * V; % Current from the 'to' bus
117
118 % Calculate branch current magnitude for each branch in
    per unit (PU)
119 I_f_mag = abs(I_f); % Magnitude of current from the "
    from" bus
120 I_t_mag = abs(I_t); % Magnitude of current from the "to"
    bus
121
122 % Take the average of the two to get branch current
    magnitude in per unit (PU)
123 I_branch_mag_pu = (I_f_mag + I_t_mag) / 2;
124
125 % Convert per-unit branch currents to amperes
126 from_buses = branch(:, 1);
127 to_buses = branch(:, 2);
128
129 % Get base voltages from the bus matrix (10th column)
130 V_base_from = bus(from_buses, 10); % Base voltage in kV
    for 'from' buses
131 V_base_to = bus(to_buses, 10); % Base voltage in kV
    for 'to' buses
132
133 % Calculate base currents using baseMVA and base voltage
    (convert kV to V)
134 I_base_from = (baseMVA * 1e6) ./ (sqrt(3) * V_base_from *
    1e3); % Base current in amperes
135 I_base_to = (baseMVA * 1e6) ./ (sqrt(3) * V_base_to * 1e3
    ); % Base current in amperes
136

```

```

137 % Convert per-unit branch current to amperes
138 I_branch_mag_amps_from = I_branch_mag_pu .* I_base_from;
      % From bus currents in amperes
139 I_branch_mag_amps_to = I_branch_mag_pu .* I_base_to;
      % To bus currents in amperes
140
141 % Take the average of the two to get branch current in
      amperes
142 I_branch_mag_amps = (I_branch_mag_amps_from +
      I_branch_mag_amps_to) / 2;
143
144 % Extract the 'from' and 'to' buses and the 7th column
      from the original branch data
145 max_current_ampere = branch(:, 7); % 7th column (assumed
      to be in amperes)
146
147 % Combine the branch data with the calculated branch
      current magnitude in amperes
148 branch_current_data = [from_buses, to_buses,
      max_current_ampere, I_branch_mag_amps];
149
150 % Save results to a .mat file for each contingency,
      including the branch current data
151 save(sprintf('power_flow_results_contingency_%d.mat',
      contingencyIndex), ...
152      'Sbus', 'V', 'Ybus_full', 'I', 'I_f', 'I_t', '
      I_f_mag', 'I_t_mag', 'I_branch_mag_pu', '
      I_branch_mag_amps', 'bus', 'gen', 'success', 'Yf'
      , 'Yt', 'Y', 'branch_current_data');
153 end

```

**Listing A.2:** MATLAB code for N-1 Contingencies Simulation for both Networks

- MATLAB code for Contingency Mapping & Recovering original indices for Reduced Network's result:

```

1 load('contingency_mapping.mat');
2 load('branch_mapping_1st.mat');

```

```

3
4 % Initialize result array
5 Combine_contingency_branch_reduced = [];
6
7 % Iterate over each row of contingencyMapping
8 for i = 1:size(contingencyMapping, 1)
9     % Extract values from the current row of
10     contingencyMapping
11     c_row = contingencyMapping(i, :);
12     % Iterate over each row of branch_mapping
13     for j = 1:size(branch_mapping, 1)
14         % Extract values from the current row of
15         branch_mapping
16         b_row = branch_mapping(j, :);
17         % Check if the values match as per the given
18         condition
19         if isequal(c_row(2:3), b_row(3:4))
20             % Check if the combination is already in the
21             result array
22             if isempty(Combine_contingency_branch_reduced) ||
23                 ~ismember([c_row(1), b_row(1:2)],
24                     Combine_contingency_branch_reduced, 'rows')
25                 % Append the combined row to the result array
26                 Combine_contingency_branch_reduced = [
27                     Combine_contingency_branch_reduced; c_row(1),
28                     b_row(1:2)];
29             end
30         end
31     end
32 end
33
34 save("Combine_contingency_branch_reduced", "
35     Combine_contingency_branch_reduced");
36
37 % Load necessary data
38 load('mapping.mat'); % 'mapping' contains two columns: [
39     original, renumbered] bus numbers

```

```
30 load('branch_mapping_1st.mat'); % [original_bus1
    original_bus2 new_bus1 new_bus2]
31 load('Combine_contingency_branch_reduced.mat');
32
33 % Extract the number of contingencies from the first column
    of Combine_contingency_branch_reduced
34 num_contingencies = length(unique(
    Combine_contingency_branch_reduced(:, 1)));
35
36 % Loop through each contingency file
37 for contingencyIndex = 1:num_contingencies
38     % Load the contingency file
39     contingency_file = sprintf('
        power_flow_results_contingency_reduced_%d.mat',
        contingencyIndex);
40     load(contingency_file); % This loads 'branch_current_data
        ', 'bus', 'gen', and other variables
41
42     % ----- Update branch_current_data
        -----
43
44     % Initialize a matrix to store the updated branch current
        data
45     updated_branch_current_data = branch_current_data;
46
47     % Iterate over each row of branch_current_data to map the
        buses
48     for i = 1:size(branch_current_data, 1)
49         % Extract the 'from' and 'to' buses in
            branch_current_data
50         new_from_bus = branch_current_data(i, 1);
51         new_to_bus = branch_current_data(i, 2);
52
53         % Find the corresponding original buses in
            branch_mapping_1st
54         mapping_idx = find(branch_mapping(:, 3) ==
            new_from_bus & branch_mapping(:, 4) == new_to_bus)
```

