



AFRL-RI-RS-TR-2020-033

LARGE-SCALE POWER GRID SIMULATION VIA EQUIVALENT CIRCUIT FORMULATION

CARNEGIE MELLON UNIVERSITY

FEBRUARY 2020

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2020-033 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

JENNIFER A. CASSETTI
Work Unit Manager

/ S /

JAMES S. PERRETTA
Deputy Chief, Information
Exploitation & Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

TABLE OF CONTENTS

Section	Page
List of Figures	iii
List of Tables	vii
1.0 SUMMARY	1
2.0 INTRODUCTION	2
3.0 BACKGROUND AND PRIOR STATE OF THE ART.....	5
3.1 Positive Sequence and Three-Phase Power Flow Formulations	5
3.2 'PQV' based Formulation for Positive Sequence Power Flow Problem	5
3.2.1 Current Injection Method for Three-Phase Power Flow Problem	6
3.2.2 Backward-Forward Sweep Method	6
3.2.3 Holomorphic Embedding Load Flow Method	7
3.2.4 Continuation Power Flow Method.....	7
3.3 Circuit Simulation Methods	7
3.3.1 Limiting methods.....	8
3.3.2 Homotopy Methods.....	9
4.0 METHODS, ASSUMPTIONS, AND PROCEDURES.....	11
4.1 Equivalent Circuit Approach	11
4.1.1 Split-Circuit Formulation.....	11
4.1.2 Equivalent Circuit Models for the Positive Sequence Power Flow Problem	12
4.1.3 Equivalent Circuit Models for Three-Phase Power Flow Problem.....	34
4.1.3.1 Slack Bus.....	34
4.1.4 Preliminary Results for Three-phase power flow	39
4.1.5 Physics Based Models.....	40
4.2 Circuit Simulation Methods for Power System Analyses.....	45
4.2.1 Limiting Methods.....	45
4.2.2 Homotopy Methods.....	48
4.2.3 Algorithm.....	53
4.2.4 Results.....	54
4.3 Combined Transmission and Distribution System Analyses.....	67
4.3.1 Background.....	68
4.3.2 General Methodology.....	69
4.3.3 Coupling port for Combined Transmission and Distribution Circuit	69
4.3.4 Combined T&D Simulation on a Single Core	71
4.3.5 Combined T&D Simulation in Parallel with Distributed Cores	76
4.3.6 Bordered Block Diagonal (BBD) Matrix Structure	78
4.3.7 Notes on Convergence	83
5.0 RESULTS AND DISCUSSIONS	85

5.1	SUGAR C++ (an ECF based solver)	85
5.1.1	Concepts and Abstractions	85
5.1.2	Module and Interface Design	87
5.1.3	ECF solver	88
5.1.4	Data pipeline.....	100
5.1.5	SugarAlgorithms.....	108
5.1.6	Network Graph Library	109
5.1.7	Settings and Input File Format.....	111
5.1.8	Used Third-Party libraries	112
5.1.9	Performance Evaluation of SUGAR C++	113
5.2	Monte Carlo Framework.....	119
5.2.1	Properties of Monte Carlo Algorithms	119
5.2.2	Design Considerations	119
5.2.3	Implementation	120
5.2.4	Probabilistic Input Data	122
5.2.5	Used Libraries.....	123
5.2.6	Correlations within Monte Carlo Simulations.....	123
5.2.7	Memory Profile of the Monte Carlo Framework.....	125
6.0	CONCLUSIONS.....	127
	Appendix A. BIG Model: Linear Model for Aggregated Load in the Power Grid	128
A.1	Background.....	128
A.2	Circuit Theoretic <i>BIG</i> Load Model	129
A.3	Contributions of <i>BIG</i> Load Model	132
A.3.1	Linearity	132
A.3.2	Captures Voltage Sensitivities.....	132
A.3.3	Captures Voltage Angle Information.....	134
A.3.4	Generic Model for both Power-flow and Transient Analyses.....	135
	Appendix B. Unified Power System Analyses and Models.....	136
B.1	Introduction	136
B.2	Validation of IM model	137
B.3	Solution consistency between the steady-state and transient analysis	138
7.0	REFERENCES.....	140
8.0	LIST OF ACRONYMS	147

List of Figures

Figure 1: Simple three-bus power flow network and its corresponding equivalent circuit.	12
Figure 2: Equivalent circuit model for PV generator model.	14
Figure 3: Voltage magnitude constraint control equivalent circuit.	14
Figure 4: Oscillations observed during PV-PQ switching in the outer loop of NR.....	16
Figure 5: Voltage constraint behavior for continuous generator model.	18
Figure 6: Generator characteristics as a function of generator convergence parameter.	21
Figure 7: Generator characteristics as a function of generator smoothness parameter.	21
Figure 8: Continuous analytical model for modeling the AGC and droop control of the generator based on participation factor.	23
Figure 9: Equivalent split-circuit PQ load model.	25
Figure 10: Real and imaginary equivalent circuit for the ZIP load model.	27
Figure 11: Equivalent circuit of a BIG load model.	27
Figure 12: Equivalent circuit for a transformer.	28
Figure 13: Real and imaginary circuit for a transformer model.	29
Figure 14: Continuous transformer tap control schematic.	30
Figure 15: Equivalent circuit of a pi-model of the transmission line.	31
Figure 16: Real and imaginary circuit for the pi-model of transmission line.	32
Figure 17: Real and imaginary circuits for slack bus in three-phase power flow problem.	34
Figure 18: Real circuit for a) wye connected ZIP load model (on left) b) delta (D) connected ZIP load model (on right).	36
Figure 19: Real circuit of a transmission line (Phase A).	37
Figure 20: Real circuit for the grounded wye – grounded wye transformer with no phase shift.	39
Figure 21: Standard 4-bus test case system.	39
Figure 22: Superimposition of DQ-axis on 3-phase induction motor.	42
Figure 23: Equivalent circuit for 3-phase induction motor: (i) Electrical circuit; and (ii) Mechanical Circuit.	44
Figure 24: Voltage profile for maximum bus voltage in 2869 bus system: a) w/o Variable limiting b) with Variable limiting.	47
Figure 25: Homotopy factor embedded in transmission line equivalent circuit.	51
Figure 26: Homotopy factor embedded in transformer equivalent circuit.	51
Figure 27: Solution of bus 3 voltage for IEEE 14 bus test system with increasing loading factors with and without circuit simulation methods.	55

Figure 28: Power flow results for 2869 bus and 9241 bus test systems with and without circuit simulation techniques.	56
Figure 29: Results for 13659 buses PEGASE system.	58
Figure 30: Convergence sweep of large cases that represent Eastern Interconnection from range of initial conditions (number of nodes for each test system given in the legend box).....	59
Figure 31: Convergence plot for ACTIVgs70k (left) and SyntheticUSA (right) testcases.	60
Figure 32: Maximum bus voltage range for contingency analysis.	63
Figure 33: System bus voltage pre and post corrective action.....	64
Figure 34: Convergence of 145 bus test case for three-phase power flow with (middle) and without (top) power stepping. For the power stepping case, the green dotted line represents the change in continuation factor λ whose evolution is shown in the bottom plot.	65
Figure 35: Coupling port for combined transmission and distribution analysis.....	70
Figure 36: General framework for performing combined transmission and distribution simulation using equivalent circuit approach.	72
Figure 37: POI voltages under normal and contingency operation with changing distribution load.....	73
Figure 38: POI voltages under normal and contingency operation with changing distribution load and with DERs in the system.....	73
Figure 39: Voltage in pu at the point of interconnection with increasing loading factor of the distribution feeder.	74
Figure 40: Voltage in pu at the point of interconnection with increasing loading factor of the distribution feeder i) with DERs and ii) without DERs.	75
Figure 41: Reverse power flow observed during increasing DERs in the distribution feeder.	76
Figure 42: Weakly coupled transmission and distribution network.	78
Figure 43: “Torn” transmission and distribution sub-systems.....	79
Figure 44: Bordered Block Diagonal structure for combined transmission and distribution system.	79
Figure 45: Comparison of combined T&D simulation algorithms: i) Single machine setup using NR (in blue), ii) Parallel simulation on distributed cores using GSN (in red).	82
Figure 46: Voltages at the POI in the outer loop of GSN.....	83
Figure 47: Concepts hierarchy of an ECF based solver.	85
Figure 48: Global design overview of SUGAR C++ solver.....	87
Figure 49: The two <i>SparseMatrix</i> classes share a common interface within SUGAR C++.....	88
Figure 50: Sparse matrix conversion during Newton-Raphson loop.....	89

Figure 51: Activity diagram of the <i>NRSolver</i> 's inner loop implementation.....	90
Figure 52: UML class diagram of the <i>SugarSolver</i>	92
Figure 53: Activity diagram of the outer control loop implemented by the <i>SugarSolver</i>	93
Figure 54: Class diagram of the <i>ContinuationMethod</i> control interface.	94
Figure 55: Core algorithm of the adaptive Tx-stepping implementation.	95
Figure 56: UML class diagram of the <i>QLimiting</i> class that implements the <i>ContinuationMethod</i> interface.....	97
Figure 57: Control algorithms implemented by <i>QLimiting</i>	98
Figure 58: UML class diagram of the <i>DeviceControl</i> class.....	99
Figure 59: Relationships between model data structures of the SUGAR C++ implementation.	100
Figure 60: UML diagram of the <i>Parser</i>	100
Figure 61: Algorithm of the SUGAR C++ parser that applies a factory pattern to instantiate the correct algorithm.....	101
Figure 62: Class diagram of the .raw file parser.....	102
Figure 63: UML class diagram of the <i>PSModels</i> class.....	103
Figure 64: UML diagram of PF, feasibility PF, and SE models with their common interface <i>Model::Base</i>	104
Figure 65: UML class diagram of a Plant model.....	106
Figure 66: UML class diagram of the <i>SugarAlgorithms</i> class.....	108
Figure 67: Flow diagram of the <i>runPowerFlow()</i> function of the <i>SugarAlgorithms</i> class.	109
Figure 68: UML diagram of structures and classes that represent the <i>SystemGraph</i> library implementation.	110
Figure 69: UML class diagram of input parameter data structures.....	111
Figure 70: Per iteration solution times of the SUGAR C++ implementation for openly available Power Flow cases using KLU.....	113
Figure 71: Initial Power Flow matrix creation time of selected power system cases. This is a one-time cost.	114
Figure 72: Performance of the SUGAR C++ solver for a real Eastern Interconnection case and good initial conditions	115
Figure 73: SUGAR C++ profile of a real eastern interconnect PF simulation using Tx- stepping.	115
Figure 74: Solution times of synthetic State Estimation samples that are crated from openly available power flow data.	116
Figure 75: Valgrind massif memory profiler output for the 9241-bus PEGASE case.	117

Figure 76: A double-logarithmic plot of requirement for different sized systems for total required memory, KLU memory (symbolic and numeric factorizations), and system matrix size.	118
Figure 77: The Monte Carlo algorithm.....	120
Figure 78: UML class diagram of the parallel <i>MonteCarlo</i> solver and related data structures.....	121
Figure 79: UML class diagrams of the Probabilistic model interface and the class interface for defining and sampling a variable's distribution.....	122
Figure 80: Example of json-based probabilistic input.....	122
Figure 81: Branch model that defines a localized correlation between its parameters "Gser" and "Bser".....	124
Figure 82: Defines wind power generation correlations between generators in a system.	124
Figure 83: Class diagram of the correlated random number generator.....	125
Figure 84: Memory scaling properties of the MC framework for three different size system with additional parallelism.	126

TABLE OF TABLES

Table 1: Results to demonstrate AGC functionality using continuous analytical model.	24
Table 2: Preliminary results for positive sequence power flow with equivalent circuit approach.....	33
Table 3: SUGAR three-phase results for 4-Bus test case.....	40
Table 4: Three-phase squirrel cage induction motor parameters.....	44
Table 5: IM results in equivalent circuit framework for steady-state (power flow) and time-domain transient analysis.....	45
Table 6: Comparison of SUGAR with and without circuit simulation techniques	57
Table 7: Comparison of results for modified 11 bus test case.....	58
Table 8: Convergence performance for large eastern interconnection test cases.....	60
Table 9: Contingency analysis for large test cases.....	61
Table 10: N-1 contingency analysis on set of critical equipment.	62
Table 11: SUGAR three-phase results for taxonomy feeders and large cases	66

1.0 SUMMARY

A robust framework for steady-state analysis (power flow and three-phase power flow problem) of the transmission as well as distribution networks is essential for operation and planning of the electric power grid. The critical nature of this analysis has led to this problem being one of the most actively researched topics in the field of energy in the last few decades. This has produced significant advances in the related technologies; however, the present state-of-the-art methods still lack the general robustness needed to securely and reliably operate as well as plan for the ever-changing power grid. The reasons for this are many, but the most important ones are: lack of general assurance toward convergence of power flow and three-phase power flow problems to the correct physical solution when a good initial state is not available; the use of disparate formulation and modeling frameworks for transmission and distribution steady-state analyses that has led to the two analyses being modeled and simulated separately.

This project has addressed the existing limitations in steady-state analysis of power grids to enable a more secure and reliable environment for power grid operation and planning. To that effect, we developed a generic framework based on equivalent circuit formulation that can model both the positive sequence network of the transmission grid and the three-phase network of the distribution grid without loss of generality. Furthermore, we demonstrated that when combined with novel as well as adapted circuit simulation techniques, the framework can robustly solve for the steady-state solution for both these network models (positive sequence and three-phase) by constraining the developed models in their physical space, independent of the choice of initial conditions. Importantly, the developed framework treats the transmission grid no differently than the distribution grid and, therefore, allows for any further advances in the field to be directly applicable to the analysis of both. One of which is the ability to robustly simulate the “combined” positive sequence network of the transmission grid and three-phase network of the distribution grid.

To validate the applicability of the proposed equivalent circuit formulation to realistic industry sized systems as well to demonstrate the robustness of the developed methods, we developed both Python and C++ versions of a prototype tool, SUGAR, and applied it for simulation of large positive-sequence and three-phase networks individually and jointly from arbitrary initial conditions to show convergence to correct physical solution. Examples for positive sequence transmission networks include 75k+ nodes test cases representing the U.S. Eastern Interconnection high-voltage grid and three-phase networks that include 8k+ nodes taxonomy feeders. To further extend this work to demonstrate its efficacy for anomaly detection, we developed models for measurement data from the grid that when incorporated into the SUGAR framework, provide for improved accuracy in State Estimation as well.

2.0 INTRODUCTION

An interconnected electric grid is a network of synchronized power providers and consumers that are connected via transmission and distribution lines and operated by one of multiple entities. Secure and reliable operation of this electric grid is of utmost importance to a country's economy and the well-being of its citizens. In the U.S., the electricity-based services are considered an essential service [1], a lack of which can result in significant societal chaos [2]. Since its early inception, the grid remained mostly unaltered and only began to experience changes in the last decade or so, some of which include [1]: (i) changing generation mix due to an electricity generation shift from a few large central plants, mostly fossil-fueled, to smaller and often variable renewable generators, (ii) changing demand loads in the retail electricity markets due to demographic and economic changes, (iii) integration of smart grid technologies, and (iv) increasing threats due to adversarial attacks or aging infrastructure. To navigate the grid through these changes while maintaining its reliability and security requires investment toward modern infrastructure, adequate policy and state-of-the-art simulation tools. The underlying work in this report is geared towards developing better power grid simulation tools and analysis methods.

At present, numerous methods exist for simulating and analyzing the electric grid. These can be broadly categorized into one of the following categories: (i) steady-state analysis in frequency domain (power flow, three phase power flow, and harmonic analyses), (ii) transient and steady-state analysis in time domain, (iii) static state-estimation, (iv) analysis for optimal dispatch of resources, and v) other market dispatch-based analyses. Among these analyses, two that are primarily used for the day-to-day operation and planning of the grid are: the steady-state analysis in the frequency domain (power flow and three-phase power flow) to obtain the system frequency voltage and current phasors and the transient analysis in time domain to obtain time domain voltages and currents following a given event. Generally, for system operation and planning, power flow and three-phase power flow runs are first performed on all relevant cases followed by more computational heavy transient analysis for specific cases to gain more insight into the steady-state results. This pattern of analysis is generally driven by inconsistencies in results between power flow and transient analysis, wherein steady-state solution obtained from transient analysis (as time tends to infinity) is considered more accurate than the solution obtained from the power flow. This inconsistency between the steady state solution obtained from the transient analysis and the steady-state solution obtained from the power flow analysis is a cause of concern and is conceptually addressed in Appendix B of this report. The focus of this project, however, is toward developing a steady-state solver for robust convergence of positive sequence and three-phase power flow problems independent of scale or complexity.

The classical power grid has evolved over time to result in an invisible divide between the transmission network and distribution network. This in the past has allowed for the two networks to be studied separately resulting in disparate solution methods and modeling frameworks for analyzing the two. For instance, the steady-state solution of the transmission system is obtained via positive sequence or balanced alternating current (AC) power flow analysis, whereas the steady-state operating point for the distribution system is obtained via three-phase AC power flow analysis. The industry standard for solving the positive sequence power flow problem is the 'PQV' formulation [3], wherein nonlinear power mismatch equations are solved for bus voltage magnitude and angle state variables that further define the steady-state operating point of the system. On the other hand, two distinct analysis methods: (i) backward-forward sweep method [4] and (ii) current injection method (CIM) [5] are primarily used interchangeably for obtaining the steady-state solution of the three-phase power flow problem. In the grid of tomorrow, however, this invisible divide between the transmission and distribution system is bound to disappear, thereby requiring solution methods and a modeling framework that can model and simulate both the transmission and distribution networks whether independently or jointly. The need for which was unequivocally highlighted by one of the speakers in an ARPA-E workshop to identify paths to large-scale deployment of renewable energy resources: "tools are not graceful in considering penetration levels at which much of the thermal fleet could get de-committed," and that "studies do not co-simulate impact of renewable injection into receiving AC systems" [6]. This project addresses these concerns by developing a generic circuit-theoretic framework for modeling both the transmission network and distribution network that can further simulate them either separately (through power flow and three-phase power flow) or jointly (through combined transmission and distribution power flow).

In general, all of the existing methods for solving the positive sequence power flow problem and the three-phase power flow problem suffer from lack of robustness [9]-[10] i.e. the analyzed cases often diverge [9] or converge to non-physical or unacceptable solutions [30]. The 'PQV' based formulation for the positive sequence power flow problem is known to diverge or converge to non-physical solutions for ill-conditioned [4] and large scale (>50k buses) systems. Similarly, the backward-forward sweep method that was proposed to solve the radial and weakly meshed distribution systems with high R/X ratios [4] has difficulties converging for heavily meshed test cases with more than a single source [11] in the network. On the other hand, the CIM method [12]-[13] based on Dommel's work in 1970 [8] has challenges with incorporating multiple PV buses in the system.

The factors in all existing solution methodologies that are the most fundamental toward making these problems challenging are: i) the use of non-physical macro-models for modeling the power grid components, and ii) in the case of 'PQV' formulation, the use of power mismatch equations with real and reactive power as system state variables

to formulate the problem. The non-physical representations of the system equipment may not capture the true behavior of the model in the entire range of system operation. For example, an approximated macro-model for a generator that is represented via positive sequence or three-phase PV model can result in convergence to a low-voltage solution or divergence due to its quadratic voltage characteristics. Similarly, the inherent non-linearities in the 'PQV' formulation almost always cause divergence for large (>50k) and ill-conditioned test cases [35] when solved from an arbitrary set of initial conditions. This lack of a physics-based formulation, along with the methods that can constrain the non-physics based models in their physical space, is what renders the existing power flow and three-phase power flow problem and solution approaches to be "non-robust."

To develop a robust solver for the steady-state solution of the power grid, it is imperative that the solver can efficiently and effectively navigate through these challenges while converging to a solution that is both meaningful and correct. Most importantly, as previously discussed, the developed framework should be able to model both the transmission and distribution network without loss of generality. Intuitively and physically, both the transmission and distribution electric grid correspond to an electric circuit. Therefore, our approach toward solving the power flow and three-phase power flow problem is to treat them as such and solve both analyses using circuit simulation fundamentals. To achieve this we propose a two pronged approach: (i) the use of equivalent circuit formulation with true state variables of currents and voltages [31]-[33] to model both the transmission and distribution power grid networks, and (ii) the use of circuit simulation methods [34]-[35] to ensure robust convergence of these networks to correct physical solutions. Furthermore, the ability to model both transmission and distribution power grids as equivalent circuits allows us to combine the two without loss of generality in order to jointly solve them.

In the appendices of this report, we explore the physics-based approach to modeling of aggregated load in the system and conceptually demonstrate that consistent results can be obtained between the steady-state solution obtained from transient analysis as well as steady-state solution obtained from power flow analysis for the same test network via unification of the two analyses.

3.0 BACKGROUND AND PRIOR STATE OF THE ART

3.1 Positive Sequence and Three-Phase Power Flow Formulations

A power grid in its simplest form can be represented by a set of buses $\mathcal{N}$, where a set of generators $\mathcal{G}$ and load demands $\mathcal{L}$ are subsets of $\mathcal{N}$, which are further connected by a set of line elements, $\mathcal{T}_x$ and a set of transformers, xfmrs. Furthermore, there is a set of slack buses represented by $\mathcal{Z}$. In addition to these, the power grid may contain other elements, such as shunts, flexible alternating current transmission system (FACTS), etc. The aim of steady-state analysis of the power grid is to model the fundamental frequency component of the power grid and further solve for the complex voltages at its buses. The high voltage transmission network of the grid generally operates under balanced conditions, and therefore, the steady-state solution of the transmission network is obtained via positive sequence power flow models and analysis. In contrast, the distribution network of the power grid can operate under unbalanced conditions, therefore, three-phase power flow models and analyses are used to obtain its steady-state solution. In the following subsections, we discuss the current state of the art methods used for steady-state analysis of transmission and distribution networks and highlight their limitations.

3.2 ‘PQV’ based Formulation for Positive Sequence Power Flow Problem

The ‘PQV’ based power flow formulation is the industry standard for solving for the steady-state solution of the high voltage transmission network (i.e. power flow problem). In this formulation, a set of $2(|\mathcal{N}| - |\mathcal{Z}|) - |\mathcal{G}|$ power mismatch equations are solved for unknown complex voltage magnitudes and angles of the system nodes using the Newton Raphson (NR) method. The set of power mismatch equations is defined [3] as follows:

$$P_G^i - P_L^i = |V_i| \sum_{l=1}^{|\mathcal{N}|} |V_l| (G_{il}^Y \cos \theta_{il} + B_{il}^Y \sin \theta_{il}) \quad (1)$$

$$Q_G^i - Q_L^i = |V_i| \sum_{l=1}^{|\mathcal{N}|} |V_l| (G_{il}^Y \sin \theta_{il} - B_{il}^Y \cos \theta_{il}) \quad (2)$$

In order to solve for unknown complex voltages $\tilde{V}_i$ in the system, the real and reactive power mismatch equations given by (1)-(2) are solved for the set of $(|\mathcal{N}| - |\mathcal{Z}| - |\mathcal{G}|)$ buses in the system, whereas only real mismatch equations (1) are solved for the set of buses with generators $\mathcal{G}$ connected to it.

The ‘PQV’ formulation is inherently non-linear, since the set of network constraints (given by power mismatch equations) that is used to model the power grid is always non-linear independent of the physics of the models used. For example, in the ‘PQV’ formulation, a linear network consisting of linear models for slack bus, transmission lines and shunts would correspond to a non-linear set of power mismatch equations, a feature that can result in convergence difficulties for systems even trivial in size.

3.2.1 Current Injection Method for Three-Phase Power Flow Problem

The current injection method (CIM) is one of the most commonly used formulation for three-phase power flow problem [5]. It was proposed to address challenges in three-phase power flow problem while using the ‘PQV’ formulation and the backward-forward sweep method. In the CIM, the non-linear current mismatch equations for the system buses are solved via the NR method for all three phases with complex rectangular real and imaginary voltages as the unknown variables. The current mismatch equations for the three-phase power flow problem are defined as follows [5]:

$$\Delta I_{Ri}^{\Omega} = \frac{(P_i^{sp})^{\Omega} V_{Ri}^{\Omega} + (Q_i^{sp})^{\Omega} V_{Ii}^{\Omega}}{(V_{Ri}^{\Omega})^2 + (V_{Ii}^{\Omega})^2} - \sum_{l=1}^{|\mathcal{N}|} \sum_{t \in \Omega_{set}} (G_{il}^{\Omega t} V_{Ri}^t - B_{il}^{\Omega t} V_{Ii}^t) \quad (3)$$

$$\Delta I_{Ii}^{\Omega} = \frac{(P_i^{sp})^{\Omega} V_{Ii}^{\Omega} - (Q_i^{sp})^{\Omega} V_{Ri}^{\Omega}}{(V_{Ri}^{\Omega})^2 + (V_{Ii}^{\Omega})^2} - \sum_{l=1}^{|\mathcal{N}|} \sum_{t \in \Omega_{set}} (G_{il}^{\Omega t} V_{Ii}^t + B_{il}^{\Omega t} V_{Ri}^t) \quad (4)$$

Although, CIM is known to improve the convergence properties for the heavily and weakly meshed three-phase radial distribution systems with high R/X ratio, the method is known to diverge for test-cases with high penetration of PV buses [12]. Traditionally, this was not a problem as the number of PV buses in the distribution system were limited to a small number; however, with the advent of large-scale installation of distributed energy resources (DERs) and voltage control devices in the distribution system, this is no longer true. Therefore, it is essential that a standard three-phase power flow formulation can robustly handle high penetration of PV buses (any bus with voltage control) in the system.

3.2.2 Backward-Forward Sweep Method

The backward-forward sweep method was initially introduced in [4] to overcome the drawbacks of ‘PQV’ formulation-based NR method for three-phase radial systems with wide ranging line resistances and reactances. This method starts by breaking the interconnected meshed grid network into a radial grid network by introducing breakpoints. The reduced radial network in this method is then solved via Kirchhoff’s voltage and current laws. To compensate for the currents at the breakpoints, the method introduces current injections at the cut-set nodes. The magnitudes of these currents are calculated by iterative compensation methods [4], [18] that include a backward and a forward sweep step. Although, the method is known to work well with radial systems, it is prone to divergence for test cases that are highly meshed or have multiple sources [11].

3.2.3 Holomorphic Embedding Load Flow Method

Holomorphic embedding load flow (HELM) formulation [14] is a direct non-iterative method proposed to solve for positive-sequence power flow problem. The method works by embedding the non-analytic power mismatch equations into the larger set of analytic equations. The method is claimed to be robust as convergence for the problem is guaranteed [14] by application of continuation methods to the analytic set of equations. However, this method as originally presented in [14] lacks the ability to model PV buses and works exclusively for systems with PQ buses. Additional work toward extending the HELM formulation to incorporate PV buses is known to suffer from numerical issues [16]-[17]. Furthermore, during application of HELM method to practical large-scale problems, singular solution matrix is often encountered, and no theoretical guarantees have yet been provided to counter this challenge, thereby making this solution method non-robust.

3.2.4 Continuation Power Flow Method

The continuation power flow method is another approach for evaluating the steady-state operating point of the power grid [19]. This algorithm has been primarily used for assessing the voltage stability of the grid and to trace the maximum available transfer capability curve of the grid. The continuation power flow method involves iteratively evaluating the steady-state operating points of the grid using a two-step predictor and corrector algorithm until the critical operating point is obtained. The algorithm adds an additional state variable: the continuation factor and a corresponding equation to the formulation. In the beginning of this algorithm, conventional NR is used to obtain the base case solution for the analysis. The predictor step is then performed to obtain the new approximated solution for the complex voltages of the system and the continuation parameter variable. A modified NR is then performed to obtain the exact solution from the approximated solution in the corrector step. This iterative process is repeated until the critical point is found. The critical point is the solution state where the tangent vector for the continuation factor is zero, and it represents an infeasible or collapsed grid state [20]. The analysis methodology is quite useful in assessing the critical point of the electric grid and in general can solve for test grids operating close to the tip of the nose curve. However, the method requires a solved base case to start with, which itself can be quite challenging to solve for in case of hard-to-solve ill-conditioned and large test cases.

3.3 Circuit Simulation Methods

A standard circuit simulator tool (e.g. SPICE) models an integrated circuit using a set of linear and non-linear equations and generally employs the use of the damped Newton-Raphson (NR) method to find the solution of the corresponding equations. This iterative NR algorithm is guaranteed to have convergence with quadratic speed, if the following conditions are satisfied [21]:

- i. The functions represented by the set of non-linear equations, which in their real domain $\mathbb{R}^n$ must be continuous and smooth.
- ii. The solution for the set of non-linear equations from the iterative algorithm must be isolated.
- iii. The initial guess for the NR algorithm must be “sufficiently close” to the final solution.

For the purposes of circuit simulation, the first two are usually easily satisfied. Circuit models are generally created such that the underlying functions are continuous and smooth. To satisfy the second condition, non-isolated solutions must be avoided. The source of non-isolated solutions in the field of circuit simulation can be generally attributed to floating nodes or loops that are formed with components that act as short-circuits at DC condition (for example, voltage sources or inductors). All circuit simulator input files are usually sanitized or pre-processed to flag such network configurations, thereby eliminating the likelihood of non-isolated solutions. The real challenge then lies in finding an initial guess that is “sufficiently close” to the final solution. If the initial guess is not sufficiently close to the final solution, then a large step during a NR iteration could easily result in divergence or numerical overflow problems. Often with the use of the limiting methods in Section 3.3.1, some of these problems can be rectified, albeit at the cost of quadratic speed. However, for hard to solve highly non-linear circuit models, limiting methods by themselves are insufficient to ensure convergence. In such cases, homotopy methods such as those described in Section 3.3.2 are generally used to achieve convergence. With homotopy methods, an original problem is broken down into a sequence of sub-problems, wherein the solution to each prior sub-problem is used as the initial guess for subsequent sub-problem. Importantly, the first sub-problem solved with the homotopy algorithm is expected to have an initial guess that is trivial and sufficiently close to the final solution of the modified problem thereby satisfying the third condition for convergence of NR algorithm. In the rest of this section, we will discuss some of the commonly used limiting and homotopy methods in state-of-the-art circuit simulators.

3.3.1 Limiting methods

Limiting methods were first developed in [23]-[24] for simulation of diodes in the early 1970s. The purpose of these methods was to ensure that a large NR step does not lead to numerical overflow problems in the simulator due to the exponential nature of the diode IV curve. The “limiting” was performed on the diode state variables between the $(k + 1)^{th}$ and k^{th} iteration of NR to prevent overflow problems. In its simplest form, the implemented logic limited the diode voltage for the $(k + 1)^{th}$ NR iteration to $\hat{V}_{k+1}$ via expressions developed as a function of thermal voltage (V_t). The choice of limiting factors of $2V_t$ and $10V_t$ in the original SPICE tool was made from empirical observations [24], and is shown below:

Condition	Action
$ V_{k+1} - V_k \leq 2V_t$	$\hat{V}_{k+1} = V_{k+1}$
$V_{k+1} > V_{CRIT}$	$\hat{V}_{k+1} = V_{k+1}$
$V_{k+1} < V_k \ \& \ 10V_t < V_k$	$\hat{V}_{k+1} = V_{k+1} - 2V_t$
$V_k < V_{k+1} \ \& \ 10V_t < V_{k+1}$	$\hat{V}_{k+1} = \max(10V_t, V_k + 2V_t)$

Another limiting technique used in NR algorithms for diodes is based upon alternating bases and was proposed by Colon and implemented by Kao [22]. This algorithm was further modified by Nagel. In this algorithm, a NR iteration based on current variables is performed instead of voltage variables if the diode conductance for the new iterate has a slope magnitude greater than a pre-specified value. The slope itself is directly related to the voltage across the diode, and therefore a critical voltage (V_{CRIT}) can be pre-defined and used as a boundary condition. Nagel [24] found that a near optimal value of V_{CRIT} is obtained when the voltage has minimum radius of convergence, given as:

$$V_{CRIT} = V_t \ln \left(\frac{V_t}{\sqrt{2I_s}} \right) \quad (5)$$

where I_s is the diode saturation current. This algorithm has the following logic [22]:

Condition	Action
$V_{k+1} \leq V_{CRIT}$	$\hat{V}_{k+1} = V_{k+1}$
$V_{k+1} > V_{CRIT}$	$\hat{V}_{k+1} = V_{k+1} + V_t \ln \left(\frac{V_{k+1} - V_k}{V_t} + 1.0 \right)$

3.3.2 Homotopy Methods

Homotopy methods are generally applied when limiting methods fail to ensure convergence. Many such homotopy methods are included in the state-of-the-art circuit simulators, amongst which, the two that are most commonly used are described below.

3.3.2.1 Gmin stepping

Gmin stepping is a combined algorithm whose implementation itself comes from a combined logic of multiple sources [28]. The method requires addition of variable conductances to each node in the system and is explain in detail by following sequence of steps:

- i. To solve for the DC solution of an integrated circuit, first a large conductance (Gmin) is connected from every node in the system to ground. This essentially “swamps” any larger resistances in the system and ensures that the circuit solution at 0th iteration has circuit node voltages very close to the value of zero. The solution to this modified system is trivial and can be found by solving the NR loop with initial system voltages set to 0.

- ii. The value of G_{min} is then gradually stepped down in subsequent sub-problems until a value close to zero for G_{min} is achieved. The initial condition for subsequent sub-problem is obtained from the final solution of the prior sub-problem thereby resulting in an “ever-so slight change” in the solutions between subsequent sub-problems.
- iii. The final solution obtained at G_{min} value of zero is the DC solution of the original circuit.

3.3.2.2 Source Stepping

An alternate homotopy technique that can be applied to circuits that have independent sources is called source stepping. The algorithm for source stepping is described in the following steps:

- i. Initially all the independent sources in the systems are turned off, thereby resulting in a trivial solution for the modified problem, wherein all the system voltage and current magnitudes are zero.
- ii. The independent sources are then gradually turned back on to their original values in small incremental steps resulting in a sequence of sub-problems. As in the case of G_{min} stepping, the initial condition for each subsequent sub-problem is obtained from the final solution of the prior sub-problem.
- iii. The final solution, which is the DC solution for the circuit, is obtained when all independent sources in the system are scaled all the way up to their original value.

Unfortunately, this method is only suited for certain types of circuit simulation problems. Simulation of a digital circuit is an example for which source stepping is not suited. During simulation of such a circuit, the turn-on of a digital transistor circuit due to a small change in voltage can cause a sudden change in circuit state, thereby resulting in convergence issues [28] while using this homotopy technique.

Other heuristics used to ensure robust convergence of circuit simulations are well documented in [21]-[27].

4.0 METHODS, ASSUMPTIONS, AND PROCEDURES

4.1 Equivalent Circuit Approach

We proposed the equivalent circuit approach [31]-[35] for steady-state analysis of the power grid to address challenges exhibited by the existing formulations. This approach for generalized modeling of the power system in steady-state (i.e. power flow and three-phase power flow) represents both the transmission and distribution power grid in terms of equivalent circuit elements without loss of generality. It was shown that each of the power system components (including constant power models, i.e. PQ and PV buses) can be directly mapped to an equivalent circuit model based on the underlying relationship between current and voltage state variables. Importantly, this formulation can represent any physics based model or measurement based semi-empirical model as a sub-circuit, as shown in [50]-[53], and these models can be combined hierarchically with other circuit abstractions to build larger aggregated models. In addition, by modeling both the transmission and distribution system equivalently using circuit fundamentals, the equivalent circuit approach allows for the combined simulation of transmission and distribution systems, a framework intractable with existing solution methods due to the use of disparate formulations and models for analysis of transmission as well as distribution systems. In the following sub-sections, we derive the equivalent circuit models for the most common network elements used in the positive sequence power flow and the three-phase power flow problems. Then, in Section 4.1.5, we demonstrate, with three-phase induction motor (IM) as an example, the development of an equivalent circuit model from the physics based first principles behavior. Appendix A introduces a new empirical load model for aggregated load in the grid that can capture the true sensitivity of the modeled load and can be fitted with available measurement data.

4.1.1 Split-Circuit Formulation

In the equivalent circuit approach, we represent the equivalent circuit models for different power grid components in the network using the current and voltage state variables. For positive sequence power flow and three-phase power flow analyses this translates to linear and non-linear functions of complex voltage and complex current state variables.

The grid components that are represented by linear functions of complex voltages and complex currents are directly mapped into an equivalent circuit using basic circuit elements such as the conductance, susceptance, independent and controlled sources. However, in case of non-linear representations of grid components, the equivalent circuit models are constructed by first linearizing the set of non-linear equations using Taylor's expansion and then representing them using a fundamental set of circuit elements. Importantly, Taylor expansion of these non-linear representations require the computation of their first-order derivative. This is not possible for complex terms with

conjugate operator, such as the ones observed in PV and PQ node fundamental constraints. This is due to the non-analyticity of these terms, because of which, they are non-differentiable. Therefore, to circumvent this challenge of non-analyticity, the derived equivalent circuit models for power flow and three-phase power flow problem are split into two coupled circuits: one real, and one imaginary [31], both of which are analytic and differentiable.

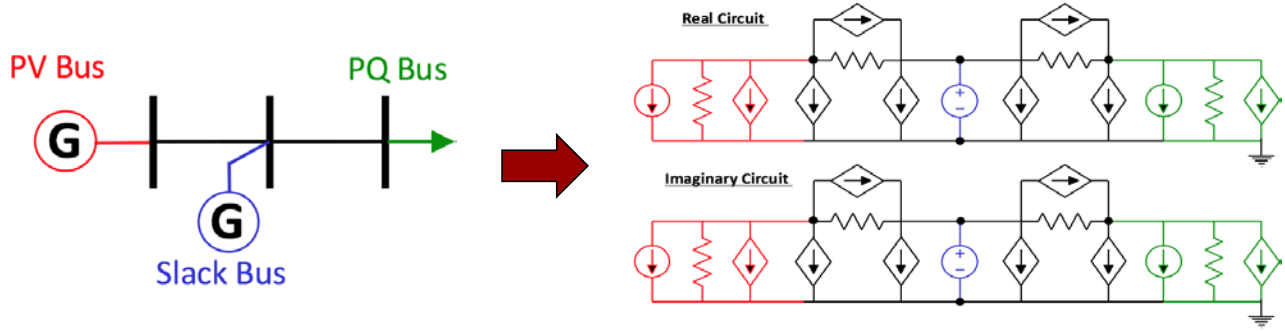


Figure 1: Simple three-bus power flow network and its corresponding equivalent circuit.

For example, consider a simple three-bus power flow network given in the left of the Figure 1. To represent the equivalent circuit model for this network, we derive the corresponding real and imaginary equivalent circuits, as shown in the right hand of Figure 1. In the following subsections, we will derive the real and imaginary equivalent circuit models for the most common power flow and three-phase power flow models, a methodology that can be easily extended to derive an equivalent circuit for any future grid component.

4.1.2 Equivalent Circuit Models for the Positive Sequence Power Flow Problem

4.1.2.1 PV Bus

The PV bus model in the positive sequence power flow problem is used to represent the aggregated characteristics of a generator that is required to hold its real power and voltage magnitude constant, as described here in terms of rectangular current-voltage state variables:

$$P_G = V_{RG}I_{RG} + V_{IG}I_{IG} \quad (6)$$

$$|V_G|^2 = V_{RG}^2 + V_{IG}^2 \quad (7)$$

In order to derive the equivalent circuit model for the PV bus behavior described by (6)-(7), we have a choice to model the PV node as a complex voltage source (as functions of complex current) [31] or a complex current source (as functions of complex voltage) [33]. The equations that represent the generator PV bus as a set of complex voltages sources are given by:

$$V_{RG} = \frac{P_G I_{RG} \pm I_{IG} \sqrt{V_G^2 (I_{RG}^2 + I_{IG}^2) - P_G^2}}{I_{RG}^2 + I_{IG}^2} \quad (8)$$

$$V_{IG} = \frac{P_G I_{IG} \pm I_{RG} \sqrt{V_G^2 (I_{RG}^2 + I_{IG}^2) - P_G^2}}{I_{RG}^2 + I_{IG}^2} \quad (9)$$

This model for PV nodes is known to have convergence issues [33] due to the existence of the square root term in equations (8)-(9) that can result in non-real values for derived complex voltage sources, if a negative term within the square root term is encountered. To address this problem, we instead model the PV node as complex current sources to mimic the characteristic behavior of equations (6)-(7). This offers superior convergence when applying NR iterations to the resulting equation system [33].

The split real and imaginary equations for complex current sources for a PV node are given by:

$$I_{RG} = \frac{P_G V_{RG} + Q_G V_{IG}}{V_{RG}^2 + V_{IG}^2} \quad (10)$$

$$I_{IG} = \frac{P_G V_{IG} - Q_G V_{RG}}{V_{RG}^2 + V_{IG}^2} \quad (11)$$

In addition to producing real power constrained generator currents, the PV node also controls the voltage magnitude either at its own node or any other remote node in the system given its reactive power output is within the limits. We represent this constraint by a control circuit, as shown in the following subsection. The reactive power Q_G of the generator is controllable and acts as the additional unknown variable for the additional constraint that is introduced due to voltage control.

To derive the equivalent circuit of the PV node, the first order terms of the Taylor expansions for (10)-(11) are used to linearize the functions, as shown in Figure 2. Linearization of the real generator current results in the following terms:

$$\begin{aligned} I_{RG}^{k+1} = & \frac{\partial I_{RG}}{\partial Q_G} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (Q_G^{k+1}) + \frac{\partial I_{RG}}{\partial V_{RG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{RG}^{k+1}) + \frac{\partial I_{RG}}{\partial V_{IG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{IG}^{k+1}) + I_{RG}^k \\ & - \frac{\partial I_{RG}}{\partial Q_G} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (Q_G^k) - \frac{\partial I_{RG}}{\partial V_{RG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{RG}^k) - \frac{\partial I_{RG}}{\partial V_{IG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{IG}^k) \end{aligned} \quad (12)$$

Similarly, linearization by Taylor's expansion for imaginary current terms results in:

$$\begin{aligned} I_{IG}^{k+1} = & \frac{\partial I_{IG}}{\partial Q_G} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (Q_G^{k+1}) + \frac{\partial I_{IG}}{\partial V_{RG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{RG}^{k+1}) + \frac{\partial I_{IG}}{\partial V_{IG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{IG}^{k+1}) + I_{IG}^k \\ & - \frac{\partial I_{IG}}{\partial Q_G} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (Q_G^k) - \frac{\partial I_{IG}}{\partial V_{RG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{RG}^k) - \frac{\partial I_{IG}}{\partial V_{IG}} \big|_{Q_G^k, V_{RG}^k, V_{IG}^k} (V_{IG}^k) \end{aligned} \quad (13)$$

The terms in (12) represent the basic elements of the real circuit of PV node. The first term in (12) represents a current source that is a function of the reactive power; the second term represents a conductance, since the real current is proportional to the real voltage; the third term represents a voltage-controlled current source, since the real current is proportional to the imaginary voltage. The remaining terms are all dependent on known values from the previous iteration, so they can be lumped together and represented as an independent current source.

Similarly, the terms in (13) represent the basic circuit elements for the imaginary circuit of the PV node. The first term in (13) represents a current source that is a function of reactive power; the second term represents a voltage-controlled current source, since the imaginary current is proportional to the real voltage; the third term represents a conductance, since the imaginary current is proportional to the imaginary voltage. The remaining terms are all dependent on known values from the previous iteration, so they can be lumped together and be represented as an independent current source as in the case of real circuit.

Figure 2 represents the derived real and imaginary circuits for a complex current source that represents the behavior of a PV node.

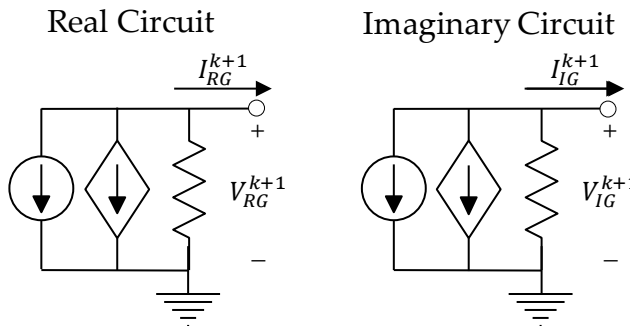


Figure 2: Equivalent circuit model for PV generator model.

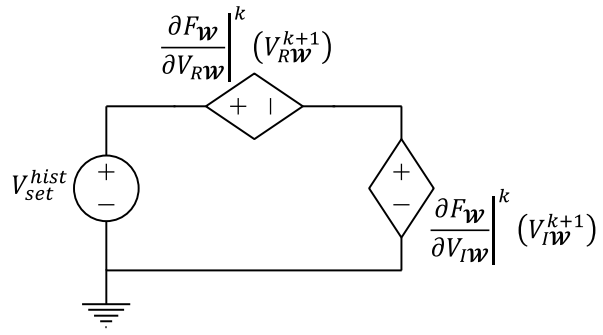


Figure 3: Voltage magnitude constraint control equivalent circuit.