```

55     ;
56     % If a mapping exists, replace the new buses with the
57     % original ones
58     if ~isempty(mapping_idx)
59         original_from_bus = branch_mapping(mapping_idx,
60             1);
61         original_to_bus = branch_mapping(mapping_idx, 2);
62
63         % Update the branch_current_data with the
64         % original bus numbers
65         updated_branch_current_data(i, 1) =
66             original_from_bus;
67         updated_branch_current_data(i, 2) =
68             original_to_bus;
69     end
70 end
71
72 % ----- Update bus matrix using mapping.mat
73 -----
74
75 % Initialize the updated bus matrix
76 updated_bus = bus;
77
78 % Iterate over the bus matrix and map the bus numbers
79 for i = 1:size(bus, 1)
80     % Find the corresponding original bus number in the
81     % mapping
82     mapping_idx = find(mapping(:, 2) == bus(i, 1));
83
84     % If a mapping exists, replace the renumbered bus
85     % with the original bus number
86     if ~isempty(mapping_idx)
87         updated_bus(i, 1) = mapping(mapping_idx, 1);
88     end
89 end
90 end

```

```

83 % ----- Update gen matrix using mapping.mat
84 % -----
85 % Initialize the updated gen matrix
86 updated_gen = gen;
87
88 % Iterate over the gen matrix and map the bus numbers
89 for i = 1:size(gen, 1)
90     % Find the corresponding original bus number in the
91     % mapping
92     mapping_idx = find(mapping(:, 2) == gen(i, 1));
93
94     % If a mapping exists, replace the renumbered bus
95     % with the original bus number
96     if ~isempty(mapping_idx)
97         updated_gen(i, 1) = mapping(mapping_idx, 1);
98     end
99 end
100
101 % ----- Save the updated matrices
102 % -----
103 % Save the updated branch_current_data, bus, and gen back
104 % into the same contingency file
105 branch_current_data = updated_branch_current_data;
106 bus = updated_bus;
107 gen = updated_gen;
108
109 save(contingency_file, 'Sbus', 'V', 'Ybus_full', 'I', '
110     I_f', 'I_t', 'I_f_mag', 'I_t_mag', ...
111     'I_branch_mag_pu', 'I_branch_mag_amps', 'bus', 'gen',
112     'success', 'Yf', 'Yt', 'Y', 'branch_current_data'
113     );
114
115 % Display a message indicating the completion of the
116 % current contingency
117 fprintf('Contingency %d updated and saved.\n',

```

```

    contingencyIndex);
111 end

```

**Listing A.3:** MATLAB code for Contingency Mapping & Recovering original indices for Reduced Network's result

• MATLAB code for Comparing Power Flow Results for each contingency:

```

1 % Load the required data
2 load('N-1_simulation_full/contingency_mapping_full.mat');
3 load('N-1_simulation_reduced/
4   Combine_contingency_branch_reduced.mat');
5 % Initialize an empty array to store the common branches with
6   contingencies
7 common_contingencies = [];
8 differing_success_full_failed_reduced_succeeded = []; % Full
9   failed, Reduced succeeded
10 differing_success_full_succeeded_reduced_failed = []; % Full
11   succeeded, Reduced failed
12 % Use a nested loop to find common branches in the full
13   network
14 for i = 1:size(Combine_contingency_branch_reduced, 1)
15     branch_reduced = Combine_contingency_branch_reduced(i,
16       2:3);
17     for j = 1:size(contingencyMapping, 1)
18         branch_full = contingencyMapping(j, 2:3);
19         if isequal(branch_reduced, branch_full)
20             % Append the contingency numbers and the common
21               branch to the result
22             common_contingencies = [common_contingencies;
23               contingencyMapping(j, 1),
24               Combine_contingency_branch_reduced(i, 1),
25               branch_full];
26             break;
27         end
28     end
29 end
30 end
31 end

```

```
22
23 % Remove duplicates and ensure proper counting
24 [~, unique_indices] = unique(common_contingencies(:, 3:4), '
    rows');
25 common_contingencies = common_contingencies(unique_indices,
    :);
26
27 % Save the common_contingencies
28 save('common_contingencies', 'common_contingencies');
29
30 % Load bus IDs from Rueda_network_3.mat for the full network
31 full_bus_data = load('N-1_simulation_full/Rueda_network_3_1.
    mat');
32 full_bus_ids = full_bus_data.bus(:,1);
33 branch_full = full_bus_data.branch(:,1:2); % Assuming the
    branch data is in the first two columns
34
35 % Load bus IDs from Reduced_Network.mat for the reduced
    network
36 reduced_bus_data = load('N-1_simulation_reduced/
    Reduced_Network.mat');
37 reduced_bus_ids = reduced_bus_data.Bus_new(:,1);
38
39 % Initialize arrays to store the differences and
    contingencies with differing success
40 voltage_differences = cell(size(common_contingencies, 1), 1);
41 Sbus_differences = cell(size(common_contingencies, 1), 1);
42 current_differences = cell(size(common_contingencies, 1), 1);
43 branch_current_differences = cell(size(common_contingencies,
    1), 1);
44
45 for idx = 1:size(common_contingencies, 1)
46     full_contingency_num = common_contingencies(idx, 1);
47     reduced_contingency_num = common_contingencies(idx, 2);
48
49     % Generate file paths for the MAT files
50     full_file = sprintf('N-1_simulation_full/
```