4.1.2.2 Voltage Regulation of the Bus

Numerous power grid elements such as generators, FACTS devices, transformers, shunts, etc., can control a voltage magnitude at a given node in the system. Moreover, they can control the voltage magnitude at either their own node ($\mathcal{O}$) or a remote node ($\mathcal{R}$) in the system. In the equivalent circuit formulation, the control of the voltage magnitude is achieved through a control circuit (Figure 3) with following mathematical behavior:

$$F_W \equiv V_{set}^2 - V_{RW}^2 - V_{IW}^2 = 0, \text{ where } W \in \mathcal{O}, \mathcal{R} \quad (14)$$

The circuit in Figure 3 is derived from the linearized version of (14). It is stamped into the matrix equations for each node W in the system whose voltage is being controlled such that there exists at least one path (whose vertices do not consist of a voltage-

controlled node) between the node $\mathcal{W}$ and the equipment's node $\mathcal{O}$ that is controlling it. The additional unknown variable for this additional constraint is dependent on the power system device that is controlling the voltage magnitude. For example, the additional unknown variable for a generator is its reactive power Q_G , whereas in the case of transformers, it is the transformer turns ratio tr , and for FACTS devices it is the firing angle φ . The previous section showed how the additional unknown variable for PV buses is integrated in its respective equivalent circuit.

Reactive power limits of a PV generator model

The PV model for the generator derived in Section 4.1.2.1 does not account for reactive power limits. However, all practical power flow solvers must enforce these limits to conserve the physics of the devices. Majority of the existing commercial tools enforce the generator reactive power limits via the use of discontinuous piecewise linear models (PV-PQ switching). Other approaches that do not use discontinuous piecewise models for enforcing the limits have also been researched extensively in academia. Of these, the most promising ones [36]-[39] use mixed complementarity constraints to represent the generator behavior with limits. This approach in general requires an optimization framework to enforce generator limits. This significantly increases the size as well as the complexity of the power flow problem thereby making the problem more prone to divergence. Modifications to this approach [38] have incorporated the complementarity constraints within the existing power flow framework; however, the approach has observed poor convergence, with the latest papers [38] on the approach limited to solving system sizes of less than a thousand nodes.

Here we further discuss discontinuous piecewise model for generator as it is the state-of-the-art in commercial tools. In this model, the generator has two discrete modes of operation: (i) voltage control mode – in this mode the reactive power of the voltage equipment is within its limits and the active set of equations include (14) (ii) set reactive power mode – in this mode the reactive power is either set to its maximum or minimum limit and the active set of constraints include (15) or (16).

$$\text{if } \hat{Q}_G > Q_{MAX} \text{ then, } Q_G = Q_{MAX} \quad (15)$$

$$\text{if } \hat{Q}_G < Q_{MIN} \text{ then, } Q_G = Q_{MIN} \quad (16)$$

To enforce reactive power limits for the PV model in the power flow analyses via the use of discontinuous models, the PV node model is switched between the voltage control mode and the set reactive power mode in the outer loop of the NR solver depending on the obtained value of reactive power ($\hat{Q}_G$) in the inner loop of the NR solver. This methodology generally tends to work well for small cases but is known to suffer from convergence issues for larger problems.

A necessary condition for guaranteed convergence of a system with discontinuous piecewise models requires that only one element/model at a time is switched to a different

piecewise linear segment per single NR iteration [45]-[46], an intractable approach while solving a larger system wherein a significant number of generator violate their limits. Therefore, due to this practical consideration, the existing state-of-the-art tools limit a larger number of generators at once. This approach can result in oscillations during NR that could further prevent convergence of the overall system. The following example demonstrates this behavior.

In this example, positive-sequence power flow simulation is run for a network case that represents a real electric grid network in Africa [47]. During simulation of this network, several generators violate their limits once the inner loop of the NR loop converges. Upon applying PV-PQ switching in the outer loop, oscillatory behavior is observed as seen in the Figure 4. The vertical axis of the figure shows the number of generators that are limited in the outer loop of NR iterations, and the horizontal axis represents the outer loop count for the power flow analyses. As seen in the figure, due to the observed oscillatory behavior, overall convergence for the test case is not achieved.

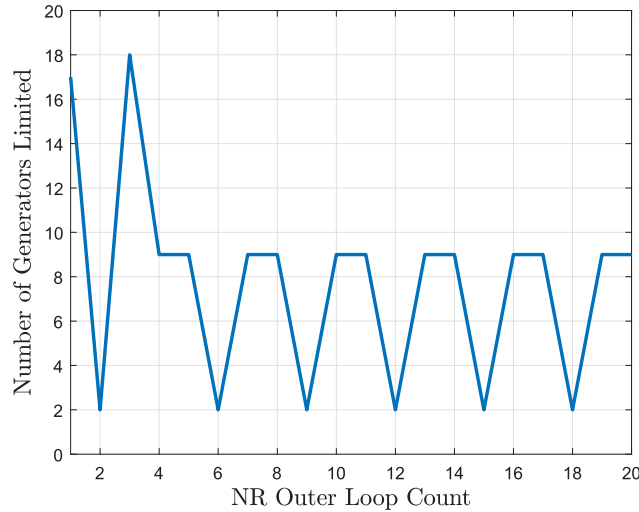


Figure 4: Oscillations observed during PV-PQ switching in the outer loop of NR.

To address this limitation wherein oscillatory behavior is observed during PV-PQ switching, we explore two distinct approaches:

- i. Setting a pre-determined parameter that is the maximum allowable count of generator switching between PV and PQ modes, a number that if any generator exceeds results in reactive power output Q_G of that generator to be permanently set to its last encountered limit value.
- ii. The use of a continuous and twice differentiable model for the generator voltage control.

The first approach is briefly discussed prior to introducing the continuous twice differentiable generator model. In this approach, a fixed pre-determined parameter (integer number) is inputted by the user that is applied in the algorithm to prevent the

generator oscillations between different discontinuous piecewise segments. The algorithm is implemented to count the number of occurrences of generator back-offs during the simulation. In case this number exceeds the specified parameter, the generator is no longer allowed to back-off and its reactive power Q_G is limited to either Q_{MAX} or Q_{MIN} for rest of the simulation, hence preventing any further oscillations between different segments. Even though this approach has been shown to prevent oscillations in many test instances, convergence is not always guaranteed [40]-[41]. This is primarily due to the shrinking of the solution space once the generator reactive power is permanently limited, which can lead to divergence due to the lack of solution in the reduced solution space. Furthermore, this can result in convergence for the generator model in an unstable state:

- i. A generator's reactive power Q_G is set to its lower reactive power limit (Q_{MIN}) while the magnitude of voltage at the controlled node is lower than the set voltage, or
- ii. A generator's reactive power Q_G is set to its higher reactive power limit (Q_{MAX}) while the magnitude of the voltage at the controlled node is higher than the set voltage.

Hence, to prevent the generator oscillations during PV-PQ switching while ensuring that the generator model converges to a stable operating point, we propose the use of the following continuous generator model. Continuous models have been previously proposed for use within power flow problem [42]-[43]. However, most of these methods were unable to solve cases larger than a few hundred nodes and were further unable to account for the remote generator buses. In the following section, we discuss a holistic approach for use of continuous models within the equivalent circuit approach that when combined with homotopy methods can ensure robust convergence for large scale realistic systems.

4.1.2.3 Continuous Model for a Generator/PV Bus

General Introduction

To address the limitations of the discontinuous piecewise generator model, we utilize a continuous and twice differentiable generator model [44]. In this model, we model the generator voltage constraint using a non-linear sigmoid function that can control the voltage of the controlled node when the reactive power of the generator is within its limits and can limit the generator reactive power when one of its limits are violated. Importantly this model is continuous and twice differentiable and does not require discontinuous switching between piecewise sections thereby preventing oscillations that otherwise can be detrimental for system convergence. Furthermore, the continuous model allows for the application of robust methods such as homotopy methods to achieve convergence during NR. Methods such as these often require that the set of network models to be continuous, which was not the case with discontinuous piecewise models.

Description of Models

The continuous model for the PV node models the generator currents as a function of complex voltages, as was in the case of the discontinuous piecewise model as shown in (10)-(11). However, this model replaces the voltage constraint given by (14) with a sigmoid function given in (17). Importantly, the model equations for this model are both continuous and twice differentiable and include the reactive power limits for the generator inherently.

$$Q_G = \frac{Q_{MAX} - Q_{MIN}}{1 + \exp(\lambda_S [\sqrt{V_{RW}^2 + V_{IW}^2} - V_{set}])} + Q_{MIN} \quad (17)$$

The sigmoid function in (17) mimics the behavior of the PV node during both the “voltage control mode” and “set reactive power mode”. In case, the reactive power needed to control the voltage of the controlled node is within its limits, the model operates in the steep slope part of the curve in Figure 5 thereby maintaining the controlled node voltage to its set value. In case, the reactive power needed to control the controlled node’s voltage exceeds its limits, the model saturates the reactive power output thereby no longer controlling the controlled node’s voltage. Importantly, the smoothing parameter (λ_S) in (17) controls the slope of the generator model voltage characteristics as shown in Figure 5. A higher value for this parameter tends to better mimic the behavior of the piecewise discontinuous generator model at the cost of more rigid non-linear behavior. In contrast, reducing the magnitude of this parameter relaxes the function non-linearities while approximating the voltage control behavior of the existing generator model. Due to the highly non-linear nature of this sigmoid model, convergence difficulties are observed for large as well as ill-conditioned systems. Therefore, in the following section we develop techniques to achieve robust convergence for the network equations while using this continuous generator model.

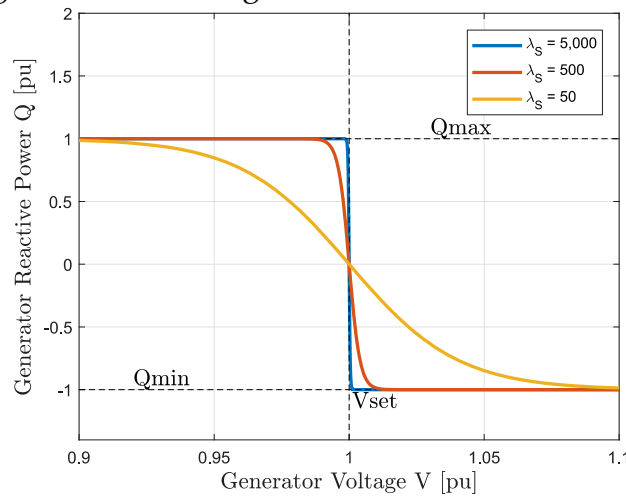


Figure 5: Voltage constraint behavior for continuous generator model.

Aid to Convergence

We propose two homotopy based approaches to achieve robust convergence for power flow test cases that include continuous models to represent generator behavior. Homotopy methods for general application to power flow and three-phase power flow analyses are discussed in depth in Section 4.2.2. However, due to the applicability of these methods to continuous generator model convergence, they are briefly discussed here.

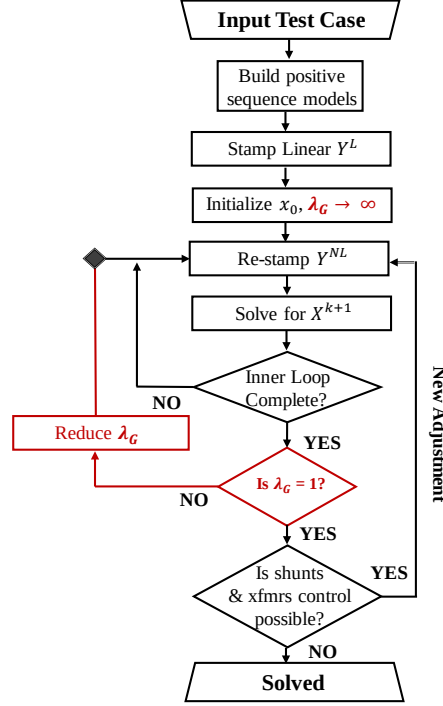
Relaxation of generator convergence parameter to enable robust convergence

To ensure the robust convergence for the continuous generator model, in the first approach, we embed a generator convergence parameter λ_G in the continuous generator model:

$$Q_G = \frac{\lambda_G(Q_{MAX} - Q_{MIN})}{1 + \exp(\lambda_S [\sqrt{V_{RW}^2 + V_{IW}^2} - V_{set}])} + \lambda_G Q_{MIN} \quad (18)$$

To use the properties of homotopy methods to achieve robust convergence, we first need to obtain the value of λ_G that will result in a trivial solution for the generator continuous model. This is achieved by calculating the initial value of generator convergence parameter λ_G^{init} via solving the inner loop of the power flow problem with generator models that have unbounded reactive power limits and choosing its value such that:

$$\lambda_G^{init} = \begin{cases} \frac{Q_G}{Q_{MAX}}, & \text{if } Q_G > Q_{MAX} \\ \frac{Q_G}{Q_{MIN}}, & \text{if } Q_G < Q_{MIN} \\ 1, & \text{otherwise} \end{cases} \quad (19)$$



Algorithm 4-1: Flowchart for dynamic handling of generator convergence parameter for better convergence.

Once we have obtained an initial value (λ_G^{init}) for the parameter that results in a trivial solution for the continuous generator model, we vary the parameter in small increments until the original problem is solved. As always is the case with the use of homotopy methods, the final solution of the previous sub-problem is chosen as the initial guess for subsequent sub-problem.

Algorithm 4-1 describes the general flow of this homotopy method. First, for all generators in the network, generator convergence parameter λ_G^{init} is initialized via (19). Following which, for each successful convergence of the inner loop of NR, the generator convergence parameter is incrementally varied until the value of unity is achieved for all λ_G . The range of generator convergence parameter λ_G is given by $[1, \infty)$. Figure 6 graphically demonstrates how the continuous PV model evolves along the homotopy path.

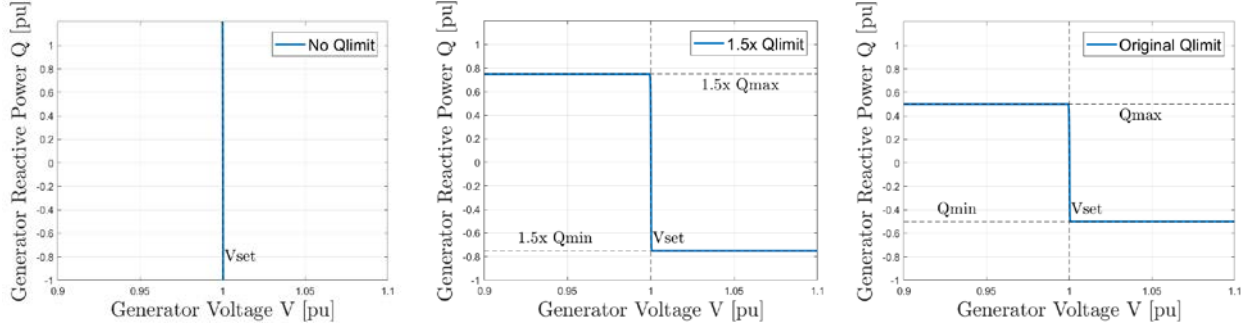


Figure 6: Generator characteristics as a function of generator convergence parameter.

Relaxation of generator smoothing parameter for achieving robust convergence

In this approach, generator smoothness parameter (λ_s) in (17) is adjusted if convergence difficulties are encountered due to the rigid non-linearities in the continuous model. The highly non-linear nature of the continuous generator model voltage constraint in the region around Q_{MAX} and Q_{MIN} is due to the steep change in the gradient of the function. To relax these non-linearities to enable smoother convergence requires adjustment of the generator smoothness factor, as shown in Figure 7.

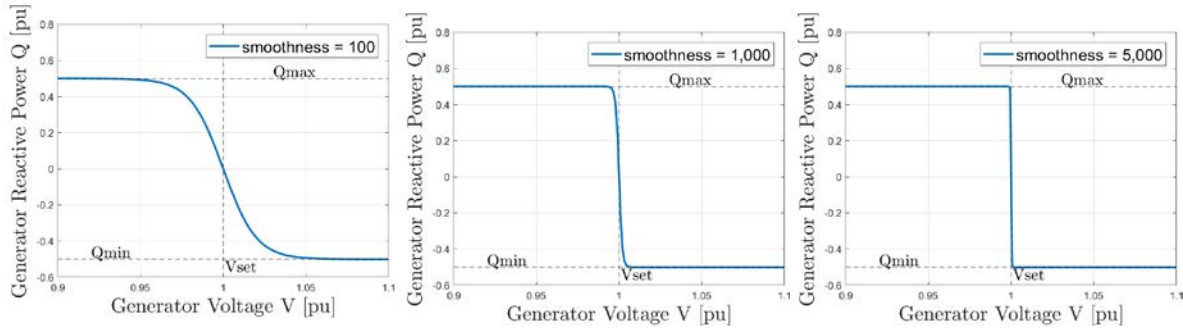
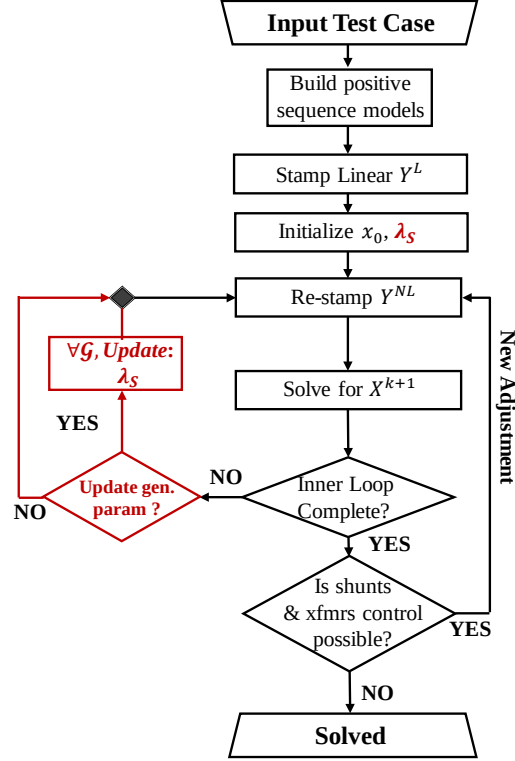


Figure 7: Generator characteristics as a function of generator smoothness parameter.

Reducing the magnitude of the generator smoothness parameter relaxes the non-linearities in generator model, whereas increasing the magnitude mimics the piecewise behavior of conventional generator model. Therefore, if convergence difficulties are encountered in the inner loop of NR, then the generator smoothness parameter is first relaxed until convergence for the continuous model is achieved (representing the trivial problem within the homotopy method). The parameter is then gradually scaled back up until the original model is solved as in the case of any homotopy method. The Algorithm 4-2 depicts the flow of this approach.



Algorithm 4-2: Flowchart for dynamic handling of generator smoothness parameter for better convergence.

Results for the Continuous Generator

Experiment 1

The purpose of this experiment is to demonstrate that the use of the continuous generator model with proposed homotopy methods can eliminate the limitations of the piecewise discontinuous generator model as shown in Section 0. In this experiment, we consider the test network case for the grid in Africa that was previously discussed in Section 0. This case when solved with the discontinuous piecewise generator model resulted in oscillations during PV-PQ switching in the outer loop of the power flow solver as shown in Figure 4 and hence, the solution for the test case could not be obtained.

However, when solved with the use of continuous generator model along with methods that aids its convergence, oscillations were easily prevented with successful convergence for the example test case.

4.1.2.4 Slack Bus

Slack bus model is used in power flow analysis primarily for two purposes. First, it provides the reference angle for the power grid circuit and second, it absorbs or produces any slack (power losses and load-generation mismatch in the system) in the system. In its most basic form, the model is the easiest bus type to model. In the real circuit, it appears as an independent voltage source of value $|V_i| \cos \theta_i$, and in the imaginary circuit it

appears as a voltage source of value $|V_i| \sin \theta_i$. When the phase θ is 0° the imaginary component appears as a short to ground.

In real life, however, no single generator covers the complete slack in the system. Generally, all the generators change their real-power set-points based on the primary droop control and furthermore a subset of them adjust their set-points based on the secondary control i.e. automatic generation control (AGC) signal. Therefore, it is important that the steady-state analysis, which is trying to mimic the true behavior of the grid during normal or contingency operation, is able to model the distributed slack behavior of the grid. We incorporate this approach within our formulation using continuous analytical models for droop control as well as for AGC [44]. Although, droop control and AGC represent two distinct phenomena within the grid operation, their models for power flow problem can be achieved via same set of equations. We achieve this by adding an additional variable delta P (ΔP_G) to generators that are participating in either droop control or AGC. These variables in turn share the slack in the grid based on the pre-defined vector of participation factors κ , which in turn can be pre-calculated based on either the size of the generator or the inertia of the machine. The equations for the distributed slack operation are as follows:

$$P_{slack} + \Delta P_{slack} = V_{slack}^R I_{slack}^R + V_{slack}^I I_{slack}^I \quad (20)$$

$$\Delta P_G = \kappa^T \Delta P_{slack} \quad \forall G \in \{\text{AGC/droop}\} \quad (21)$$

$$P_G^{MIN} < P_G + \Delta P_G < P_G^{MAX} \quad (22)$$

where ΔP_{slack} is a diagonal matrix with each element as ΔP_{slack} and ΔP_G is the vector of additional power produced by the generators participating in AGC or droop control.

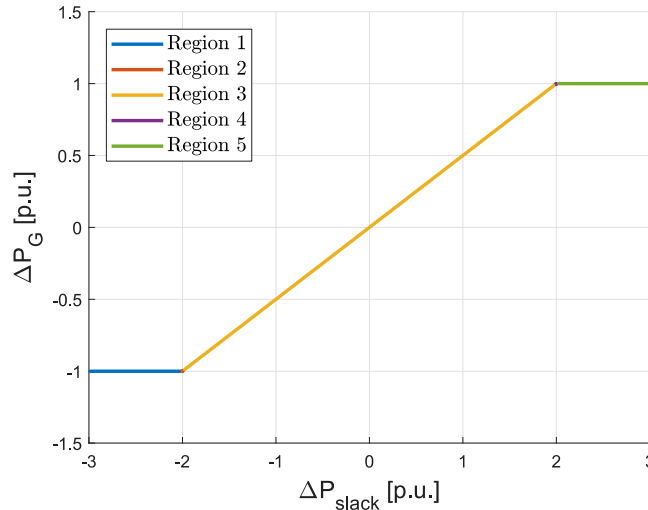


Figure 8: Continuous analytical model for modeling the AGC and droop control of the generator based on participation factor.