```
    power_flow_results_contingency_%d.mat',
    full_contingency_num);
51 reduced_file = sprintf('N-1_simulation_reduced/
    power_flow_results_contingency_reduced_%d.mat',
    reduced_contingency_num);
52
53 % Load the MAT files for full and reduced networks
54 full_data = load(full_file);
55 reduced_data = load(reduced_file);
56
57 % Case 1: Both power flows succeed
58 if full_data.success == 1 && reduced_data.success == 1
59     % Extract voltage, Sbus, and current data
60     full_voltage = full_data.V;
61     reduced_voltage = reduced_data.V;
62     full_Sbus = full_data.Sbus;
63     reduced_Sbus = reduced_data.Sbus;
64     full_current = full_data.I;
65     reduced_current = reduced_data.I;
66
67 % Find common buses between full and reduced networks
68 [common_bus_ids, full_idx, reduced_idx] = intersect(
    full_bus_ids, reduced_bus_ids);
69
70 % Extract voltages, Sbus, and current for common
    buses
71 full_common_voltage = full_voltage(full_idx);
72 reduced_common_voltage = reduced_voltage(reduced_idx)
    ;
73 full_common_Sbus = full_Sbus(full_idx);
74 reduced_common_Sbus = reduced_Sbus(reduced_idx);
75 full_common_current = full_current(full_idx);
76 reduced_common_current = reduced_current(reduced_idx)
    ;
77
78 % Compute the differences and store them in cell
    arrays
```

```
79     voltage_differences{idx} = full_common_voltage -
      reduced_common_voltage;
80     Sbus_differences{idx} = full_common_Sbus -
      reduced_common_Sbus;
81     current_differences{idx} = full_common_current -
      reduced_common_current;
82
83     % Compute branch current differences
84     branch_full_data = full_data.branch_current_data;
85     branch_reduced_data = reduced_data.
      branch_current_data;
86
87     % Find common branches between full and reduced
      branch data
88     [common_branches, full_branch_idx, reduced_branch_idx
      ] = intersect(branch_full_data(:, 1:2),
      branch_reduced_data(:, 1:2), 'rows');
89
90     % Compute the differences in branch current
      magnitudes (column 4 in branch_current_data)
91     branch_current_diff = branch_full_data(
      full_branch_idx, 4) - branch_reduced_data(
      reduced_branch_idx, 4);
92
93     % Ensure unique branches and aggregate current
      differences for duplicates
94     [unique_branches, ~, unique_idx] = unique(
      common_branches, 'rows');
95     branch_current_aggregated = accumarray(unique_idx,
      branch_current_diff, [], @mean); % Using mean
      aggregation
96
97     % Store the unique branches and aggregated current
      differences
98     branch_current_differences{idx} =
      branch_current_aggregated;
99
```

```

100 % Plot the absolute value of voltage differences
101 figure;
102 bar(common_bus_ids, abs(voltage_differences{idx}));
103 title(sprintf('Voltage Differences for Contingency %d',
104             ', idx));
105 xlabel('Bus Number');
106 ylabel('Absolute Voltage Difference in p.u.');
```

```

107 % Plot the absolute value of Sbus differences
108 figure;
109 bar(common_bus_ids, abs(Sbus_differences{idx}));
110 title(sprintf('Sbus Differences for Contingency %d',
111             idx));
112 xlabel('Bus Number');
113 ylabel('Absolute Sbus Difference in p.u.');
```

```

114 % Plot the absolute value of bus current differences
115 figure;
116 bar(common_bus_ids, abs(current_differences{idx}));
117 title(sprintf('Bus Current Differences for
118             Contingency %d', idx));
119 xlabel('Bus Number');
120 ylabel('Absolute Bus Current Difference in p.u.');
```

```

121 % Generate unique XData for the bar plot
122 XData = (1:length(unique_branches(:, 1)))'; % Unique
123 XData values for bar plot
```

```

124 % Format the branch pairs as strings (e.g., "1-9")
125 branch_labels = arrayfun(@(i) sprintf('%d-%d',
126             unique_branches(i, 1), unique_branches(i, 2)), ...
127             1:size(unique_branches, 1),
128             'UniformOutput', false);
```

```

129 % Plot the absolute value of branch current
130 differences (with unique branches)
131 figure;
```

```

130     bar(XData, abs(branch_current_differences{idx}));
131     title(sprintf('Branch_Current_Differences_for_
132         Contingency_%d', idx));
133     xlabel('Branches'); % Change xlabel to "Branches"
134     ylabel('Branch_Current_Difference_in_Amperes');
135     set(gca, 'XTick', XData, 'XTickLabel', branch_labels,
136         'XTickLabelRotation', 45); % Set X labels and
137         rotate them
138
139 % Case 2: Full failed, Reduced succeeded
140 elseif full_data.success == 0 && reduced_data.success ==
141     1
142     contingency_info = [common_contingencies(idx, 1),
143         common_contingencies(idx, 2), common_contingencies
144         (idx, 3:4)];
145     differing_success_full_failed_reduced_succeeded = [
146         differing_success_full_failed_reduced_succeeded;
147         contingency_info];
148     fprintf('Contingency_%d: Full_Network_PF_failed_but_
149         Reduced_Network_PF_succeeded.\n', idx);
150
151 % Case 3: Full succeeded, Reduced failed
152 elseif full_data.success == 1 && reduced_data.success ==
153     0
154     contingency_info = [common_contingencies(idx, 1),
155         common_contingencies(idx, 2), common_contingencies
156         (idx, 3:4)];
157     differing_success_full_succeeded_reduced_failed = [
158         differing_success_full_succeeded_reduced_failed;
159         contingency_info];
160     fprintf('Contingency_%d: Full_Network_PF_succeeded_
161         but_Reduced_Network_PF_failed.\n', idx);
162 end
163 end
164
165 % Save the differences and contingencies
166 save('voltage_differences.mat', 'voltage_differences');

```