For modeling AGC as well as droop control, we represent the constraints given by (21) and (22) through the use of continuous analytical model whose behavior is seen in Figure 8. The model adjusts the real power output (P_G) of the generator based on pre-defined participation factor as well as slack in the system until it exceeds its limits. The model consists of a set of functions that together include three linear segments (Region 1, 3, and 5 in the Figure 8) patched with two quadratic segments (Region 2 and 4 in the Figure 8) to produce a net continuous differentiable function [44]. With the use of this model, all or selected sub-set of the participating generators contribute toward the slack in the system until they hit their minimum or maximum limit. Importantly, the model is differentiable and continuous and hence can be directly included implicitly within the inner loop of NR solver thus utilizing all the circuit heuristics developed within this project to ensure robust convergence. This is a significant improvement over the existing methodologies for AGC and droop in commercial power flow tools that implement this feature using outer loop around the NR solver with discontinuous piecewise models.

To demonstrate our approach for AGC and droop control we run an experiment on a sample 23-node (savnw) system. We first run the base case (pre-contingency) without enabling AGC or droop control and note the real power generation for different generators. We then perform a N-1 contingency on the base case by taking the generator on bus 211 off-line and further noting the updated real power generation for different generators: i) with AGC and droop control enabled ii) with AGC and droop control disabled. As tabulated in Table 1, when the generator contingency is performed with AGC disabled, the slack generator picks up all the real power generation mismatch due to the loss of generator on bus 211. However, in the case with AGC enabled, the real power generation is distributed amongst different generators (in AGC) based on the participation factors until they hit their limits, upon which the participating factors are re-distributed automatically, and remaining generators share the remaining slack.

Table 1: Results to demonstrate AGC functionality using continuous analytical model.

Generator ID	P_G^{MAX}	P_G^{MIN}	κ	Real Power Generation [MW]		
	[MW]	[MW]		Pre-contingency	Post-contingency	
				AGC-Disabled	AGC-Enabled	AGC-Disabled
101	810	0	0.23	750	810	750
102	810	0	0.23	750	810	750
206	900	0	0.25	800	900	800
211*	616	0	0.18	600	0*	0*
3011	900	0	0.08	257.74	635.22	864.39
3018	117	0	0.03	100	117	100

*Generator taken off-line during a contingency

4.1.2.5 PQ Bus

Like the PV bus, the constant power node (PQ bus) is also represented as an equivalent circuit via either a complex voltage source or a complex current source. It has been empirically determined that superior convergence is observed when the load bus is modeled as a complex current source. The two fundamental equations that represent the behavior of the PQ load model, are given by:

$$P_L = V_{RL}I_{RL} + V_{IL}I_{IL} \quad (23)$$

$$Q_L = V_{RL}I_{IL} - V_{IL}I_{RL} \quad (24)$$

The terms in equation (23)-(24) are re-arranged to derive the complex current sources of the PQ node as a function of complex voltage state variables:

$$I_{RL} = \frac{P_L V_{RL} + Q_L V_{IL}}{V_{RL}^2 + V_{IL}^2} \quad (25)$$

$$I_{IL} = \frac{P_L V_{IL} - Q_L V_{RL}}{V_{RL}^2 + V_{IL}^2} \quad (26)$$

Linearizing the load model in (25) and (26) as shown in (27)-(28) via Taylor expansion results in three elements in parallel for both the real and imaginary circuits: a conductance, a voltage-controlled current source, and an independent current source.

$$I_{RL}^{k+1} = \frac{\partial I_{RL}}{\partial V_{RL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{RL}^{k+1}) + \frac{\partial I_{RL}}{\partial V_{IL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{IL}^{k+1}) + I_{RL}^k - \frac{\partial I_{RL}}{\partial V_{RL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{RL}^k) - \frac{\partial I_{RL}}{\partial V_{IL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{IL}^k) \quad (27)$$

$$I_{IL}^{k+1} = \frac{\partial I_{IL}}{\partial V_{RL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{RL}^{k+1}) + \frac{\partial I_{IL}}{\partial V_{IL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{IL}^{k+1}) + I_{IL}^k - \frac{\partial I_{IL}}{\partial V_{RL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{RL}^k) - \frac{\partial I_{IL}}{\partial V_{IL}} \big|_{Q_L^k, V_{RL}^k, V_{IL}^k} (V_{IL}^k) \quad (28)$$

The linearized elements in (27)-(28) are represented in Figure 9 to represent the split equivalent circuit for the PQ load model.

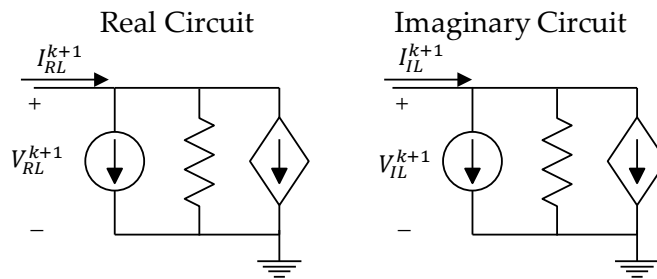


Figure 9: Equivalent split-circuit PQ load model.

4.1.2.6 ZIP Model

The currents consumed by the PQ load model are insensitive to voltage magnitude at its terminal. This can result in inaccurate results for the power flow analysis for cases where the system solution has either visibly high or low voltages. Therefore, a more comprehensive load model such as the ZIP load model is needed to capture the voltage sensitive nature of the aggregated load. The ZIP load model models the aggregated load in the system as a mix of constant impedance, constant current, and constant power load models that can be mathematically represented as follows:

$$P_i^{ZIP} = Z_P(|V_i|)^2 + I_P(|V_i|) + S_P \quad (29)$$

$$Q_i^{ZIP} = Z_Q(|V_i|)^2 + I_Q(|V_i|) + S_Q \quad (30)$$

In the equivalent circuit approach, the equations for the ZIP load model can be re-written as:

$$I_{Ri}^{ZIP} = Z_P V_{Ri} - Z_Q V_{Li} + \frac{S_P V_{Ri} + S_Q V_{Li}}{(V_{Ri})^2 + (V_{Li})^2} + \left(\sqrt{I_P^2 + I_Q^2} \right) \cdot \cos(I_{pf} + \delta_i) \quad (31)$$

$$I_{Li}^{ZIP} = Z_P V_{Li} + Z_Q V_{Ri} + \frac{S_P V_{Li} - S_Q V_{Ri}}{(V_{Ri})^2 + (V_{Li})^2} + \left(\sqrt{I_P^2 + I_Q^2} \right) \cdot \sin(I_{pf} + \delta_i) \quad (32)$$

where:

$$I_{pf} = \tan^{-1} \left(\frac{I_Q}{I_P} \right) \quad (33)$$

$$\delta_i = \tan^{-1} \left(\frac{V_{Li}}{V_{Ri}} \right) \quad (34)$$

For the load model given in (31) through (34), the constant impedance part of the load is linear, whereas the constant current and constant power part of the aggregated load is nonlinear. Linearizing the set of equations using Taylor expansion results in the following expressions:

$$I_{Ri}^{ZIP^{k+1}} = I_{Ri}^{ZIP^k} + \left(\frac{\partial I_{Ri}^{ZIP}}{\partial V_{Ri}} \right)_{V_{Ri}^k, V_{Li}^k} (V_{Ri}^{k+1} - V_{Ri}^k) + \left(\frac{\partial I_{Ri}^{ZIP}}{\partial V_{Li}} \right)_{V_{Ri}^k, V_{Li}^k} (V_{Li}^{k+1} - V_{Li}^k) \quad (35)$$

$$I_{Li}^{ZIP^{k+1}} = I_{Li}^{ZIP^k} + \left(\frac{\partial I_{Li}^{ZIP}}{\partial V_{Ri}} \right)_{V_{Ri}^k, V_{Li}^k} (V_{Ri}^{k+1} - V_{Ri}^k) + \left(\frac{\partial I_{Li}^{ZIP}}{\partial V_{Li}} \right)_{V_{Ri}^k, V_{Li}^k} (V_{Li}^{k+1} - V_{Li}^k) \quad (36)$$

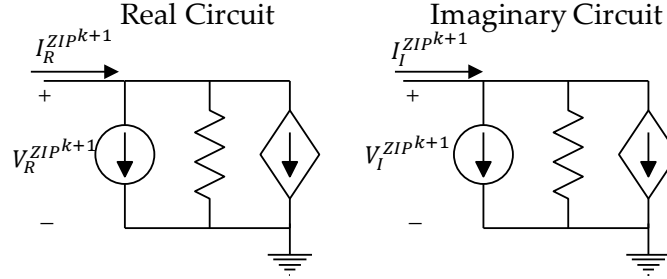


Figure 10: Real and imaginary equivalent circuit for the ZIP load model.

The linearized set of equations can then be mapped into the equivalent circuit for ZIP load model that is shown in Figure 10.

4.1.2.7 BIG Model

The BIG aggregated load model introduced in [52]-[54] (Figure 11) more accurately captures the true behavior of the aggregated load when compared against the traditional non-linear PQ load model, and is comparable in accuracy to the more comprehensive non-linear ZIP load model. Importantly, the BIG load model can be easily fitted with real-time measurement data and is linear while it also captures the true sensitivities of the aggregated load. As the network constraints (for lines and transformers etc.) are linear in equivalent circuit approach with non-linearities limited to PV and PQ models, the linear BIG load model results in linear equality constraints for the load nodes.

The BIG model is defined by a combination of constant current source ($\alpha_R^{BIG} + j\alpha_I^{BIG}$), a conductance (G^{BIG}) and a susceptance (B^{BIG}) whose real and imaginary currents are given by:

$$I_R^{BIG} + jI_I^{BIG} = \alpha_R^{BIG} + j\alpha_I^{BIG} + (V_R^{BIG} + jV_I^{BIG})(G^{BIG} + B^{BIG}) \quad (37)$$

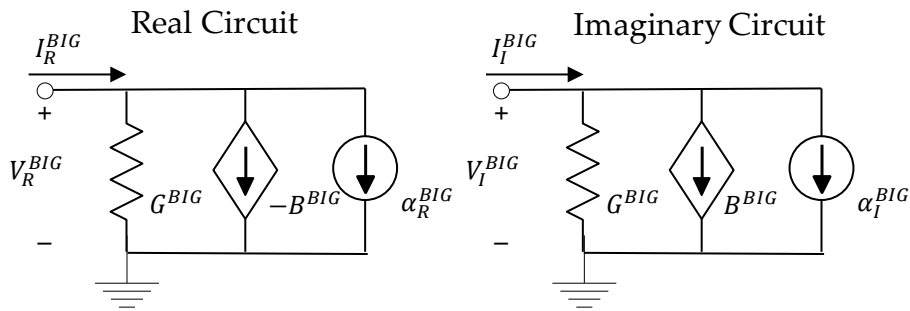


Figure 11: Equivalent circuit of a BIG load model.

The appendix A of this report delves deeper into the proposed BIG load model and discusses its advantages over other existing aggregated load models in detail. It also refers to machine learning methods that have been developed within our group to fit the BIG load model to capture the true sensitivities of the load currents.

4.1.2.8 Transformer

Transformers are an integral part of the electric grid and are used to step-up or step-down the grid voltages. In addition, some transformers contain a built-in phase shifter and have a capability to introduce phase shifts between the buses to which they are connected. We derive the equivalent circuit of the transformer with the transformer impedance modeled on the secondary of the transformer as shown in Figure 12.

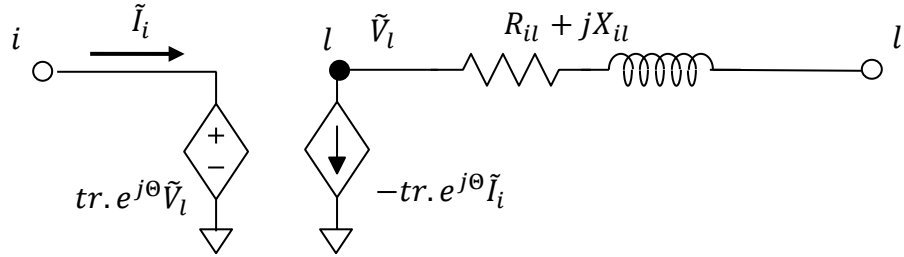


Figure 12: Equivalent circuit for a transformer.

To derive the split circuit equivalent model of the transformer, we begin by relating the primary and secondary voltages ($\tilde{V}_i$ and $\tilde{V}_l$) by the turns ratio tr and the phase angle θ :

$$\frac{\tilde{V}_i}{\tilde{V}_l} = tr \cdot e^{j\theta} \rightarrow \frac{V_R^i + jV_I^i}{V_R^l + jV_I^l} = tr(\cos \theta + j \sin \theta) \quad (38)$$

Representing the primary transformer voltages as functions of secondary transformer voltages by splitting them into real and imaginary parts results in:

$$V_R^i = tr(V_R^l \cos \theta - V_I^l \sin \theta) \quad (39)$$

$$V_I^i = tr(V_I^l \cos \theta + V_R^l \sin \theta) \quad (40)$$

The first term of (39) represents a voltage-controlled voltage source, where the controlling voltage is the secondary side voltage in the real circuit. The second term is a voltage-controlled voltage source, but here the controlling voltage is the secondary side voltage in the imaginary circuit. The equation (40) represents similar terms. These terms can be used to represent the primary side of transformer equivalent circuit as shown in Figure 13.

Similarly, the real and imaginary equivalent circuits for the secondary of the transformer can be developed by the primary and secondary current relationship. The relationship between primary and secondary complex currents ($\tilde{I}_i$ and $\tilde{I}_l$) in terms of the turns ratio is given by:

$$\frac{\tilde{I}_l}{\tilde{I}_i} = -tr \cdot e^{-j\theta} \rightarrow \frac{I_R^l + jI_I^l}{I_R^i + jI_I^i} = -tr(\cos \theta - j \sin \theta) \quad (41)$$

We derive the currents for the secondary side of the transformer as a function of primary side currents and split them into respective real and imaginary terms:

$$I_R^l = -tr(I_R^i \cos \theta + I_I^i \sin \theta) \quad (42)$$

$$I_I^l = -tr(I_I^i \cos \theta - I_R^i \sin \theta) \quad (43)$$

The first term of (42) represents a current-controlled current source, where the controlling current is the current which flows through the primary side in the real circuit. The second term represents a current-controlled current source, but here the controlling current is the current which flows through the primary side in the imaginary circuit. The equation (43) represents similar terms. These terms can be used to represent the secondary side of transformer equivalent circuit as shown in Figure 13 for the phase shifter value of 0.

The leakage term of the transformer $Z_{il} = R_{il} + jX_{il}$ is modeled on the secondary side. We model it using same approach as that of the transmission line given in Section 4.1.2.9 resulting in the following real and imaginary terms:

$$I_R^l = V_R^{l'l} G_{il} - V_I^{l'l} B_{il} \quad (44)$$

$$I_I^l = V_I^{l'l} G_{il} + V_R^{l'l} B_{il} \quad (45)$$

The first term of (44) is a conductance and the second term is a voltage-controlled current source; likewise for equation (45). A full equivalent circuit model for the transformer for phase shifter magnitude of 0° is shown in Figure 13.

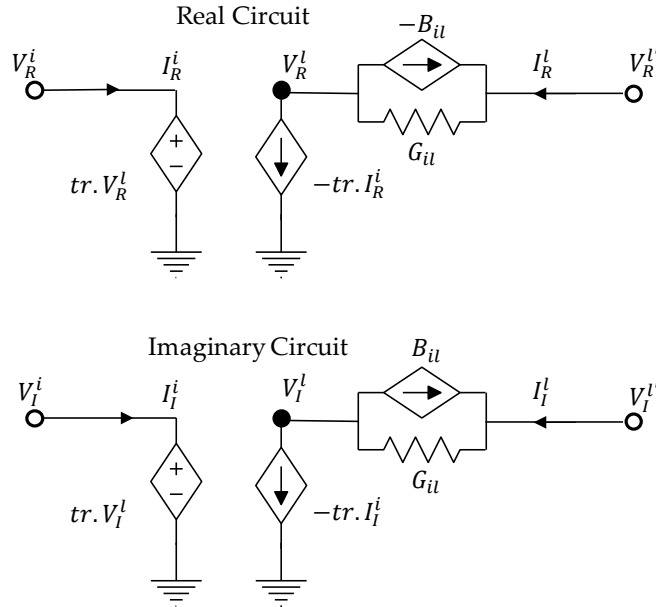


Figure 13: Real and imaginary circuit for a transformer model.

Control of transformer taps

Transformers with controllable taps can control the voltage at either its own node or another node in the system given by $\mathcal{W}$. In the existing methodology, transformers taps are generally adjusted in the outer loop of the solver based on the system voltages obtained in the inner loop NR solution. However, this technique suffers from oscillations and convergence to non-physical solutions as described in the case of generators in Section 0. Therefore, to overcome these challenges, we propose the use of a continuous transformer model for the control of transformer taps in the system.

In this continuous twice differential transformer model, a sigmoid curve is used to describe the relationship between the transformer turns ratio and the voltage at the controlled node. Within its limits, the transformer taps adjust its value to control the controlled node's voltage. However, if the transformer taps ratio exceeds its limit, the turns ratio value saturates and no longer controls the voltage of the controlled node.

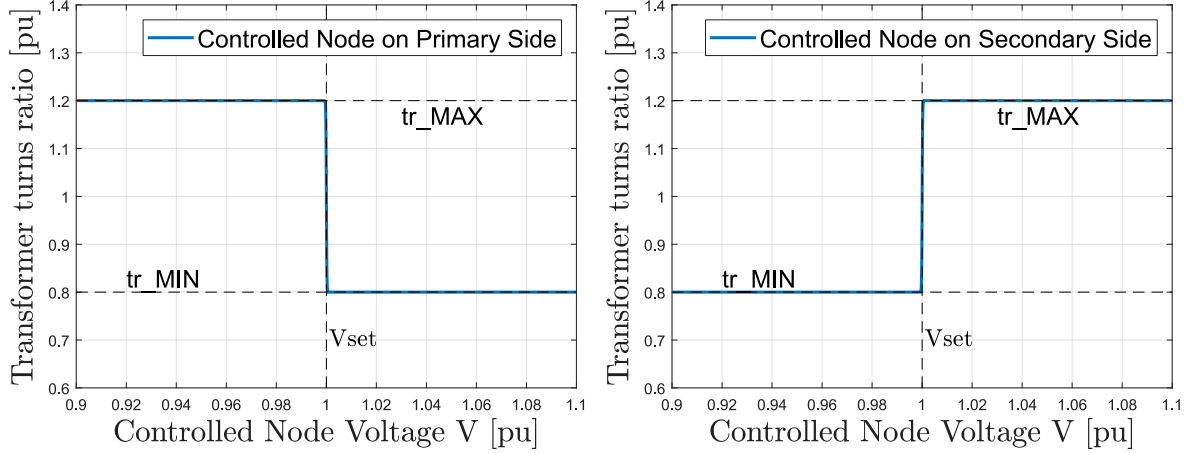


Figure 14: Continuous transformer tap control schematic.

Unlike the generator continuous model in Section 4.1.2.3, in the transformer continuous model a set of sigmoid curves is used to control the voltage of the controlled node $\mathcal{W}$ as a function of turns ratio. The choice of the sigmoid curve depends on the relative location of the controlled bus as shown in Figure 14. For instance, consider a bus connected to the primary side of the transformer whose voltage is being controlled. If the observed voltages on the controlled bus are lower than the set voltage, then the primary taps are increased to adjust the voltage toward the set voltage whereas if the observed voltages are higher than the set value then the primary taps are reduced to adjust the voltage. On the other hand, if the controlled bus $\mathcal{W}$ is on the secondary side of the transformer, then the primary taps are reduced to increase the voltage of the controlled bus $\mathcal{W}$ and increased to the reduce the voltage of the controlled bus on the secondary side. The voltage constraint characteristics for the controlled node $\mathcal{W}$ whose relative location is on the primary side of the transformer is:

$$tr = \frac{tr_{MAX} - tr_{MIN}}{1 + \exp(\lambda_S * [\sqrt{V_{RW}^2 + V_{IW}^2} - V_{set}])} + tr_{MIN} \quad (46)$$

In case the controlled node $\mathcal{W}$'s relative location is on the secondary side of the transformer, then the voltage constraint characteristics are given by:

$$tr = \frac{tr_{MIN} - tr_{MAX}}{1 + \exp(\lambda_S * [\sqrt{V_{RW}^2 + V_{IW}^2} - V_{set}])} + tr_{MAX} \quad (47)$$

In the continuous model for the transformer voltage control, the turns ratio parameter (tr) is an unknown variable (with continuous range) and requires additional stamps in

the system Jacobian for incorporating the sensitivities of transformer currents to transformer turns ratio. Importantly, the voltage and current equations are no longer linear functions of unknown variables, and therefore, are linearized prior to being stamped in the Jacobian matrix.

Importantly, the actual tap adjustment in the transformer is discrete, therefore, once the inner loop of NR is completed with the continuous model, the taps are snapped to their closest discrete value to obtain the final solution. In practice, it is rare that the snapping back action could result in system to diverge. However, theoretically it is possible due to two reasons:

- i. The modified system state due to the change in transformer tap magnitude from its continuous to discrete value could result in an infeasible network.
- ii. The set of non-linear equations representing the modified system state may diverge with prior solution as the initial condition.

In our solver, we make use of continuation (like methodology in Section 0) and optimization-based methods [71] to address these concerns. In case the system is infeasible due to the change in transformer tap magnitude from its continuous to discrete value, the optimization-based methods [71] can identify the system infeasibility and accordingly adjust the discrete elements values such that the system is feasible. In case, the divergence is due to the lack of good initial conditions for the snapped system state, continuation methods can be used to gradually modify the discrete elements parameters from their continuous value to the closest discrete value until convergence is achieved.

4.1.2.9 Transmission Line

Positive-sequence power flow analysis mainly uses a simplified pi-model for the transmission line. The model is shown in Figure 15. In this model, both the series and the shunt impedances are approximated by an equivalent lumped pi-model, which is a linear model with branch currents as linear functions of from and to node voltages. We trivially derive the split circuits for this model using Kirchhoff's current law. In this derivation, for the sake of simplicity, the real and imaginary series and the shunt current terms for the transmission line are derived separately and then later combined to represent the whole model.