```

152 save('Sbus_differences.mat', 'Sbus_differences');
153 save('current_differences.mat', 'current_differences');
154 save('branch_current_differences.mat', '
    branch_current_differences');
155 save('differing_success_full_failed_reduced_succeeded.mat', '
    differing_success_full_failed_reduced_succeeded');
156 save('differing_success_full_succeeded_reduced_failed.mat', '
    differing_success_full_succeeded_reduced_failed');
157
158 % Load the processed data
159 load('voltage_differences.mat'); % Contains the '
    voltage_differences' cell array
160 load('common_contingencies.mat'); % Contains the '
    common_contingencies' matrix
161
162 % Initialize an array to store the maximum voltage
    differences
163 max_voltage_diff = zeros(size(common_contingencies, 1), 1);
164
165 % Loop to calculate the maximum voltage differences
166 for idx = 1:size(common_contingencies, 1)
167     if ~isempty(voltage_differences{idx})
168         max_voltage_diff(idx) = max(abs(voltage_differences{
            idx}));
169     else
170         max_voltage_diff(idx) = NaN; % Handle cases where
            power flow failed
171     end
172     % Display progress for debugging
173     fprintf('Processed □ contingency □ %d □ of □ %d \n', idx, size(
        common_contingencies, 1));
174 end
175
176 % Create a graph from the branch matrix
177 if exist('h', 'var') && isvalid(h)
178     % If 'h' exists and is valid, reuse it
179     figure;

```

```

180 else
181     % Plot the full network graph if 'h' doesn't exist or is
        not valid
182     G = graph(branch_full(:, 1), branch_full(:, 2));
183     figure;
184     h = plot(G, 'Layout', 'force');
185 end
186
187 % Store handles for the different colors
188 red_branches = [];
189 yellow_branches = [];
190 green_branches = [];
191 black_branches = []; % For differing success contingencies
192
193 % Highlight branches based on max voltage difference and
        differing success
194 for idx = 1:size(common_contingencies, 1)
195     % Get the branch corresponding to the current contingency
196     branch_idx = find((branch_full(:, 1) ==
        common_contingencies(idx, 3)) & ...
197                     (branch_full(:, 2) ==
        common_contingencies(idx, 4)));
198
199     % Check if the branch exists in the full network
200     if ~isempty(branch_idx)
201         % Ensure non-empty matrices before accessing them
202         full_failed_reduced_succeeded = ~isempty(
        differing_success_full_failed_reduced_succeeded)
        && any(
        differing_success_full_failed_reduced_succeeded(:,
        1) == common_contingencies(idx, 1));
203         full_succeeded_reduced_failed = ~isempty(
        differing_success_full_succeeded_reduced_failed)
        && any(
        differing_success_full_succeeded_reduced_failed(:,
        1) == common_contingencies(idx, 1));
204

```

```

205 % Check for differing success contingencies
206 if full_failed_reduced_succeeded ||
    full_succeeded_reduced_failed
207     edge_color = [0, 0, 0]; % Black for differing
        success
208     black_branches = [black_branches; branch_full(
        branch_idx, :)];
209 else
210     % Determine the color based on the max voltage
        difference
211     if max_voltage_diff(idx) >= 0.1
212         edge_color = [1, 0, 0]; % Red
213         red_branches = [red_branches; branch_full(
        branch_idx, :)];
214     elseif max_voltage_diff(idx) >= 0.01
215         edge_color = [1, 1, 0]; % Yellow
216         yellow_branches = [yellow_branches;
        branch_full(branch_idx, :)];
217     else
218         edge_color = [0, 1, 0]; % Green
219         green_branches = [green_branches; branch_full
        (branch_idx, :)];
220     end
221 end
222
223 % Highlight the branch
224 try
225     highlight(h, branch_full(branch_idx, 1),
        branch_full(branch_idx, 2), ...
        'EdgeColor', edge_color, 'LineWidth', 2);
226 catch
227     warning('Error highlighting branch %d. Skipping
        ...', idx);
228 end
229
230 end
231 % Display progress for debugging
232 fprintf('Highlighted branch %d of %d\n', idx, size(

```

```
        common_contingencies, 1));
233 end
234
235 % Find nodes that are external to the reduced network
236 external_nodes = setdiff(full_bus_ids, reduced_bus_ids);
237
238 % Highlight the external nodes in a different color
239 highlight(h, external_nodes, 'NodeColor', 'magenta', '
    MarkerSize', 7);
240
241 % Set title and labels
242 title('Full Network Graph with Voltage Difference
    Highlighting');
243
244 % Create dummy lines for the legend
245 hold on;
246 red_line = plot(nan, nan, 'r-', 'LineWidth', 2); % Red for
    max_voltage_diff >= 0.1
247 yellow_line = plot(nan, nan, 'y-', 'LineWidth', 2); % Yellow
    for 0.01 <= max_voltage_diff < 0.1
248 green_line = plot(nan, nan, 'g-', 'LineWidth', 2); % Green
    for max_voltage_diff < 0.01
249 black_line = plot(nan, nan, 'k-', 'LineWidth', 2); % Black
    for differing success contingencies
250 magenta_node = plot(nan, nan, 'om', 'MarkerSize', 7, '
    MarkerFaceColor', 'magenta'); % Magenta for external nodes
251
252 % Add the legend using the line handles (including external
    nodes)
253 legend([red_line, yellow_line, green_line, black_line,
    magenta_node], ...
254         'Voltage Diff >= 0.1 (Red)', ...
255         '0.01 <= Voltage Diff < 0.1 (Yellow)', ...
256         'Voltage Diff < 0.01 (Green)', ...
257         'Differing Success (Black)', ...
258         'External Nodes (Magenta)', 'Location', 'Best');
```

**Listing A.4:** MATLAB code for Comparing Power Flow Results for each contingency

- MATLAB Code for Relative Error(V) Computation:

```

1 % Load voltage differences and full network voltage data
2 load('voltage_differences.mat'); % Contains
   voltage_differences cell array
3 load('common_contingencies.mat'); % Contains
   common_contingencies matrix
4
5 % Initialize arrays to store max, avg, and min values for
   voltage differences
6 voltage_differences_max = zeros(size(common_contingencies, 1)
   , 1);
7 voltage_differences_avg = zeros(size(common_contingencies, 1)
   , 1);
8 voltage_differences_min = zeros(size(common_contingencies, 1)
   , 1);
9
10 % Initialize arrays for relative errors
11 relative_error_max = zeros(size(common_contingencies, 1), 1);
12 relative_error_avg = zeros(size(common_contingencies, 1), 1);
13 relative_error_min = zeros(size(common_contingencies, 1), 1);
14
15 % Loop over each contingency to compute the required
   statistics
16 for idx = 1:size(common_contingencies, 1)
17     % Skip if voltage differences are empty
18     if isempty(voltage_differences{idx})
19         continue;
20     end
21
22     % Extract voltage differences for the current contingency
23     v_diff = voltage_differences{idx};
24
25     % Compute max, avg, and min values of the voltage
       differences

```

```
26     voltage_differences_max(idx) = max(abs(v_diff));
27     voltage_differences_avg(idx) = mean(abs(v_diff));
28     voltage_differences_min(idx) = min(abs(v_diff));
29
30     % Load the full network voltage data for the current
31     contingency
32     full_contingency_num = common_contingencies(idx, 1);
33     full_file = sprintf('N-1_simulation_full/
34     power_flow_results_contingency_%d.mat',
35     full_contingency_num);
36     full_data = load(full_file);
37     full_voltage = full_data.V;
38
39     % Compute max, avg, and min voltages for the full network
40     V_max = max(abs(full_voltage));
41     V_avg = mean(abs(full_voltage));
42     V_min = min(abs(full_voltage));
43
44     % Compute relative errors
45     relative_error_max(idx) = voltage_differences_max(idx) /
46     V_max;
47     relative_error_avg(idx) = voltage_differences_avg(idx) /
48     V_avg;
49     relative_error_min(idx) = voltage_differences_min(idx) /
50     V_min;
51
52     % Display progress for debugging
53     fprintf('Processed □contingency □%d □of □%d\n', idx, size(
54     common_contingencies, 1));
55
56 end
57
58 % Save the results to MAT files
59 save('voltage_differences_max.mat', 'voltage_differences_max'
60     );
61 save('voltage_differences_avg.mat', 'voltage_differences_avg'
62     );
63 save('voltage_differences_min.mat', 'voltage_differences_min'
```

```

);
54 save('relative_error_max.mat', 'relative_error_max');
55 save('relative_error_avg.mat', 'relative_error_avg');
56 save('relative_error_min.mat', 'relative_error_min');

```

Listing A.5: MATLAB Code for Relative Error(V) Computation

• MATLAB Code for Relative Error(Q) Computation:

```

1 % Load the required data
2 load('Sbus_differences.mat'); % Contains
   Sbus_differences cell array (reactive power differences)
3 load('common_contingencies.mat'); % Contains
   common_contingencies matrix
4
5 % Initialize arrays to store max, avg, and min values of
   reactive power (Q) differences
6 Q_differences_max = zeros(size(common_contingencies, 1), 1);
7 Q_differences_avg = zeros(size(common_contingencies, 1), 1);
8 Q_differences_min = zeros(size(common_contingencies, 1), 1);
9
10 % Initialize arrays to store relative errors for reactive
   power
11 relative_error_Q_max = zeros(size(common_contingencies, 1),
   1);
12 relative_error_Q_avg = zeros(size(common_contingencies, 1),
   1);
13 relative_error_Q_min = zeros(size(common_contingencies, 1),
   1);
14
15 % Load bus data from the reduced network to identify
   generator buses
16 reduced_bus_data = load('N-1_simulation_reduced/
   Reduced_Network.mat');
17 reduced_gen_buses = reduced_bus_data.Gen_new(:, 1); %
   Assuming Gen_new has generator buses
18
19 % Loop over each contingency to compute the required

```