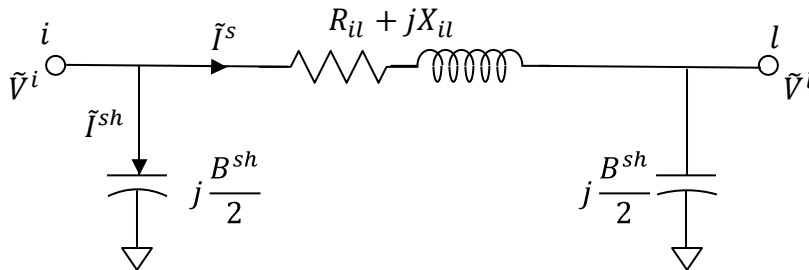


Figure 15: Equivalent circuit of a pi-model of the transmission line.

The series complex current for the transmission line between nodes i and l can be calculated from Ohm's law:

$$I_R^s + jI_I^s = \frac{V_R^{il} + jV_I^{il}}{R_{il} + jX_{il}} \quad (48)$$

The real and imaginary terms of (48) can be split into their respective equations:

$$I_R^s = V_R^{il} \frac{R_{il}}{R_{il}^2 + jX_{il}^2} + V_I^{il} \frac{X_{il}}{R_{il}^2 + jX_{il}^2} \quad (49)$$

$$I_I^s = V_I^{il} \frac{R_{il}}{R_{il}^2 + jX_{il}^2} - V_R^{il} \frac{X_{il}}{R_{il}^2 + jX_{il}^2} \quad (50)$$

Finally, the conductance (G_{il}) and susceptance (B_{il}) values can be used to simplify the derived terms, which can then be mapped into the equivalent circuit.

$$I_R^s = V_R^{il} G_{il} - V_I^{il} B_{il} \quad (51)$$

$$I_I^s = V_I^{il} G_{il} + V_R^{il} B_{il} \quad (52)$$

Like series terms, the shunt terms for the line model can be calculated via Ohm's law, which can then be split into their real and imaginary terms as given in (54)-(55). The series and shunt terms can then be combined and mapped into their equivalent circuits as shown in Figure 16.

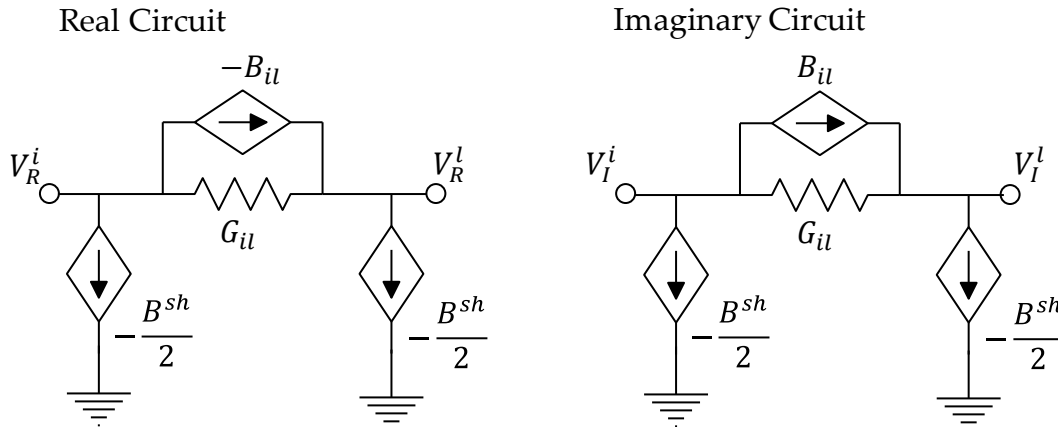


Figure 16: Real and imaginary circuit for the pi-model of transmission line.

$$I_R^{sh} + jI_I^{sh} = (V_R^i + jV_I^i)j \frac{B^{sh}}{2} \quad (53)$$

$$I_R^{sh} = -V_I^i \frac{B^{sh}}{2} \quad (54)$$

$$I_I^{sh} = V_R^i \frac{B^{sh}}{2} \quad (55)$$

4.1.2.10 Preliminary Results for Positive Sequence Power Flow

The purpose of this experiment is to validate the equivalent circuit approach for positive sequence power flow. To do so we simulate multiple cases from the flat start and document the results in Table 2. We report the case as converged if the solution obtained

from our framework can be plugged into a commercial solver to result in the same solution. Additionally, we also document the number of iterations it took for the given case to converge.

Table 2: Preliminary results for positive sequence power flow with equivalent circuit approach

Case Name	Number of Nodes	Reference	Iteration Count	Solution
case14	14	IEEE	4	Converged
case118	118	IEEE	5	Converged
case145	145	IEEE	14	Converged
SouthCarolina500	500	ACTIVSg500	4	Converged
Texas2000_June2016	2000	ICSEG	5	Converged
case1354pegase	1354	PEGASE	5	Converged
Case13659pegase	13659	PEGASE	xx	Diverged
bench	1648	PSSE benchmark	7	Converged
bench2	7917	PSSE benchmark	xx	Diverged

Results in Table 2 demonstrate that the equivalent circuit approach can solve the positive sequence power problem for most of the test cases from flat start. However, as expected some of the cases diverge when simulated from the flat start. Therefore, in the rest of this report, we will develop methods that can ensure convergence for hard-to-solve ill conditioned and large test cases from arbitrary initial conditions. First, however, we extend the equivalent circuit formulation framework described in this section for positive-sequence power grid models to three-phase power grid models.

4.1.3 Equivalent Circuit Models for Three-Phase Power Flow Problem

In this section, we develop equivalent circuit models for some of the most commonly used elements in the distribution grid for three-phase power flow analysis.

4.1.3.1 Slack Bus

In the distribution system analysis, the transmission edge of the grid is usually modeled as an infinite bus, which is represented via a substation or infinite bus that generally feeds into but rarely absorbs power from the distribution system. Each phase of the infinite or the slack bus can be represented in the real circuit as an independent voltage source of value $|V_i^\Omega| \cos(\theta_i^\Omega)$, and in the imaginary circuit (imaginary portion of the split circuit) as an independent voltage source of value $|V_i^\Omega| \sin(\theta_i^\Omega)$. It should be noted that if the slack bus is connected in a wye configuration, its magnitude represents the line-to-neutral voltage, whereas if connected in delta configuration, it will represent the line-to-line voltage. The complete split circuit model for a 3-phase slack bus connected in grounded wye configuration is shown in Figure 17. Importantly, it should be noted that in future with presence of multiple large generation resources within the distributed grid, it is likely that the distribution grid will contribute control during primary and secondary control as in the case of transmission grid. This can be easily incorporated into our framework using distributed slack framework following the methodology stipulated in Section 4.1.2.4.

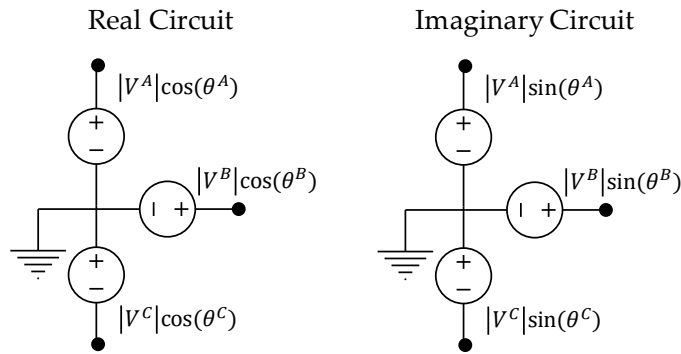


Figure 17: Real and imaginary circuits for slack bus in three-phase power flow problem.

4.1.3.2 ZIP Load Model

Amongst all the existing aggregated load models in distribution system analysis, the ZIP load model is the most comprehensive. It models the aggregated load in the system as a mix of constant impedance, constant current, and constant power load models, which can be mathematically represented for each phase as follows:

$$(P_i^{ZIP})^\Omega = Z_p^\Omega (|V_i^\Omega|)^2 + I_p^\Omega (|V_i^\Omega|) + S_p^\Omega \quad (56)$$

$$(Q_i^{ZIP})^\Omega = Z_Q^\Omega (|V_i^\Omega|)^2 + I_Q^\Omega (|V_i^\Omega|) + S_Q^\Omega \quad (57)$$

In the equivalent circuit approach, the equations for the ZIP load model can be re-written as:

$$(I_{Ri}^{ZIP})^\Omega = Z_P^\Omega V_{Ri}^\Omega - Z_Q^\Omega V_{Ii}^\Omega + \frac{S_P^\Omega V_{Ri}^\Omega + S_Q^\Omega V_{Ii}^\Omega}{(V_{Ri}^\Omega)^2 + (V_{Ii}^\Omega)^2} + \left(\sqrt{I_P^{\Omega^2} + I_Q^{\Omega^2}} \right) \cdot \cos(I_{pf}^\Omega + \delta_i^\Omega) \quad (58)$$

$$(I_{Ii}^{ZIP})^\Omega = Z_P^\Omega V_{Ii}^\Omega + Z_Q^\Omega V_{Ri}^\Omega + \frac{S_P^\Omega V_{Ii}^\Omega - S_Q^\Omega V_{Ri}^\Omega}{(V_{Ri}^\Omega)^2 + (V_{Ii}^\Omega)^2} + \left(\sqrt{I_P^{\Omega^2} + I_Q^{\Omega^2}} \right) \cdot \sin(I_{pf}^\Omega + \delta_i^\Omega) \quad (59)$$

where:

$$I_{pf}^\Omega = \tan^{-1} \left(\frac{I_Q^\Omega}{I_P^\Omega} \right) \quad (60)$$

$$\delta_i^\Omega = \tan^{-1} \left(\frac{V_{Ii}^\Omega}{V_{Ri}^\Omega} \right) \quad (61)$$

For the load model given in (58) through (61), the constant impedance part of the load is linear, whereas the constant current and constant power part of the aggregated load is nonlinear. Linearizing the set of equations using Taylor expansion results in the following expression for each phase Ω in Ω_{set} :

$$\begin{aligned} (I_{Ri}^{ZIP})^{\Omega^{k+1}} &= (I_{Ri}^{ZIP})^{\Omega^k} + \left(\frac{\partial I_{Ri}^{ZIP}}{\partial V_{Ri}} \bigg|_{V_{Ri}^k, V_{Ii}^k} \right)^\Omega (V_{Ri}^{\Omega^{k+1}} - V_{Ri}^{\Omega^k}) \\ &\quad + \left(\frac{\partial I_{Ri}^{ZIP}}{\partial V_{Ii}} \bigg|_{V_{Ri}^k, V_{Ii}^k} \right)^\Omega (V_{Ii}^{\Omega^{k+1}} - V_{Ii}^{\Omega^k}) \end{aligned} \quad (62)$$

$$\begin{aligned} (I_{Ii}^{ZIP})^{\Omega^{k+1}} &= (I_{Ii}^{ZIP})^{\Omega^k} + \left(\frac{\partial I_{Ii}^{ZIP}}{\partial V_{Ri}} \bigg|_{V_{Ri}^k, V_{Ii}^k} \right)^\Omega (V_{Ri}^{\Omega^{k+1}} - V_{Ri}^{\Omega^k}) \\ &\quad + \left(\frac{\partial I_{Ii}^{ZIP}}{\partial V_{Ii}} \bigg|_{V_{Ri}^k, V_{Ii}^k} \right)^\Omega (V_{Ii}^{\Omega^{k+1}} - V_{Ii}^{\Omega^k}) \end{aligned} \quad (63)$$

The linearized set of equations can then be mapped into the equivalent three-phase model of the ZIP load either in wye (Y) or delta (D) formation, as shown in Figure 18.

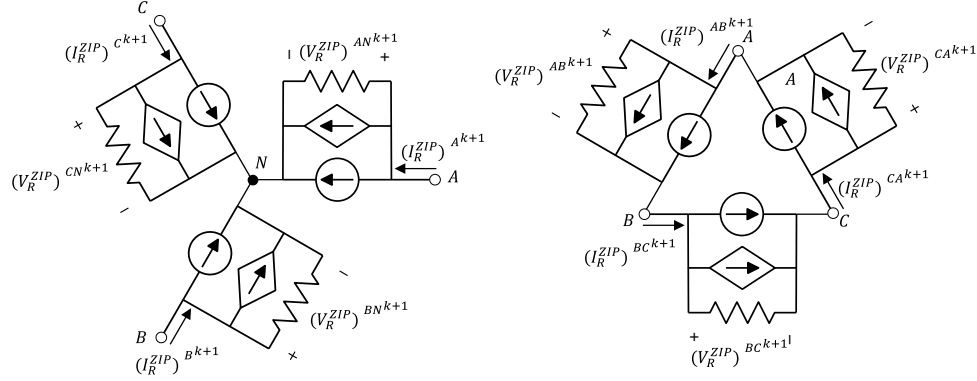


Figure 18: Real circuit for a) wye connected ZIP load model (on left) b) delta (D) connected ZIP load model (on right).

It is important to note that the ZIP model results in non-linear network constraints for both the PQV and CIM formulations, which further adds to already existing non-linearities in the formulation. We propose to replace the non-linear ZIP model with a linear three-phase BIG model that provides comparable accuracy.

4.1.3.3 Three-phase BIG load model

The proposed linear positive sequence BIG load model in Section 4.1.2.7 is further extended to a linear three-phase aggregated load model that can be connected in either wye or delta connection as in the case of the ZIP load model. Refer to Appendix A for more detailed explanation of the BIG load model.

4.1.3.4 Transmission Line

The three main types of transmission lines in the distribution grid are the overhead line, underground cable, and the triplex cable. The overhead line generally consists of a 4-wire configuration with three phase conductors and one neutral conductor. The concentric underground cable, on the other hand, consists of a 7-wire configuration with three phase conductors, along with corresponding neutral conductors and an additional neutral conductor. The triplex cable consists of three wires with two hot conductors and one neutral conductor.

The impedance matrix for the overhead, underground and triplex lines are of the order 4x4, 7x7, and 3x3, respectively. However, with the use of Kron's reduction [32], we can eliminate neutral wires from the models resulting in 3x3, 3x3 and 2x2 impedance matrices for overhead lines, underground cables and triplex cables, respectively. Finally, admittance matrix ($\tilde{Y}_{line}$) for the line model can then be calculated by finding the inverse of the impedance matrix (Z_{line}):

$$\tilde{Y}_{line} = Z_{line}^{-1} \quad (64)$$

With the admittance matrix calculated, the transmission line branch currents can be represented by Ohm's Law, where $\tilde{V}_{Aa}$, $\tilde{V}_{Bb}$ and $\tilde{V}_{Cc}$ are the voltage drops across the lines:

$$\begin{bmatrix} \tilde{I}_A \\ \tilde{I}_B \\ \tilde{I}_C \end{bmatrix} = \begin{bmatrix} \tilde{Y}_{aa} & \tilde{Y}_{ab} & \tilde{Y}_{ac} \\ \tilde{Y}_{ba} & \tilde{Y}_{bb} & \tilde{Y}_{bc} \\ \tilde{Y}_{ca} & \tilde{Y}_{cb} & \tilde{Y}_{cc} \end{bmatrix} \begin{bmatrix} \tilde{V}_{Aa} \\ \tilde{V}_{Bb} \\ \tilde{V}_{Cc} \end{bmatrix} \quad (65)$$

Since the series admittances $G_{il}^{\mathbf{x}} + jB_{il}^{\mathbf{x}}$ of the branches have both real and imaginary components, the system of equations from (65) can be split as:

$$\begin{bmatrix} I_R^A \\ I_I^A \\ I_R^B \\ I_I^B \\ I_R^C \\ I_I^C \end{bmatrix} = \begin{bmatrix} G_{aa}^{\mathbf{x}} & -B_{aa}^{\mathbf{x}} & G_{ab}^{\mathbf{x}} & -B_{ab}^{\mathbf{x}} & G_{ac}^{\mathbf{x}} & -B_{ac}^{\mathbf{x}} \\ B_{aa}^{\mathbf{x}} & G_{aa}^{\mathbf{x}} & B_{ab}^{\mathbf{x}} & G_{ab}^{\mathbf{x}} & B_{ac}^{\mathbf{x}} & G_{ac}^{\mathbf{x}} \\ G_{ba}^{\mathbf{x}} & -B_{ba}^{\mathbf{x}} & G_{bb}^{\mathbf{x}} & -B_{bb}^{\mathbf{x}} & G_{bc}^{\mathbf{x}} & -B_{bc}^{\mathbf{x}} \\ B_{ba}^{\mathbf{x}} & G_{ba}^{\mathbf{x}} & B_{bb}^{\mathbf{x}} & G_{bb}^{\mathbf{x}} & B_{bc}^{\mathbf{x}} & G_{bc}^{\mathbf{x}} \\ G_{ca}^{\mathbf{x}} & -B_{ca}^{\mathbf{x}} & G_{cb}^{\mathbf{x}} & -B_{cb}^{\mathbf{x}} & G_{cc}^{\mathbf{x}} & -B_{cc}^{\mathbf{x}} \\ B_{ca}^{\mathbf{x}} & G_{ca}^{\mathbf{x}} & B_{cb}^{\mathbf{x}} & G_{cb}^{\mathbf{x}} & B_{cc}^{\mathbf{x}} & G_{cc}^{\mathbf{x}} \end{bmatrix} \begin{bmatrix} V_R^{Aa} \\ V_I^{Aa} \\ V_R^{Bb} \\ V_I^{Bb} \\ V_R^{Cc} \\ V_I^{Cc} \end{bmatrix} \quad (66)$$

Using the same approach, the transmission line shunt currents can be derived, where $\tilde{V}_A$, $\tilde{V}_B$ and $\tilde{V}_C$ are the line-to-ground nodal voltages. Since the admittance of the shunt elements in the pi-model is purely imaginary ($\tilde{Y}_i^{sh} = jB_i^{sh}$), we derive the following set of equations from Ohm's law:

$$\begin{bmatrix} I_R^{Ash} \\ I_I^{Ash} \\ I_R^{Bsh} \\ I_I^{Bsh} \\ I_R^{Csh} \\ I_I^{Csh} \end{bmatrix} = \begin{bmatrix} 0 & -B_{aa}^{sh} & 0 & -B_{ab}^{sh} & 0 & -B_{ac}^{sh} \\ B_{aa}^{sh} & 0 & B_{ab}^{sh} & 0 & B_{ac}^{sh} & 0 \\ 0 & -B_{ba}^{sh} & 0 & -B_{bb}^{sh} & 0 & -B_{bc}^{sh} \\ B_{ba}^{sh} & 0 & B_{bb}^{sh} & 0 & B_{bc}^{sh} & 0 \\ 0 & -B_{ca}^{sh} & 0 & -B_{cb}^{sh} & 0 & -B_{cc}^{sh} \\ B_{ca}^{sh} & 0 & B_{cb}^{sh} & 0 & B_{cc}^{sh} & 0 \end{bmatrix} \begin{bmatrix} V_R^A \\ V_I^A \\ V_R^B \\ V_I^B \\ V_R^C \\ V_I^C \end{bmatrix} \quad (67)$$

Equations (66) and (67) for the transmission line are then mapped into an equivalent circuit given by linear resistors and voltage-controlled current sources. Figure 19 shows the real sub-circuit for one of the phases of a transmission line.

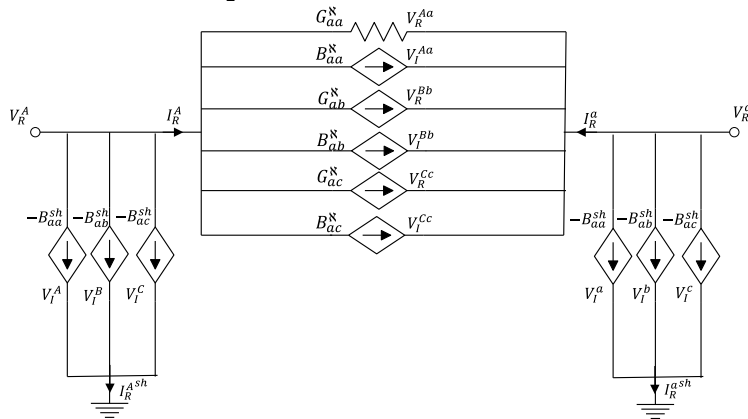


Figure 19: Real circuit of a transmission line (Phase A).

4.1.3.5 Three-Phase Transformers

Three-phase distribution transformers are used to transform the voltages from transmission level to sub-transmission level and to isolate single-phase circuits from three-phase circuits. Various configurations for the three-phase transformers exist in the distribution system, some of which include:

- i. Grounded wye – grounded wye (grY – grY)
- ii. Delta – delta (D – D)
- iii. Wye – delta (wye – D)
- iv. Grounded wye – delta (grY – D)
- v. Delta – wye (D – wye)
- vi. Open wye – open delta

In the next subsection we will derive the equivalent circuit model for the grounded wye – grounded wye three-phase transformer configuration. Following the same methodology, equivalent circuits for the other transformer configurations can also be derived.

Grounded wye – grounded wye (grY – grY) Configuration

In the grounded wye – grounded wye configuration of the three-phase transformer, the relationship between the primary and secondary currents ($\tilde{I}_{pri}^\Omega$, $\tilde{I}_{sec}^\Omega$) and voltages ($\tilde{V}_{pri}^\Omega$, $\tilde{V}_{sec}^\Omega$) for each individual phase is given by:

$$\tilde{V}_{pri}^\Omega = tr. \tilde{V}_{sec}^\Omega e^{j\theta^\Omega} \quad (68)$$

$$\tilde{I}_{sec}^\Omega = -tr. \tilde{I}_{pri}^\Omega e^{-j\theta^\Omega} \quad (69)$$

Splitting of these current and voltage equations into real and imaginary terms results in the following set of equations:

$$V_{R\ pri}^\Omega = tr (V_{R\ sec}^\Omega \cos\theta^\Omega - V_{I\ sec}^\Omega \sin\theta^\Omega) \quad (70)$$

$$V_{I\ pri}^\Omega = tr (V_{R\ sec}^\Omega \sin\theta^\Omega + V_{I\ sec}^\Omega \cos\theta^\Omega) \quad (71)$$

$$I_{R\ sec}^\Omega = -tr (I_{R\ pri}^\Omega \cos\theta^\Omega + I_{I\ sec}^\Omega \sin\theta^\Omega) \quad (72)$$

$$I_{I\ sec}^\Omega = -tr (-I_{R\ pri}^\Omega \sin\theta^\Omega + I_{I\ sec}^\Omega \cos\theta^\Omega) \quad (73)$$

These equations given by (70) through (73) are mapped into equivalent circuit model of the transformer by using controlled voltage and current sources. In addition, the transformer losses for each of the three-phases are modeled on the secondary of the transformer, represented by following loss admittance relationship:

$$\tilde{I}_R^\Omega = G_{loss} \tilde{V}_R^{\Omega\Omega'} - B_{loss} \tilde{V}_I^{\Omega\Omega'} \quad (74)$$

$$\tilde{I}_I^\Omega = G_{loss} \tilde{V}_I^{\Omega\Omega'} + B_{loss} \tilde{V}_R^{\Omega\Omega'} \quad (75)$$

Figure 20 shows the real circuit of the grounded wye – grounded wye transformer model with zero phase shift.