```

    statistics for generator buses only
20 for idx = 1:size(common_contingencies, 1)
21     % Skip if Sbus differences are empty
22     if isempty(Sbus_differences{idx})
23         continue;
24     end
25
26     % Extract Sbus differences (reactive power differences)
    for the current contingency
27     Q_diff = imag(Sbus_differences{idx}); % Imaginary part
    represents reactive power differences (Q)
28
29     % Extract the Q differences for generator buses only
30     [~, reduced_gen_idx] = intersect(reduced_gen_buses,
    reduced_gen_buses); % Simply use reduced generator
    buses
31     Q_diff_gen = Q_diff(reduced_gen_idx);
32
33     % Compute max, avg, and min values of reactive power
    differences for generator buses
34     if ~isempty(Q_diff_gen)
35         Q_differences_max(idx) = max(abs(Q_diff_gen));
36         Q_differences_avg(idx) = mean(abs(Q_diff_gen));
37         Q_differences_min(idx) = min(abs(Q_diff_gen));
38     end
39
40     % Load the full network power flow data for the current
    contingency
41     full_contingency_num = common_contingencies(idx, 1); %
    Full network contingency number
42     full_file = sprintf('N-1_simulation_full/
    power_flow_results_contingency_%d.mat',
    full_contingency_num);
43     full_data = load(full_file);
44     full_gen_data = full_data.gen(:, 3); % Extract the Q (
    Reactive Power) from the generator data (column 3)
45
```

```

46 % Compute Q_max for the full network (generator buses
    only)
47 [~, full_gen_idx] = intersect(full_data.gen(:, 1),
    reduced_gen_buses); % Find generator buses in full
    network
48 Q_max_full = max(abs(full_gen_data(full_gen_idx)));
49
50 % Calculate the Reactive Power (Q) as 50% of Q_max
51 Q = 0.5 * Q_max_full;
52
53 % Compute relative errors for Q differences
54 relative_error_Q_max(idx) = Q_differences_max(idx) / Q;
55 relative_error_Q_avg(idx) = Q_differences_avg(idx) / Q;
56 relative_error_Q_min(idx) = Q_differences_min(idx) / Q;
57
58 % Display progress for debugging
59 fprintf('Processed contingency %d of %d\n', idx, size(
    common_contingencies, 1));
60 end
61
62 % Save the results to MAT files
63 save('Q_differences_max.mat', 'Q_differences_max');
64 save('Q_differences_avg.mat', 'Q_differences_avg');
65 save('Q_differences_min.mat', 'Q_differences_min');
66 save('relative_error_Q_max.mat', 'relative_error_Q_max');
67 save('relative_error_Q_avg.mat', 'relative_error_Q_avg');
68 save('relative_error_Q_min.mat', 'relative_error_Q_min');

```

Listing A.6: MATLAB Code for Relative Error(Q) Computation

• MATLAB Code for Relative Error(I<sub>branch</sub>) Computation:

```

1 % Load the branch current differences and common
    contingencies
2 load('branch_current_differences.mat'); % Contains '
    branch_current_differences' cell array
3 load('common_contingencies.mat'); % Contains '
    common_contingencies' matrix

```

```
4
5 % Initialize arrays to store max, avg, and min values of
  branch current differences
6 branch_current_differences_max = zeros(size(
  common_contingencies, 1), 1);
7 branch_current_differences_avg = zeros(size(
  common_contingencies, 1), 1);
8 branch_current_differences_min = zeros(size(
  common_contingencies, 1), 1);
9
10 % Initialize arrays to store relative errors
11 relative_error_br_current_max = zeros(size(
  common_contingencies, 1), 1);
12 relative_error_br_current_avg = zeros(size(
  common_contingencies, 1), 1);
13 relative_error_br_current_min = zeros(size(
  common_contingencies, 1), 1);
14
15 % Loop over each contingency to compute the required
  statistics
16 for idx = 1:size(common_contingencies, 1)
17     % Skip if branch current differences are empty
18     if isempty(branch_current_differences{idx})
19         continue;
20     end
21
22     % Extract branch current differences for the current
  contingency
23     branch_curr_diff = branch_current_differences{idx};
24
25     % Compute max, avg, and min values of branch current
  differences
26     branch_current_differences_max(idx) = max(abs(
  branch_curr_diff));
27     branch_current_differences_avg(idx) = mean(abs(
  branch_curr_diff));
28     branch_current_differences_min(idx) = min(abs(
```

```

    branch_curr_diff));
29
30 % Load the reduced network branch current data for the
    current contingency
31 reduced_contingency_num = common_contingencies(idx, 2); %
    Reduced network contingency number
32 reduced_file = sprintf('N-1_simulation_reduced/
    power_flow_results_contingency_reduced_%d.mat',
    reduced_contingency_num);
33 reduced_data = load(reduced_file);
34 reduced_branch_current = reduced_data.branch_current_data
    (:, 3); % Branch current magnitudes are in column 3
35
36 % Compute I_max (maximum branch current)
37 I_max = max(abs(reduced_branch_current));
38 I = I_max; % Use I_max as per instruction
39
40 % Compute relative errors for branch current differences
41 relative_error_br_current_max(idx) =
    branch_current_differences_max(idx) / I;
42 relative_error_br_current_avg(idx) =
    branch_current_differences_avg(idx) / I;
43 relative_error_br_current_min(idx) =
    branch_current_differences_min(idx) / I;
44
45 % Display progress for debugging
46 fprintf('Processed %d of %d\n', idx, size(
    common_contingencies, 1));
47 end
48
49 % Save the results to MAT files
50 save('branch_current_differences_max.mat', '
    branch_current_differences_max');
51 save('branch_current_differences_avg.mat', '
    branch_current_differences_avg');
52 save('branch_current_differences_min.mat', '
    branch_current_differences_min');

```