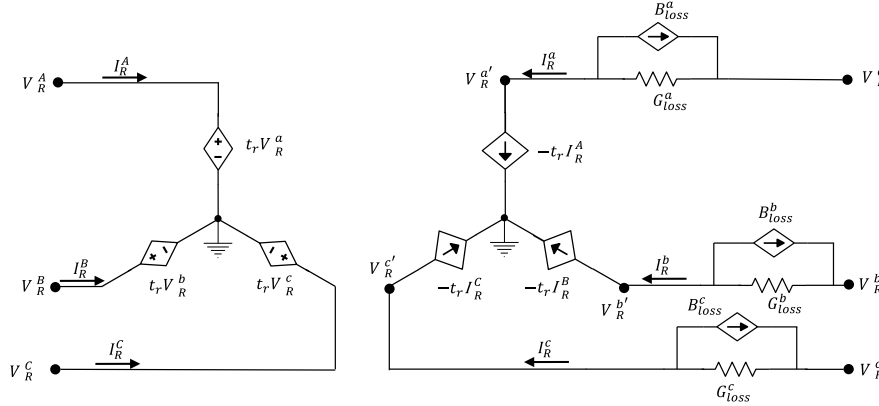


Figure 20: Real circuit for the grounded wye – grounded wye transformer with no phase shift.

Importantly, akin to positive-sequence transformer, the three-phase transformer can also control the phase voltages either at its primary or secondary side via the control of its turns ratio. In the solver, this control can either be modeled in the outer loop via the use of discontinuous piecewise model or be modeled implicitly through the use of continuous transformer tap model, which is extended from the positive-sequence model of the transformer shown in Section 0.

4.1.4 Preliminary Results for Three-phase power flow

In this section, we document preliminary three-phase power flow results for a standard 4-bus test case [29] in equivalent circuit approach.

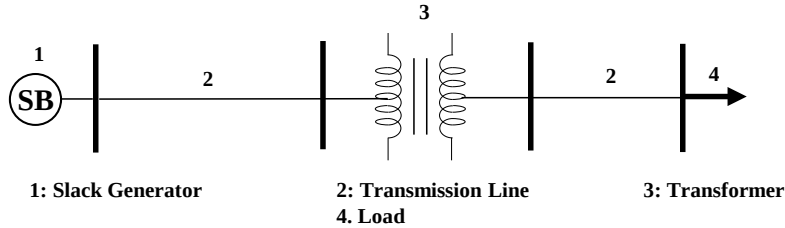


Figure 21: Standard 4-bus test case system.

The schematic of the standard 4-bus test case is shown in Figure 21. The results for this test case are shown for different transformer configurations in Table 3. The tabulated results represent the complex phase voltages at the load bus shown in Figure 21 and those compare well against the results obtained from the standard reference [29] for the same set of transformer configurations.

Table 3: SUGAR three-phase results for 4-Bus test case

Configuration	Balanced		Unbalanced	
	SUGAR 3-Phase [V $\angle^\circ$]	Results in [29] [V $\angle^\circ$]	SUGAR 3-Phase [V $\angle^\circ$]	Results in [29] [V $\angle^\circ$]
Step-down grY-grY	V _A : 1918 $\angle$ -9.1 V _B : 2061 $\angle$ -128.3 V _C : 1981 $\angle$ 110.9	V _A : 1918 $\angle$ -9.1 V _B : 2061 $\angle$ -128.3 V _C : 1981 $\angle$ 110.9	V _A : 2175 $\angle$ -4.1 V _B : 1930 $\angle$ -126.8 V _C : 1833 $\angle$ 102.8	V _A : 2175 $\angle$ -4.1 V _B : 1930 $\angle$ -126.8 V _C : 1833 $\angle$ 102.8
Step-down D-D	V _{AB} : 3442 $\angle$ 22.3 V _{BC} : 3497 $\angle$ -99.4 V _{CA} : 3384 $\angle$ 140.7	V _{AB} : 3442 $\angle$ 22.3 V _{BC} : 3497 $\angle$ -99.4 V _{CA} : 3384 $\angle$ 140.7	V _{AB} : 3431 $\angle$ 24.3 V _{BC} : 3647 $\angle$ -100.4 V _{CA} : 3294 $\angle$ 138.6	V _{AB} : 3431 $\angle$ 24.3 V _{BC} : 3647 $\angle$ -100.4 V _{CA} : 3294 $\angle$ 138.6
Step-down Y-D	V _{AB} : 3437 $\angle$ -7.8 V _{BC} : 3497 $\angle$ -129.3 V _{CA} : 3388 $\angle$ 110.6	V _A : 3437 $\angle$ -7.8 V _B : 3497 $\angle$ -129.3 V _C : 3388 $\angle$ 110.6	V _{AB} : 3425 $\angle$ -5.8 V _{BC} : 3646 $\angle$ -130.3 V _{CA} : 3298 $\angle$ 108.6	V _A : 3425 $\angle$ -5.8 V _B : 3646 $\angle$ -130.3 V _C : 3298 $\angle$ 108.6
Step-up grY-grY	V _A : 13630 $\angle$ -3.5 V _B : 13681 $\angle$ -123.5 V _C : 13665 $\angle$ 116.5	V _{AB} : 13631 $\angle$ -3.5 V _{BC} : 13682 $\angle$ -123.5 V _{CA} : 13661 $\angle$ 116.5	V _A : 13814 $\angle$ -2.2 V _B : 13613 $\angle$ -123.4 V _C : 13618 $\angle$ 114.9	V _{AB} : 13815 $\angle$ -2.2 V _{BC} : 13614 $\angle$ -123.4 V _{CA} : 13615 $\angle$ 114.9
Step-up D-D	V _{AB} : 23658 $\angle$ 26.6 V _{BC} : 23688 $\angle$ -93.5 V _{CA} : 23625 $\angle$ 146.5	V _{AB} : 23657 $\angle$ 26.6 V _{BC} : 23688 $\angle$ -93.5 V _{CA} : 23625 $\angle$ 146.5	V _{AB} : 23611 $\angle$ 27.2 V _{BC} : 24015 $\angle$ -93.7 V _{CA} : 23492 $\angle$ 145.9	V _{AB} : 23610 $\angle$ 27.2 V _{BC} : 24015 $\angle$ -93.7 V _{CA} : 23492 $\angle$ 145.9
Step-up Y-D	V _{AB} : 23682 $\angle$ 56.6 V _{BC} : 23664 $\angle$ -63.6 V _{CA} : 23626 $\angle$ 176.5	V _{AB} : 23681 $\angle$ 56.6 V _{BC} : 23664 $\angle$ -63.6 V _{CA} : 23625 $\angle$ 176.5	V _{AB} : 23638 $\angle$ 57.1 V _{BC} : 23995 $\angle$ -63.8 V _{CA} : 23496 $\angle$ 175.9	V _{AB} : 23637 $\angle$ 57.1 V _{BC} : 23995 $\angle$ -63.8 V _{CA} : 23495 $\angle$ 175.9

These results validate the equivalent circuit models for the three-phase power flow elements as was achieved in the case of positive-sequence power flow models in Section 4.1.2.10. However, in general, solely representing the grid elements as equivalent circuit models cannot ensure convergence for three-phase power flow test cases from arbitrary initial conditions and therefore, the following Sections in this report will develop models and techniques that can ensure robust convergence for any three-phase test case from arbitrary initial conditions.

4.1.5 Physics Based Models

We have previously shown in [50] that any physics-based device model can be directly mapped into an equivalent circuit to be used in both the steady-state analysis (discussed here) as well as the transient analysis (see Appendix B). In general, physics-based models developed from fundamental principles are used for time-domain transient analysis [58]. However, in both the power flow and the three-phase power flow analyses, simplified, aggregated models are often used, resulting in less accurate and inconsistent results. Understandably, it is often necessary to use simplified aggregated models for system load and generation due to the lack of data pertaining to individual grid elements. However, this is not always the case, and more accurate grid operating state with true voltage

sensitivities can be obtained by using physics-based models in the power flow and three-phase power flow analyses.

Existing frameworks for power flow and three-phase power flow are generally unable to directly incorporate physics-based models based on the current and voltage state variables into the problem formulation. In contrast, our equivalent circuit formulation can directly incorporate any physics-based model based on the current and voltage variables into the analysis framework without loss of generality. To demonstrate this further, using induction motor as an example, we derive an equivalent circuit model of the same from fundamental principles that is further used in power flow as well as three-phase power flow analyses. In Appendix B, we show that the same derived model can also be used for time-domain transient analysis to result in consistent solution with the ones obtained from power flow and three-phase power flow analyses.

4.1.5.1 Physics Based Model for Induction Motor (IM)

Electric motors comprise roughly 45% of the total global electricity consumption [59], the majority of which can be attributed to IMs. Importantly, modeling these IMs in detail based on the true physics of the device can significantly improve the characterization of aggregated load in the grid. Often IMs are represented in the network model via PQ load or ZIP load models or are aggregated with other loads in the system that are further represented by the same. More advanced models for IMs that are based on the true physics of the device are sometimes used in three-phase power flow analysis [60]. However, these in-depth models tend to assume fixed speed operation (hence ignoring speed-flux non-linearities) thereby not capturing the true speed-dependent characteristics of the IM. In this section, we develop a physics-based model for IM that can be used in both power flow and three-phase power flow analyses. To derive this model for IM, we first introduce DQ transformation.

Direct-Quadrature (DQ) Transformation

The flux generated by the three-phase IM in rotating ABC frame has time varying coefficients in its voltage terms due to the sinusoidal nature of the mutual inductance. This makes the analysis of three phase IM cumbersome in the ABC (i.e. three-phase rotating) reference frame. However, this undesirable feature can be eliminated by use of the DQ transformation. DQ transformation can be performed in one of the three reference frames: i) synchronous reference frame; ii) stationary reference frame; and iii) rotating reference frame.

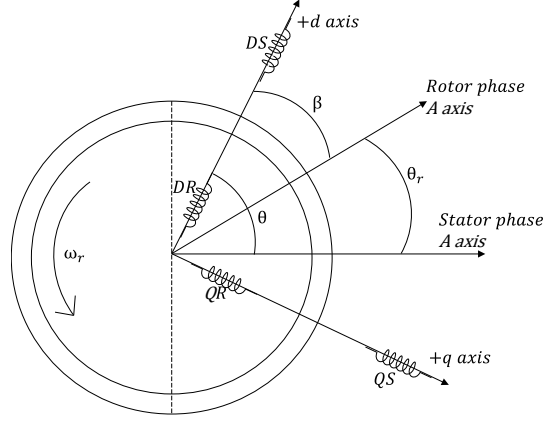


Figure 22: Superimposition of DQ-axis on 3-phase induction motor.

The final response of the IM is independent of the chosen reference frame. However, each of the reference frames has its own advantages and disadvantages depending on the problem that is being investigated [61]. For the purposes of this derivation, we make use of the synchronously rotating reference frame where DQ transformation matrix P_θ for the stator variables is as follows:

$$[P_\theta] = \frac{2}{3} \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ \cos(\theta) & \cos(\theta - \lambda) & \cos(\theta + \lambda) \\ \sin(\theta) & \sin(\theta - \lambda) & \sin(\theta + \lambda) \end{bmatrix} \quad (76)$$

with,

$$[F_{0dq}] = [P_\theta] \cdot [F_{abc}] \quad (77)$$

where function F can represent either currents or voltages.

For rotor variable transformation, θ is replaced with β in the equations above. For synchronous reference frame, the machine angle and speed variables are defined as follows:

$$\omega = p\theta = \omega_s \quad (78)$$

$$\beta = \theta - \theta_r = \theta_s - \theta_r \quad (79)$$

where p is the differential operator. ω_s and ω_r are the synchronous and rotor speed of the motor, respectively, and θ_s and θ_r are the stator and rotor position, respectively.

Motor Equations in Transient Domain

As we have transformed the three-phase parameters of IM into the DQ-frame, we can further derive the model of IM. The set of electrical equations that define the true behavior of the IM in time-domain are as follows [58]:

$$v_{ds} = R_s I_{ds} + p\psi_{ds} - \psi_{qs} p\theta \quad (80)$$

$$v_{qs} = R_s I_{qs} + p\psi_{qs} + \psi_{ds} p\theta \quad (81)$$

$$v_{dr} = R_r I_{dr} + p\psi_{dr} - \psi_{qr} p\beta \quad (82)$$

$$v_{qr} = R_r I_{qr} + p\psi_{qr} + \psi_{dr} p\beta \quad (83)$$

The flux linkages of the IM in the equations above are represented by the symbol ψ and are calculated using the following formulas:

$$\psi_{ds} = (L_{ls} + L_m)I_{ds} + L_m I_{dr} \quad (84)$$

$$\psi_{dr} = (L_{ls} + L_m)I_{dr} + L_m I_{ds} \quad (85)$$

$$\psi_{qs} = (L_{ls} + L_m)I_{qs} + L_m I_{qr} \quad (86)$$

$$\psi_{qr} = (L_{ls} + L_m)I_{qr} + L_m I_{qs} \quad (87)$$

where L_{ls} and L_{lr} represent the leakage-inductance of stator circuit and rotor circuit, respectively. L_m is the mutual inductance between the rotor and stator circuits. R_s and R_r are the stator and rotor resistance, respectively. The non-linearity in the electrical part of the IM is due to the speed voltage terms.

In addition to the equations above, the mechanical part of the IM is defined by a single differential equation [58]:

$$p\omega_r = \frac{(T_e - T_L - D\omega_r)}{J} \quad (88)$$

where

$$T_e = \frac{3}{4} L_m \text{poles} (I_{dr} I_{qs} - I_{qr} I_{ds}) \quad (89)$$

and T_e is the electrical torque of the IM in N.m and J is the motor net inertia in kg.m². poles represents the number of poles in the induction motor. The load torque (T_L) is generally described with a polynomial function of rotor speed.

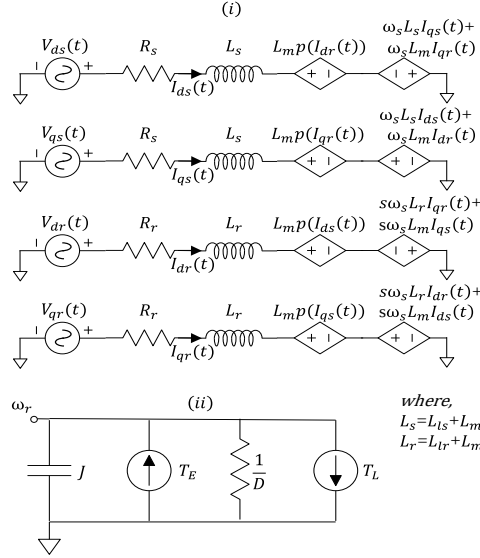


Figure 23: Equivalent circuit for 3-phase induction motor: (i) Electrical circuit; and (ii) Mechanical Circuit.

The equations derived above describe the time-domain behavior of a balanced three-phase squirrel cage IM and these can be directly mapped into an equivalent circuit following the steps detailed in [51], and is shown in Figure 23.

4.1.5.2 Steady-State Fundamental Frequency Model

To further use this model for positive sequence and three-phase power flow analysis, we zero out the time-derivative terms. Due to the use of the DQ-transformation, once the time-derivative terms are nulled, we obtain the steady-state model in source frequency. Furthermore, the model can be further extended to include zero sequence terms for unbalanced three-phase power flow analysis by adding an extra equation. If the motor were to have negative torque due to negative sequence terms, it would have to be separately calculated and added to (89).

To validate this IM model for power flow analysis, we make use of a 20 hp, 460 volts three-phase single squirrel cage induction motor. The motor data is given in Table 4.

Table 4: Three-phase squirrel cage induction motor parameters

V_{LL} (Volts)	f (Hz)	R_s (Ω)	R_r (Ω)	L_{ls} and L_{lr} (mH)
460	0.2761	0.2761	0.1645	2.191
L_m (mH)	poles	J (kg.m ²)	D (N.m.s)	T_L (N.m)
76.14	2	0.1	0.01771	10

For the validation test, the IM model is connected to a slack bus via a transmission line. The IM is then simulated at mechanical load of 10 N.m at rated source voltage. The results are documented in Table 5 and are converted to SI units from pu to compare against the

steady-state results obtained from MATLAB SimscapePowerSystems toolbox for the same test case. The results are an exact match thereby validating the model.

Table 5: IM results in equivalent circuit framework for steady-state (power flow) and time-domain transient analysis

Parameter	Unit	Equivalent Circuit Framework	SimScapePowerSystems
Rotor Speed	rad.s ⁻¹	375.01	375.01
Electric Torque	N.m	16.64	16.64
Stator direct-axis current	Amps	-11.36	-11.36
Stator quadrature-axis current	Amps	13.09	13.09
Rotor direct-axis current	Amps	11.56	11.56
Rotor quadrature-axis current	Amps	-0.49	-0.49

4.2 Circuit Simulation Methods for Power System Analyses

Decades of research in circuit simulation have demonstrated that circuit simulation methods can be applied for determining the DC state of highly non-linear circuits using NR. These techniques have been shown to make NR robust and practical for large-scale circuit problems [51], even those consisting of billions of nodes. Most notable is the ability to guarantee convergence to the correct physical solution (i.e. global convergence) and the capability of finding multiple operating points [56]. We propose analogous techniques for ensuring convergence to the correct physical solution for the power flow problem [34]-[35]. In this section, we provide a short overview of these techniques that can be applied to both positive sequence power flow and three-phase power flow problems without loss of generality. Note that throughout this section, the symbol superscript Ω in the mathematical expressions represents a phase from the set Ω_{set} of three phases a, b and c for the three-phase problem and represents the *positive sequence* (p) value for the power flow problem.

4.2.1 Limiting Methods

Here we discuss some of the methods that we apply to the NR algorithm during positive sequence and three-phase power flow problems to prevent divergence. We refer to these methods as limiting methods and they all relax the Newton's step during the iterations. Rather than relaxing the Newton's step by damping the whole vector of the update, we utilize the unique physics of the individual models and independently relax their Newton step during iterations.

4.2.1.1 Variable Limiting

The solution space of the system node voltages in a power flow problem is well defined. While solving the power flow problem, a large NR step may step out of this solution space and result in either divergence or convergence to a non-physical solution. It is, therefore, important to limit the NR step before an invalid step out of the solution space is made. In [34] we proposed variable limiting to achieve the postulated goal. In this technique, the state variables that are most sensitive to initial guesses are damped when the NR algorithm takes a large step out of the pre-defined solution space. Note, however, that *not all* of the system variables are damped for the variable limiting technique, as is done for traditional damped NR. Circuit simulation research has shown that damping most sensitive variables provides superior convergence compared to damped NR in general [51].

In the power flow and three-phase problem, the voltages on the PV node are highly sensitive to the reactive power (Q_G) value at that node. In the equivalent circuit formulation of the power flow and three-phase power flow problem each PV node augments the solution space by additional unknown variable Q_G for which initial guess must be assigned. However, unlike the node voltages, it is very hard to choose the appropriate initial guess for these Q_G variables, as they exhibit a large solution space. Therefore, with an arbitrary choice of these initial values, the power flow or three-phase power flow problem may diverge or converge to the wrong solution.

To tackle this problem the voltages at the PV node are damped during the NR iterations whenever they make a large step out of the pre-defined solution space. Figure 24 can be used to demonstrate this graphically. The plot in Figure 24 shows results for a 2869 PEGASE bus test system and simulations were run on it for six different initial guesses for unspecified Q_G . The maximum bus voltage from the solution of the power flow problem for each initial guess was then plotted for two scenarios: without and with variable limiting technique enabled. The plots in the figure show that when variable limiting is not enabled, the voltage solution diverges to very high magnitudes (up to 10^4) and may not converge even in 100 iterations. However, when the variable limiting option is enabled, divergence is not observed, and the bounded bus voltages result in fast convergence.

To apply variable limiting in our prototype simulator, the mathematical expressions for the PV nodes in the system are modified as follows:

$$I_{CG}^{\Omega^{k+1}} = \varsigma \frac{\partial I_{CG}^{\Omega}}{\partial V_{RG}^{\Omega}} \underbrace{(V_{RG}^{\Omega^{k+1}} - V_{RG}^{\Omega^k})}_{\Delta V_{RG}^{\Omega}} + I_{CG}^{\Omega^k} + \varsigma \frac{\partial I_{CG}^{\Omega}}{\partial V_{IG}^{\Omega}} \underbrace{(V_{IG}^{\Omega^{k+1}} - V_{IG}^{\Omega^k})}_{\Delta V_{IG}^{\Omega}} + \frac{\partial I_{CG}^{\Omega}}{\partial Q_G^{\Omega}} (Q_G^{\Omega^{k+1}} - Q_G^{\Omega^k}) \quad (90)$$

where, $0 \leq \varsigma \leq 1$. The magnitude of ς is dynamically varied through heuristics such that

convergence to the correct physical solution is achieved in the most efficient manner. The heuristics depend on the largest delta voltage (ΔV_{RG}^Ω , ΔV_{IG}^Ω) step during subsequent NR iterations. If during subsequent NR iterations, a large step (ΔV_{RG}^Ω , ΔV_{IG}^Ω) is encountered, then the factor ς is decreased. The factor ς is scaled back up if consecutive NR steps result in monotonically decreasing absolute values for the largest error.

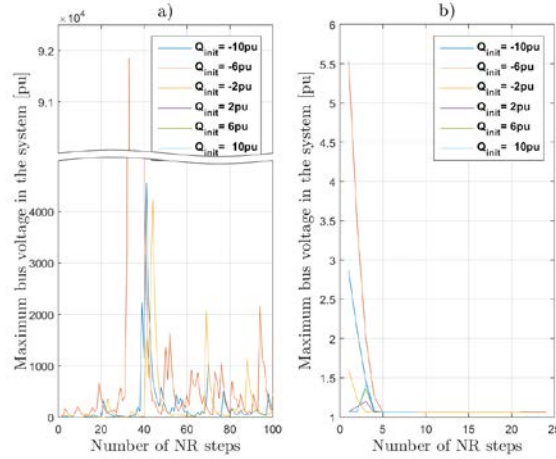


Figure 24: Voltage profile for maximum bus voltage in 2869 bus system: a) w/o Variable limiting b) with Variable limiting.

4.2.1.2 Voltage Limiting

An equally simple, yet effective, technique is to limit the absolute value of the delta step that the real and imaginary voltage vectors can make during each NR iteration. This is analogous to the voltage limiting technique used for diodes in circuit simulation, wherein the maximum allowable voltage step during NR is limited to twice the thermal voltage of the diode [51]. Similarly, for the power flow and three-phase power flow analyses, a hard limit is enforced on the normalized real and imaginary voltages in the system. The mathematical implementation of voltage limiting in our formulation is as follows:

$$(V_C^\Omega)^{k+1} = \min_{V_C^{min}} \max_{V_C^{max}} \left((V_C^\Omega)^k + \delta_S \min \left(|\Delta(V_C^\Omega)^k|, \Delta V_C^{max} \right) \right)$$

$$\min_{V_C^{min}} \max_{V_C^{max}} = \begin{cases} V_C^{max}, & \text{if } x > V_C^{max} \\ V_C^{min}, & \text{if } x < V_C^{min} \\ x, & \text{otherwise} \end{cases} \quad (91)$$

and $\delta_S = \text{sign}(\Delta(V_C^\Omega)^k)$ and $C \in \{R, I\}$ represents the placeholder for real and imaginary parts.

4.2.1.3 Limiting Methods for other System Variables

Similar to limiting of voltages during power flow and three-phase power flow problem, other system variables are also limited to constrain the behavior of the network

components in their physical space. In general, a good limiting technique is one that can exploit knowledge of system physics to well-define a narrow normal operating range within which the variable can be constrained. However, this is not always possible. For instance, the generator reactive power variable Q_G can have a wide range for its operating setpoint depending on the size of the generator. In such scenarios, the variables are limited by first mapping them into another variable for which we can define a better operating range. In case of generators, reactive power Q_G variables are limited by first mapping the Q_G 's into calculated currents $I_C^\Omega + \Delta(I_C^\Omega)^{k+1}$ at $(k+1)^{\text{th}}$ NR step, and then finding the new Q_G^{k+1} from the inverse function (f^{-1}) of limited currents $\left(I_C^\Omega + \Delta(I_C^\Omega)^{k+1}\right)$. Similar approaches can also be used to limit other system variables in future.

4.2.2 Homotopy Methods

Limiting methods may fail to ensure convergence for certain ill-conditioned and large test systems when solved from an arbitrary set of initial guesses. To ensure convergence for these network models to the correct physical solutions independent of the choice of initial conditions, we propose the use of homotopy methods.

Homotopy methods are not new to the field of power system simulation. Homotopy methods in the past have been used to study the voltage collapse of a given network or to determine the maximum available transfer capability [19]-[20]. They have also been researched for locating all solutions to a power flow problem [49], [63]. However, their use for enabling convergence for hard to solve positive sequence and three-phase power flow problems has been limited. Of the proposed methods for providing better convergence [9], [48] most have suffered from convergence to low voltage solutions or divergence. On the other hand, some of them have been developed for formulations that do not apply to both positive sequence as well as three-phase power flow [64] problems. Furthermore, none of the previously proposed homotopy methods are known to scale up to test systems that are of the size of the European or the US grids, and in general they are not extendable to the three-phase power flow problem.

4.2.2.1 General Introduction

In the homotopy approach, the original problem is replaced with a set of sub-problems that are sequentially solved. The set of sub-problems exhibit certain properties, namely, the first sub-problem has a trivial solution and each subsequent sub-problem has a solution very close to the solution of the prior sub-problem. Mathematically this can be described via the following expression:

$$\mathcal{H}(x, \lambda) = (1 - \lambda)F(x) + \lambda G(x) \quad (92)$$

where $\lambda \in [0, 1]$.

The method begins by replacing the original problem $F(x) = 0$ with $\mathcal{H}(x, \lambda) = 0$. The equation set $\mathcal{G}(x)$ is a representation of the system that has a trivial solution. The homotopy factor λ has the value of 1 for the first sub-problem, and therefore, the initial solution for $\mathcal{H}(x, \lambda)$ is equal to the trivial solution of $\mathcal{G}(x)$. For the final sub-problem that corresponds to the original problem, the homotopy factor λ has the value of zero. To generate sequential sub-problems, the homotopy factor is dynamically decreased in small steps until it has reached the value of zero.

In the following sections, we discuss two homotopy methods that are specifically developed for the power flow and three-phase power flow analyses i.e. Tx stepping and dynamic power stepping method.

4.2.2.2 Tx Stepping

We propose a new homotopy approach, “Tx Stepping,” that is specifically invented for the non-linearities observed in the power flow and three-phase power flow problems. In Tx stepping method, the series elements in the system (transmission lines, transformers etc.) are first “virtually” shorted to solve the initial problem that has a trivial solution. Specifically, a large conductance ($\gg G_{il}$) and a large susceptance ($\gg B_{il}$) are added in parallel to each transmission line and transformer model in the system. In case of three-phase power flow, a large self-impedance ($\gg Y_{\Omega\Omega}^{il}$) is added in parallel to each phase of the transmission line and transformer model. Furthermore, the shunts in the system, are open-circuited by modifying the original shunt conductance and susceptance values. Importantly, the solution to this initial problem results in high system voltages (magnitudes), as they are essentially driven by the slack bus complex voltages and the PV bus voltage magnitudes due to the low voltage drops in the lines and transformers (as expected with virtually shorted systems). Similarly, the solution for the bus voltage angles lies within an ϵ -small radius around the slack bus angle. Subsequently, like other continuation methods, the formulated system problem is then gradually relaxed to represent the original system by taking small increment steps of the homotopy factor (λ) until convergence to the solution of the original problem is achieved. Mathematically, the line and transformer impedances during homotopy for the power flow is expressed by:

$$\forall il \in T_X, xfmrs: \hat{G}_{il} + j\hat{B}_{il} = (G_{il} + jB_{il})(1 + \lambda\gamma) \quad (93)$$

and for the three-phase problem:

$$\begin{bmatrix} \hat{Y}_{aa}^{il} & \hat{Y}_{ab}^{il} & \hat{Y}_{ac}^{il} \\ \hat{Y}_{ba}^{il} & \hat{Y}_{bb}^{il} & \hat{Y}_{bc}^{il} \\ \hat{Y}_{ca}^{il} & \hat{Y}_{cb}^{il} & \hat{Y}_{cc}^{il} \end{bmatrix} = \begin{bmatrix} Y_{aa}^{il}(1 + \gamma\lambda) & Y_{ab}^{il} & Y_{ac}^{il} \\ Y_{ba}^{il} & Y_{bb}^{il}(1 + \gamma\lambda) & Y_{bc}^{il} \\ Y_{ca}^{il} & Y_{cb}^{il} & Y_{cc}^{il}(1 + \gamma\lambda) \end{bmatrix} \quad (94)$$

where, G_{il} , B_{il} , and $Y_{\Omega\Omega}^{il}$ are the original system impedances and $\hat{G}_{il}$, $\hat{B}_{il}$, and $\hat{Y}_{\Omega\Omega}^{il}$ are the system impedances used while iterating from the trivial problem to the original problem. The parameter γ is used as a scaling factor for the conductances and susceptances. If the

homotopy factor (λ) takes the value of one, the system has a trivial solution and if it takes the value zero, the original system is represented.

Along with ensuring convergence for a challenging steady-state problem, Tx stepping can avoid the undesirable low voltage solutions for the positive sequence power flow and three-phase power flow problem since the initial sub-problem within the homotopy path results in a solution with high system voltages, and each subsequent step of the homotopy approach continues and deviates ever so slightly from this initial solution, thereby guaranteeing convergence to the high voltage solution for the original problem.

Handling of Transformer Phase Shifters and Taps

To “virtually short” a power system, we must also account for transformer taps tr^Ω and phase shifting angles θ^Ω . In a “virtually” shorted condition, all the nodes in the system must have complex voltages that are near the slack bus or PV bus complex voltages, which can be intuitively defined by a small epsilon norm ball around these voltages. Therefore, to achieve the following form, we must modify the transformer taps and phase shifter angles such that at $\lambda = 1$, their turns ratios and phase shift angles correspond to a magnitude of 1 pu and 0° , respectively. Subsequently, the homotopy factor λ is varied such that the original problem is solved with original transformer tap and phase shifter settings. This can be mathematically expressed as follows:

$$\forall i \in xfms : \hat{tr}_i^\Omega = tr_i^\Omega + \lambda(1 - tr_i^\Omega) \quad (95)$$

$$\forall i \in xfms : \hat{\theta}_i^\Omega = \theta_i^\Omega - \lambda\theta_i^\Omega \quad (96)$$

Handling of Voltage Control for Remote Buses

To achieve a trivial solution during the first step of Tx stepping it is essential that we also handle remote voltage control appropriately. Remote voltage control refers to a device on node $\mathcal{O}$ in the system controlling the voltage of another node $\mathcal{W}$ in the system. This behavior is highly non-linear and if not handled correctly can result in divergence or convergence to a low voltage solution. Existing commercial tools for power flow and three-phase power flow analyses have difficulties dealing with this problem and suffer from lack of robust convergence when modeling remote voltage control in general. With Tx stepping we can handle this problem efficiently and effectively. We first incorporate a “virtually short path” between the controlling node ($\mathcal{O}$) and the controlled node ($\mathcal{W}$) at $\lambda = 1$, such that the device at the controlling node can easily supply the current needed for node $\mathcal{W}$ to control its voltage. Then following the homotopy progression, we gradually relax the system such that the additional line connecting the controlling node ($\mathcal{O}$) and controlled node ($\mathcal{W}$) is open at $\lambda = 0$.

Implementation of Tx Stepping in Equivalent Circuit Formulation

Unlike traditional implementations of homotopy methods, in equivalent circuit formulation we do not directly modify the non-linear set of mathematical equations, but

instead embed a homotopy factor in each of the equivalent circuit models for the power grid components. In doing so we allow for extension of Tx-stepping method to any power system model within the equivalent circuit formulation framework, without loss of generality. Furthermore, we ensure, that the physics of the system is preserved along the complete homotopy path. Figure 25 and Figure 26 demonstrates how the homotopy factor is embedded into the equivalent circuit of the transmission line and transformer, respectively.

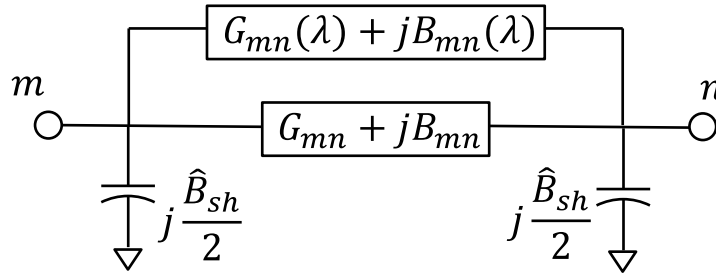


Figure 25: Homotopy factor embedded in transmission line equivalent circuit.

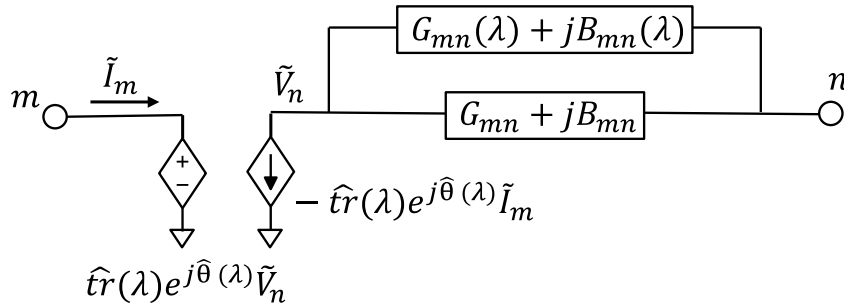


Figure 26: Homotopy factor embedded in transformer equivalent circuit.

Convergence

The proposed Tx stepping method is within the subset of homotopy methods and to ensure convergence (i.e. be globally convergent) for any homotopy method the following conditions must be met [70]:

- i. Defined path for the homotopy method i.e. $c(\lambda) \in \mathcal{H}^{-1}(0)$ with $(x, \lambda) \in \text{range}(c)$ must be smooth and should exist.
- ii. If a curve c exists, then it should intersect the final solution at $\lambda = 0$.

The first condition can be met through implicit function theorem and requires that the Jacobian ($\mathcal{H}'(\lambda)$) of the homotopy function is of full rank for all values of λ along the curve. In the power flow or three-phase flow problem this corresponds to the Jacobian (J) matrix of the network constraints that in the case of equivalent circuit formulation are the linearized Kirchhoff's current laws at each node in the system. Based on domain knowledge of power systems, it is understood that the network Jacobian matrix is singular if the system is operating at its limits (tip of the nose curve) [57] or beyond (infeasible system). This is an unlikely case from the physics perspective for any well-

conditioned system over the range of λ , and therefore, the Jacobian (J) is generally full rank over the complete range of λ . In rare cases, the network Jacobian ($\mathcal{H}'(\lambda)$) defined for λ value on the curve (i.e. $c(\lambda)$, $\lambda \in [1, 0]$) can be singular. This is either because the system is infeasible such that no further power transfer is possible or that the system is highly ill-conditioned and is operating at the tip of the nose curve. For such infeasible or highly ill-conditioned networks, it is possible that a rank deficient Jacobian may be encountered along the homotopy curve. To achieve robust convergence for such networks that are either infeasible or highly ill-conditioned, optimization-based methods [71] or techniques for structural perturbations are used [72]. An example of the optimization-based method is the addition of current sources to all system nodes during Tx-stepping while minimizing their value [71], whereas an example for structural perturbation includes adding and removing transmission lines to the network dynamically during the homotopy path. In the optimization-based method, the network is guaranteed to have a feasible solution for some value of complex current sources, thereby asserting the existence of a full rank Jacobian matrix. Similarly, a full rank Jacobian matrix can also be ensured for ill-conditioned systems by structurally relaxing the weak part of the grid by adding more lines at the start and gradually removing them for the original problem.

The second condition is more easily met and is linked to existence theorems in non-linear analyses [70]. If some boundary condition exists that prevents the curve from extending to infinity prior to intersecting the solution at $\lambda = 0$, then this condition is met. In our formulation, different limiting techniques ensure that the solution at any point on the curve c does not diverge and extend to infinity.

4.2.2.3 Dynamic Power Stepping

Another homotopy technique that can ensure robust convergence for systems that have a low percentage of constant voltage nodes in the system is the dynamic power stepping method. Existing distribution systems and small transmission systems tend to belong to this class of systems and, therefore, dynamic power stepping can be applied to robustly obtain their steady-solution by solving either the power flow or the three-phase power flow problem. This method has been described for the positive-sequence power flow and three-phase power flow problem in [34], [73] and is analogous to the source stepping and Gmin stepping approaches in standard circuit simulation solvers.

In the dynamic power stepping method, the system loads and generation are scaled back by a factor of β until the convergence is achieved. If these loads and generations are scaled down all the way to zero, then the constraints for the PQ buses in the system result in linear network constraints. Similarly, current source non-linearities of the PV buses that are due to the constant real power are also eliminated. Therefore, by applying the power stepping factor, the non-linearities in the system are greatly eased and convergence is easily achieved. Upon convergence, the factor is gradually scaled back up to unity to solve the original problem. In this method, as in all continuation methods, the

solution from the prior step is used as the initial condition for the next step. The mathematical representation of dynamic power stepping method for the three-phase power flow and positive sequence power flow problem is as follows:

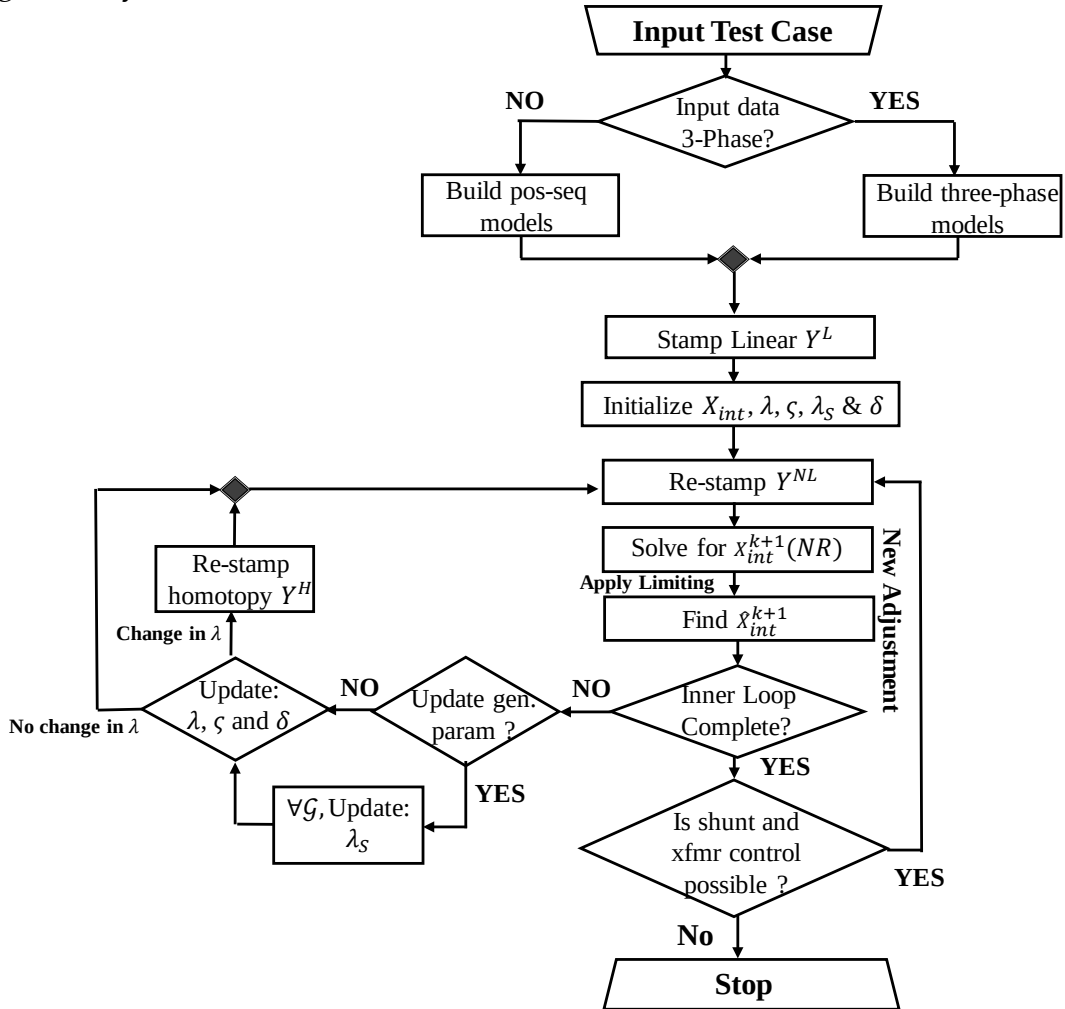
$$\forall G \in PV: \hat{P}_G^\Omega = \beta P_G^\Omega \quad (97)$$

$$\forall L \in PQ: \hat{P}_L^\Omega = \beta P_L^\Omega \text{ and } \hat{Q}_L^\Omega = \beta Q_L^\Omega \quad (98)$$

where, PQ are all load nodes and PV are all generator nodes.

4.2.3 Algorithm

In this section, the Algorithm 4-3 shows the recipe for solving the positive-sequence as well as three-phase power flow problem in the equivalent circuit approach with the use of circuit simulation methods. This algorithm is implemented in our novel tool: SUGAR (Simulation with Unified Grid Analyses and Renewables) and can be used to solve any positive-sequence power flow problem or three-phase power flow problem without the loss of generality.



Algorithm 4-3: SUGAR algorithm for solving positive sequence and three-phase power flow problems.

The developed solver begins with parsing the input file and gauging if the input data are for the positive sequence or three-phase power flow problem. Based on the type of data (i.e. power flow or three-phase power flow data), it starts with building the system models. An empty Jacobian matrix structure is initialized based on the size of the system and non-changing linear models (Y^L, J^L) are first stamped in it. These stamps remain constant throughout the NR iterations. Input state variables and other continuation parameters ($x_0, \delta, \zeta, \lambda, \lambda_s$) are then initialized following which the non-linear models are stamped (Y^{NL}, J^{NL}) and NR is applied with limiting methods enabled to calculate the next iterate for the voltages, the generator reactive powers and any other continuous control variables ($\hat{X}^{k+1}$). In the solver, from the practical point of view, the available initial conditions in the input file are first used as the solver's initial conditions. Generally, with the use of proposed limiting methods, the system solves within 7 to 10 iterations with these initial conditions. However, in cases where the system is ill-conditioned or lacking a good initial guess, the solver begins to gradually increase the homotopy factor (λ) until a trivial solution is found (this method does not require a good initial guess as homotopy methods have trivial solution for the first step). Once a trivial solution is obtained, homotopy factors and other continuation factors for generators are dynamically updated (in this case decreased), and homotopy models (Y^H, J^H) are stamped or re-stamped in the Jacobian to ensure convergence to the correct physical solution for the original problem at ($\lambda = 0$). Upon convergence of the inner loop, remaining controllable switched shunts and transformer taps are adjusted and the inner loop is repeated until the final solution is achieved. In cases, where continuous models are used for the control of discrete shunts and discrete transformer taps, a final loop is implemented to snap them to their closest discrete values.

4.2.4 Results

Here we report on some of the results obtained via the use of the equivalent circuit framework with the use of circuit simulation methods. To run the test cases and validate our approach, we integrated these methods into SUGAR. The results from SUGAR will demonstrate the ability of our framework to solve ill-conditioned, large real-life, and in general hard-to-solve positive-sequence and three-phase power flow test cases from arbitrary initial conditions. The following result section is divided into positive-sequence power flow results sub-section and three-phase power flow results sub-section.

4.2.4.1 Positive Sequence Power Flow Results

The first few sub-sections discuss results for the positive sequence power flow analysis. The set of results include ill-conditioned and hard to solve test cases as well as large test cases. However, first we demonstrate the efficacy of the proposed circuit simulation methods.

Efficacy of Circuit Simulation Methods

We use the first set of results to demonstrate that the use of circuit simulation methods developed within this section can significantly improve the robustness of convergence within the equivalent circuit framework. The section shows how the results obtained in the equivalent circuit framework with the use of circuit simulation methods fare against those obtained in the equivalent circuit framework without the use of circuit simulation methods.