```

53 save('relative_error_br_current_max.mat', '
    relative_error_br_current_max');
54 save('relative_error_br_current_avg.mat', '
    relative_error_br_current_avg');
55 save('relative_error_br_current_min.mat', '
    relative_error_br_current_min');

```

**Listing A.7:** MATLAB Code for Relative Error ( $I_{\text{branch}}$ ) Computation

- **MATLAB Code for PQ and PV Bus Classification in Full and Reduced Networks:**

```

1 % Define tolerance levels
2 deltaV_tolerance = 10^(-5);
3 deltaQ_tolerance = 10^(-3);
4
5 % Initialize matrices to store classification results for
6   both networks
7 classificationResultsFull = [];
8 classificationResultsReduced = [];
9
10 % ===== Process Full Network
11   =====
12 full_network_path = 'N-1_simulation_full/
13   contingency_mapping_full.mat';
14 load(full_network_path, 'contingencyMapping'); % Load
15   contingency mapping for the full network
16 num_contingencies_full = length(unique(contingencyMapping(:,
17   1)));
18
19 for contingencyIndex = 1:num_contingencies_full
20   contingency_file_full = sprintf('N-1_simulation_full/
21   power_flow_results_contingency_%d.mat',
22   contingencyIndex);
23   load(contingency_file_full); % Load 'bus', 'gen', 'V',
24   etc.
25
26   Q = gen(:, 3);

```

```

19   Q_max = gen(:, 4);
20   Q_min = gen(:, 5);
21   V_setpoint = gen(:, 6);
22
23   contingencyClassificationFull = [];
24
25   for i = 1:size(gen, 1)
26       gen_bus = gen(i, 1);
27       V_gen_bus = abs(V(gen_bus));
28
29       deltaQ_min = abs(Q(i) - Q_min(i));
30       deltaQ_max = abs(Q_max(i) - Q(i));
31       deltaV = abs(V_gen_bus - V_setpoint(i));
32
33       if (deltaQ_min <= deltaQ_tolerance || deltaQ_max <=
34           deltaQ_tolerance) && deltaV > deltaV_tolerance
35           contingencyClassificationFull = [
36               contingencyClassificationFull;
37               contingencyIndex, gen_bus, 1];
38       elseif (deltaQ_min > deltaQ_tolerance || deltaQ_max >
39           deltaQ_tolerance) && deltaV <= deltaV_tolerance
40           contingencyClassificationFull = [
41               contingencyClassificationFull;
42               contingencyIndex, gen_bus, 2];
43       end
44   end
45
46   classificationResultsFull = [classificationResultsFull;
47       contingencyClassificationFull];
48
49   end
50
51   save('classification_results_full.mat', '
52       classificationResultsFull');
53
54   % ===== Process Reduced Network
55   % =====
56
57   reduced_network_path = 'N-1_simulation_reduced/
58       Combine_contingency_branch_reduced.mat';

```

```
46 load(reduced_network_path, '  
    Combine_contingency_branch_reduced');  
47 num_contingencies_reduced = length(unique(  
    Combine_contingency_branch_reduced(:, 1)));  
48  
49 for contingencyIndex = 1:num_contingencies_reduced  
50     contingency_file_reduced = sprintf('N-1  
        _simulation_reduced/  
        power_flow_results_contingency_reduced_%d.mat',  
        contingencyIndex);  
51     load(contingency_file_reduced);  
52  
53     Q = gen(:, 3);  
54     Q_max = gen(:, 4);  
55     Q_min = gen(:, 5);  
56     V_setpoint = gen(:, 6);  
57  
58     contingencyClassificationReduced = [];  
59  
60     for i = 1:size(gen, 1)  
61         gen_bus = gen(i, 1);  
62         V_gen_bus = abs(V(gen_bus));  
63  
64         deltaQ_min = abs(Q(i) - Q_min(i));  
65         deltaQ_max = abs(Q_max(i) - Q(i));  
66         deltaV = abs(V_gen_bus - V_setpoint(i));  
67  
68         if (deltaQ_min <= deltaQ_tolerance || deltaQ_max <=  
            deltaQ_tolerance) && deltaV > deltaV_tolerance  
69             contingencyClassificationReduced = [  
                contingencyClassificationReduced;  
                contingencyIndex, gen_bus, 1];  
70         elseif (deltaQ_min > deltaQ_tolerance || deltaQ_max >  
            deltaQ_tolerance) && deltaV <= deltaV_tolerance  
71             contingencyClassificationReduced = [  
                contingencyClassificationReduced;  
                contingencyIndex, gen_bus, 2];
```

```
72         end
73     end
74
75     classificationResultsReduced = [
76         classificationResultsReduced;
77         contingencyClassificationReduced];
78 end
79
80 save('classification_results_reduced.mat', '
81     classificationResultsReduced');
```

**Listing A.8:** MATLAB Code for PQ and PV Bus Classification in Full and Reduced Networks

## List of Figures

1.1	Single-line diagram of the 66-bus <i>Rueda_network_3_1</i> used in this thesis for applying Ward's network reduction technique. . . . .	10
3.1	Flowchart for Power Flow Analysis Algorithm . . . . .	24
3.2	Flowchart for the Breadth-First Search-like Approach to Node Identification	27
3.3	Full Network Graph with Internal, Boundary, and External Nodes . . . . .	29
3.4	Flowchart for Ward's Reduction . . . . .	32
3.5	Flowchart for Updating the Branch Matrix after Reduction . . . . .	35
3.6	Flowchart for Updating R and X values of Boundary Branches after Reduction	37
3.7	Flowchart for Updating Power Demands after Reduction . . . . .	39
3.8	Flowchart for Updating Generator Matrix after Reduction . . . . .	42
3.9	Flowchart for Checking the Slack Bus after Reduction . . . . .	44
3.10	Flowchart for Remaining Modifications of the Reduced Network . . . . .	48
3.11	Flowchart for Renumbering and Mapping the Reduced Network . . . . .	50
4.1	Comparison of Power Flow Results between Full and Reduced Networks under Normal Conditions. . . . .	57
5.1	Full Network with Radial Branches Highlighted. Radial branches are shown in red, while non-radial branches are shown in black. . . . .	63
5.2	Reduced Network with Radial Branches Highlighted. Radial branches are shown in red, while non-radial branches are shown in black. The bus numbers in the reduced network have been renumbered during the network reduction process, but a mapping exists to recover the original bus numbers for comparison with the full network. . . . .	64
5.3	Flowchart for Non-Radial Branch Detection . . . . .	65
5.4	Flowchart for Contingency Mapping Process . . . . .	68
5.5	Flowchart for Power Flow Analysis during Contingency Simulation . . . . .	68
5.6	Flowchart for Recovering Original Indices for the Reduced Network . . . . .	71
5.7	Flowchart for Updating Contingency Results with Original Bus Numbers . . . . .	72

5.8	Flowchart for Finding Common Contingencies between Full and Reduced Networks . . . . .	74
5.9	Flowchart for Power Flow Comparison between Full and Reduced Networks	76
5.10	Relative Errors ( $\Delta V/V$ ) for Each Common Contingency (Branch). . . . .	78
5.11	Full Network Graph with Voltage Difference Highlighting. The branches are color-coded to reflect voltage discrepancies between the full and reduced networks for each common contingency. . . . .	80
5.12	PV to PQ Status Change of Generator Bus in the Reduced Network against Full Network Status. The graph tracks generator buses that change classification from PV (voltage control) to PQ (no voltage control) during contingencies. . . . .	82
5.13	Single-line diagram of the 66-bus <i>Rueda_network_3_1</i> with impacted area highlighted for contingencies 5, 4, and 3. . . . .	84
5.14	Voltage differences for Contingency 5. . . . .	84
5.15	Relative Errors ( $\Delta Q/Q$ ) for Each Common Contingency (Generator Buses). The graph compares the maximum, average, and minimum relative reactive power errors between the full and reduced networks for each common contingency. . . . .	86
5.16	Relative Errors ( $\Delta I/I$ ) for Each Common Contingency (Branch). The graph compares the maximum, average, and minimum relative current errors between the full and reduced networks for each common contingency. . . . .	87
5.17	Network with Boundary nodes 18, 39 and 62 . . . . .	89
5.18	Graph with Internal(green), Boundary(blue), and External(red) Nodes After Boundary Shift. . . . .	90
5.19	Relative Voltage Errors ( $\Delta V/V$ ) for Each Common Contingency After Boundary Shift. . . . .	91
5.20	Full Network Graph with Voltage Difference Highlighting After Boundary Shift. . . . .	92
5.21	PV to PQ Status Change for Generator Buses After Boundary Shift. . . . .	93
5.22	Relative Branch Current Errors ( $\Delta I/I$ ) for Each Common Contingency After Boundary Shift. . . . .	94
5.23	Relative Reactive Power Errors ( $\Delta Q/Q$ ) for Each Common Contingency After Boundary Shift. . . . .	95

# List of Tables

2.1 Comparison of Network Reduction Techniques . . . . . 19



## List of Symbols

Variable	Description	SI unit
$V$	Voltage	Volt
$I$	Current	Ampere
$R_{ij}$	Resistance between i and j	ohm
$X_{ij}$	Inductance between i and j	ohm
$Y$	Admittance Matrix	mho or Siemens
$S$	Apparent Power	VA
$P$	Real Power	Watt
$Q$	Reactive Power	VAR
$y_{ik}$	Line Admittance between i and k	mho or Siemens
$B_{ik}$	Line Charging Susceptance between i and k	mho or Siemens
$G_{ik}$	Line Charging Conductance between i and k	mho or Siemens
$B_s$	Bus Shunt Susceptance	mho or Siemens
$baseMVA$	base Power	VA



## Acknowledgements

I would like to thank my advisor Prof. Alberto Berizzi and co-advisor Prof. Valentin Ilea for this challenge, that has given me the opportunity to explore the world of Power System Research.

Most importantly, I would like to thank my parents Akhil ch. Konwar and Dulumoni Konwar, my wife Indrani Chetia Konwar for the love and support through out the period. Finally, I would like to thank Assam Power Distribution Company Limited for the support through out my study leave period. Thank you all.