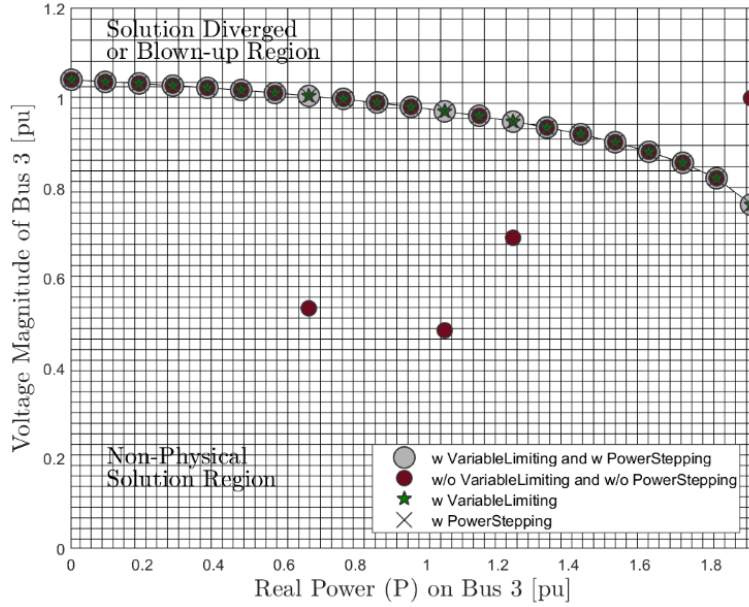


Figure 27: Solution of bus 3 voltage for IEEE 14 bus test system with increasing loading factors with and without circuit simulation methods.

Experiment 1

In this experiment, positive-sequence power flow simulations are run on the IEEE 14 bus test system (from flat start) in steps of increasing loading factors (up to 4x) for the following four scenarios: 1) both power stepping and variable limiting option disabled, 2) with power stepping option enabled and variable limiting disabled, 3) with variable limiting option enabled and power stepping disabled, and 4) both power stepping and variable limiting option enabled. The solutions for the bus 3 voltage magnitude at the end of each simulation are then plotted in Figure 27. The plot shows that convergence to the correct physical solution is achieved for each simulation instance when either variable limiting, or power stepping option is enabled. However, without these options enabled in SUGAR, the solution in many simulation instances has either converged to the wrong solution or diverged altogether.

Experiment 2

In this experiment, power flow simulations are run on the 2869 PEGASE test system and 9241 PEGASE test system for 20 different initial guesses of Q_G values that are uniformly distributed in the range of -10 pu and 10 pu. All 20 simulations are run for each of these solver settings under the same four scenarios as were used in the case for Experiment 1. The convergence results plotted in Figure 28 show that without the use of circuit simulation techniques, most of the test case instances either diverge or converge to the wrong solution. Convergence to the correct physical solution is only observed when both variable limiting and power stepping are enabled.

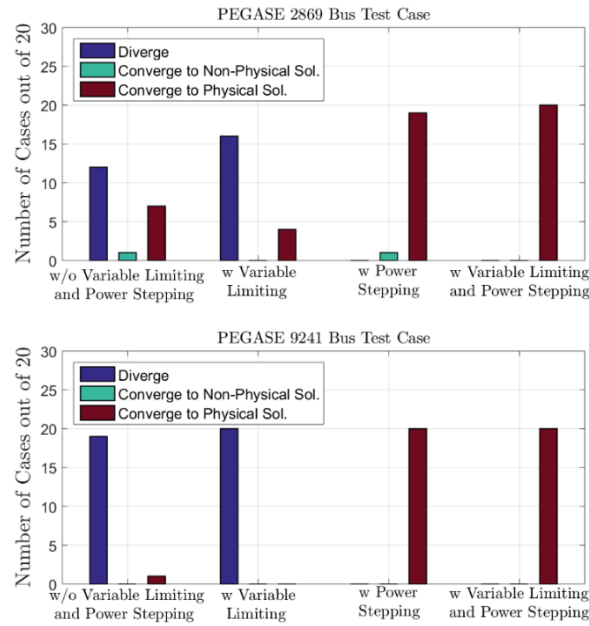


Figure 28: Power flow results for 2869 bus and 9241 bus test systems with and without circuit simulation techniques.

Experiment 3

To show the efficacy of circuit simulation methods in this experiment, contingencies were simulated on two hard to solve test-cases that represent different operating conditions for a real sub-network within the US power grid network models. The base cases for both test systems are first solved via the Tx-stepping method whose solutions are then used as initial conditions for the set of contingencies that were further run for two settings i) without the use of circuit simulation methods in SUGAR ii) with the use of circuit simulation methods in SUGAR. The contingencies in the contingency set include the loss of the largest 10% of the online generators and loss of 10% of the highest capacity lines and transformers taken off-line one at a time from the base case to create a single contingency instance within the contingency set.

The results in the Table 6 validate that the use of circuit simulation methods when applied to equivalent circuit formulation can significantly increase the robustness of the power flow solver as in the case when circuit simulation methods were disabled, convergence to the correct physical solution was achieved for far fewer contingency instances.

Table 6: Comparison of SUGAR with and without circuit simulation techniques

Case Id	# Bus	# Total Cases	SUGAR w/o Circuit Simulation Methods		SUGAR with Circuit Simulation Methods	
			Converge	Diverge /Infeasible	Converge	Diverge /Infeasible
Case 1	5944	754	735	19	750	4*
Case 2	7029	801	706	95	793	8*

*These cases were confirmed to be infeasible following the methodology in [71].

Ill-Conditioned Test Cases

In this sub-section, we demonstrate results of our approach when applied to ill-conditioned test cases. A large condition number for a given matrix indicates that the matrix and the system corresponding to that matrix are ill-conditioned. In the power flow problem, the matrix of interest is the Jacobian that is used to calculate the Newton step for each NR iteration. If the condition number of the Jacobian matrix is large at the solution point, then the system is assumed to be ill-conditioned.

The following set of results are generated from SUGAR with circuit simulation methods enabled and are compared against those produced by the standard commercial tools used in the industry today.

Experiment 1: Ill-conditioned test cases in literature

The 11-bus, 13-bus, and 43-bus test cases from the power system literature [57] are considered to be ill-conditioned systems. However, it is systematically shown in [57] that out of these three systems, the 11-bus system is the only genuine ill-conditioned system with a maximum loading of 99.82 %. The 13-bus system is not an ill-conditioned system and can easily be solved via any power flow method, and the 43-bus test case has a maximum loading of 58%, and the system is infeasible for base loading of 100%.

Table 7 shows the comparison of the results for a modified 11 bus ill-conditioned test case at 99.82% loading for different set of initial conditions. With standard commercial tools, for most initial conditions, the system converges to a low-voltage solution or diverges altogether. The commercial solver can only converge to the correct physical solution if the initial condition supplied is the solution itself. However, SUGAR can converge to the correct physical solution from arbitrary initial conditions when Tx Stepping is applied.

Table 7: Comparison of results for modified 11 bus test case

Initial Condition		Ill Conditioned 11 Bus Test Case	
V_{mag} (pu)	V_{ang} (°)	Standard Commercial Tool ²	SUGAR ¹
1	0	Low Voltage	High Voltage
0.76	23	Low Voltage	High Voltage
0.71	45	Low Voltage	High Voltage
High Voltage	High Voltage	High Voltage	High Voltage

1. Tx Stepping was enabled while running simulations in SUGAR

2. Full Newton Raphson was the solver used in Standard Commercial Tool

Experiment 2: A large ill-condition system at operating point

Another notable example of a case with a higher condition number at the operating point is the 13659-bus system from the PEGASE test cases. At the solution point, the approximate condition number of the system Jacobian is $1.7\text{e}8$. Figure 29 shows convergence results for this test case from ten arbitrary initial conditions for a standard commercial tool and SUGAR. The ten initial conditions were chosen uniformly from the set of:

$$V_R \in [0.6, 1.1], \quad V_I = \{x \in \mathbb{R}^n \mid x = 1 - V_r\} \quad (99)$$

From the set of 10 initial conditions, the standard commercial tool converged to the correct physical solution once, diverged 8 times, and converged to the angular unstable solution one time. SUGAR, however, with variable limiting and voltage limiting enabled was able to converge to the correct physical solution for all ten initial guesses.

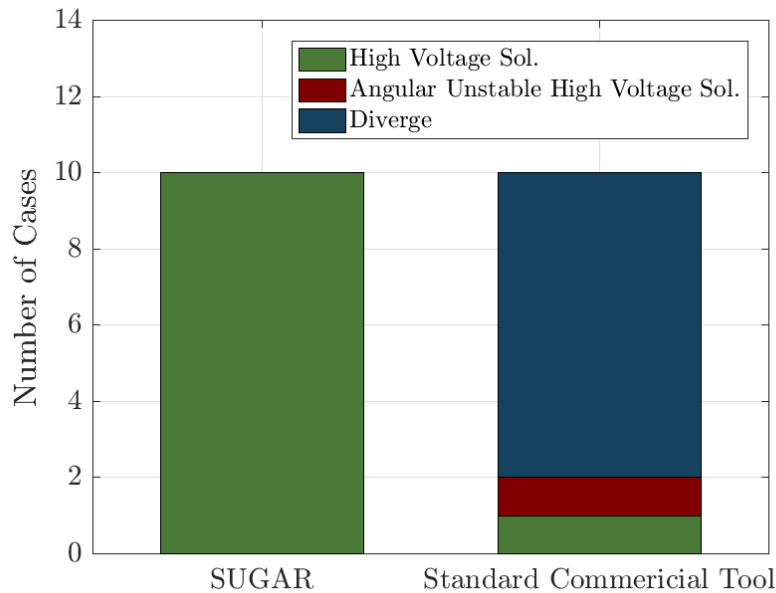


Figure 29: Results for 13659 buses PEGASE system.

Large Test Cases

In this experiment we demonstrate that SUGAR can robustly solve any large test case from arbitrary initial conditions to result in correct physical solution. Figure 30 shows the results for six distinct test systems that represent the eastern interconnection network of the US power grid under different loading conditions (Summer/Winter) and time periods (2017, 2018, 2021, 2026 etc.). The simulations were run on these systems from a set of different initial conditions that were uniformly chosen from the sets of:

$$V_{ang} \in [-50, 50], \quad V_{mag} \in [0.6, 1] \quad (100)$$

The vertical and horizontal axes of the figure represent the set of initial conditions (V_{ang} , V_{mag}) for a given case, respectively and the legend within each sub-graph represent the numbers of nodes in the test system. If the case converged to a correct physical solution, it is marked via a green mark; whereas if the case diverged then it is marked via a red mark. The figure indicates that SUGAR was able to converge to correct physical solution for all the six large eastern interconnection systems independent of the choice of initial conditions. The run-time per iteration for the eastern interconnection test cases in SUGAR is comparable to other available commercial tools (less than 0.4s per iteration). The total simulation time for the test cases was dependent on the choice of initial conditions.

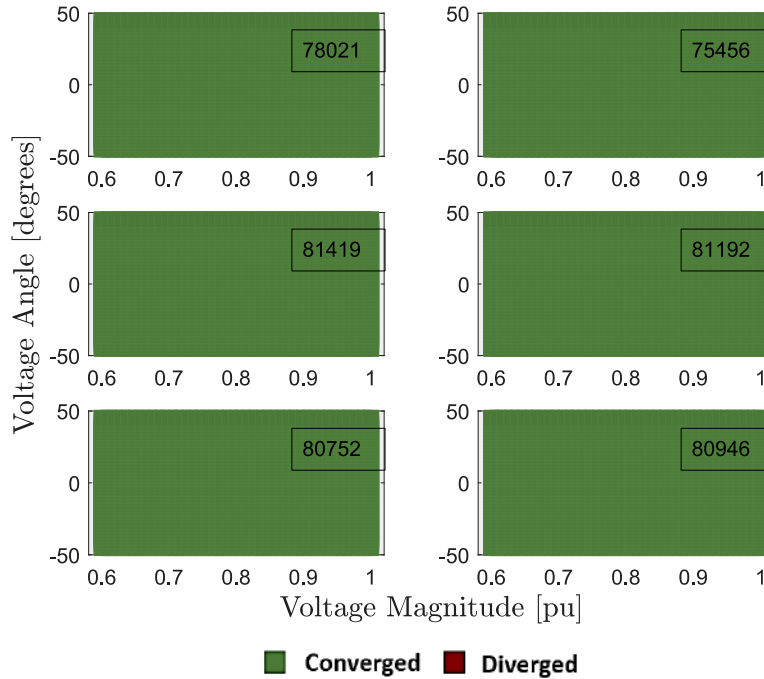


Figure 30: Convergence sweep of large cases that represent Eastern Interconnection from range of initial conditions (number of nodes for each test system given in the legend box).

We also repeated the same experiment on the publicly available SythenticUSA and ACTIVgs70k test cases [74] and demonstrated the same robust convergence as in the case of Eastern Interconnection test cases, as shown in Figure 31.

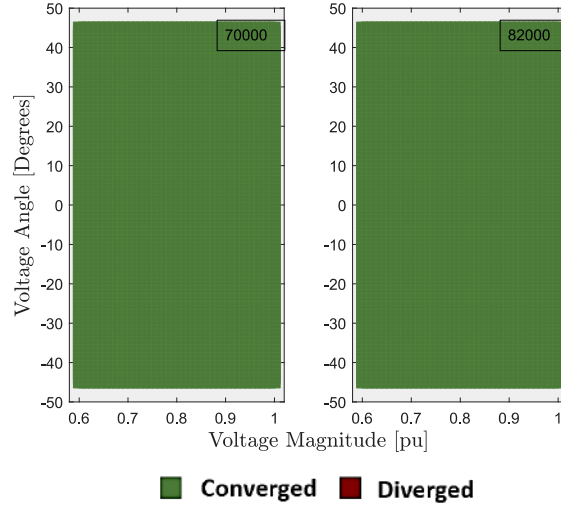


Figure 31: Convergence plot for ACTIVgs70k (left) and SyntheticUSA (right) testcases.

We performed a similar experiment to compare the robustness of SUGAR tool against a standard commercial tool. To conduct the experiment, we ran three real and two synthetic [74] eastern interconnection sized systems for 15 different initial conditions in both the SUGAR tool and the standard commercial tool. The set of initial conditions for this experiment for all buses were identical and were uniformly sampled from:

$$V_{ang} \in [-40, 40], V_{mag} \in [0.9, 1.1]. \quad (101)$$

The results in Table 8 show that from any of the 15 different initial conditions, the standard commercial tool was unable to solve the system, whereas SUGAR with Tx stepping enabled was able to converge to the correct physical solution in all cases.

Table 8: Convergence performance for large eastern interconnection test cases

Case Name	# Nodes	Standard Tool		SUGAR	
		# Converge	# Diverge	# Converge	# Diverge
Case 1	80778	0	15	15	0
Case 2	76228	0	15	15	0
Case 3	81904	0	15	15	0
SyntheticUSA	82000	0	15	15	0
ACTIVSg70k	70000	0	15	15	0

Contingency Analysis

To further demonstrate the robustness of our approach, we consider a set of scenarios wherein we plan a realistic contingency on large test cases and other hard to solve test cases. We compare the obtained results from SUGAR tool against those produced by the standard commercial tool.

Experiment 1: Contingency on Eastern Interconnection Test Cases

In this first experiment, we run contingency analysis on test cases that represent different operating and loading conditions for the U.S. eastern interconnection network. The extreme contingencies in these cases are defined by loss of either two (N-2) or three (N-3) generators in the system. To obtain and further compare the results, we solve these contingency instances with both the standard commercial tool and the SUGAR tool. The initial conditions for all the cases during simulation are chosen to be the solution state prior to the contingency i.e. solution of the base case (thereby suggesting that the system initial conditions depict the pre-contingency operating state).

Table 9: Contingency analysis for large test cases

Case	Contingency Type	No. of Buses	Standard Commercial Tool	SUGAR
Case 1	N-2	75456	Diverged	Converged
Case 2	N-2	78021	Diverged	Converged
Case 3	N-3	80293	Diverged	Converged
Case 4	N-3	81238	Diverged	Converged

The results in Table 9 demonstrate that while SUGAR was able to converge to the correct solution for all the contingency instances, the standard commercial tool diverged for all thereby further strengthening the argument for robustness of our framework. Importantly, robustness of our tool toward solving contingencies can be extremely vital to grid operation and planning engineers who are required by NERC to evaluate each failed N-1 contingency [75].

Experiment 2: Contingency of hard-to-solve real life test cases

To further strengthen our argument for robust of SUGAR for contingency analysis, we run this experiment. In this example, we consider a yet another real-life test grid that represents a sub-set of the US grid. The case is known to be hard-to-solve and we perform N-1 contingency analysis on it. The set of contingencies include loss of 10% of the highest capacity links (transformers and branches) and the loss of 10% of the largest online generators taken off-line one at a time from the base case configuration to create a single contingency instance. This results in total number of 774 contingency instances within the contingency set. To run the contingency analysis, we first solve the base case with the use of the Tx-stepping method. With the solution of the base case as the initial condition,

we simulate the contingency instances in SUGAR. The standard commercial tool was unable to solve the base case, and therefore, we were not able to simulate the contingency instances on those. Table 10 results from SUGAR contingency runs show that SUGAR was able to solve the base case as well as all contingency instances robustly.

Table 10: N-1 contingency analysis on set of critical equipment.

Solver	Number of Contingencies	System Convergence	
		Converged	Infeasible
SUGAR	774	774	0
Standard Commercial Tool	774	NA	NA

4.2.4.2 N-1+1 Analysis (Contingency Analysis + Corrective Action)

In this experiment we simulated yet another real-life test case that represents an electric grid network from Africa that it is pushed to its limits. For this experiment, we first perform N-1 contingency analysis on the test case and based on the results we recommend a corrective action. We refer to this corrective methodology as N-1+1 analysis. In the set of contingencies for this analysis, we consider all the transformers, lines and generators dropped one at a time that resulted in a total of 717 contingency instances. Of these 717 contingency instances, 684 were found to be feasible whereas 33 instances were found to be infeasible. The 33 of them were confirmed to be infeasible based on the methodology documented in [71]. Furthermore, from the results gathered from the contingency analysis, it was found that akin to the base case, a significant number of simulation instances resulted in very high voltages as shown in Figure 32.

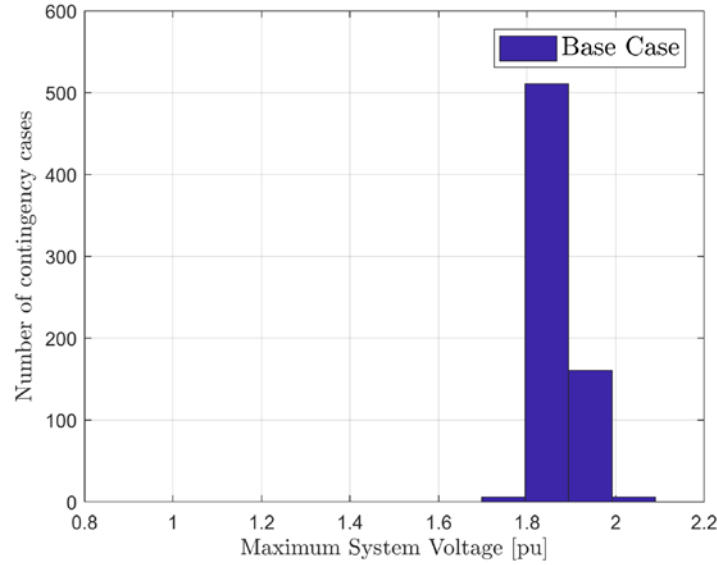


Figure 32: Maximum bus voltage range for contingency analysis.

Therefore, as a corrective action to improve the voltages in this system, we propose N-1+1 analysis. The algorithm for this analysis is as follows:

N-1+1 Algorithm

1. **procedure:**
 2. **run** $N - 1$ contingency
 3. **identify** all regions (R_{inf}) with abnormal bus voltages in the system
 4. **for** $\forall R_{inf}$:
 - a. **add** reactive power compensating device to every bus in R_{inf} ($N + 1$ scenario)
 - b. **redo** $\rightarrow N - 1$, find number of infeasible cases (N_{inf}) and range of voltages ($V_{max} - V_{min}$)
 5. **choose**, $N + 1$ scenario, with fewest infeasible cases (N_{inf}) and lowest spread of system voltages.
-

Based on the algorithm, we added a reactive power compensating device to the most sensitive bus in the system and were able to reduce the system voltages for the base case and the contingency cases while resulting in fewer infeasible cases. The maximum bus voltage range pre- and post- corrective action is shown in Figure 33.

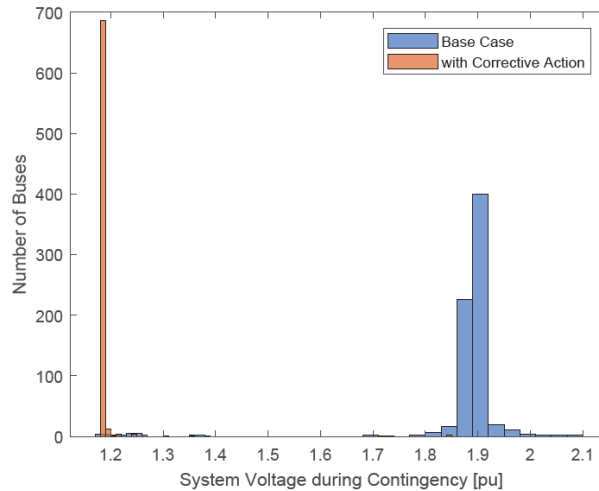


Figure 33: System bus voltage pre and post corrective action.

4.2.4.3 Three-Phase Power Flow Results

In the following section, we discuss results for three-phase power flow analysis. Like positive sequence power flow analysis, we first demonstrate the efficacy of circuit simulation methods toward robust convergence of distribution grid three-phase test cases.

Efficacy of Circuit Simulation Methods

In this experiment, we demonstrate that the use of circuit simulation methods in three-phase power flow simulation can ensure convergence for hard-to-solve test cases that were otherwise found unsolvable without the use of these methods. To demonstrate one such example, we extended the standard 145 node positive-sequence transmission system model into a balanced three-phase network model. Figure 34 plots the convergence results for this test case with and without the use of the dynamic power stepping technique. It is shown that without the use of dynamic power stepping method, the test system did not converge to an acceptable solution within the maximum number of allowable iterations; however, with the use of dynamic power stepping method, the system robustly converged to the correct physical solution.

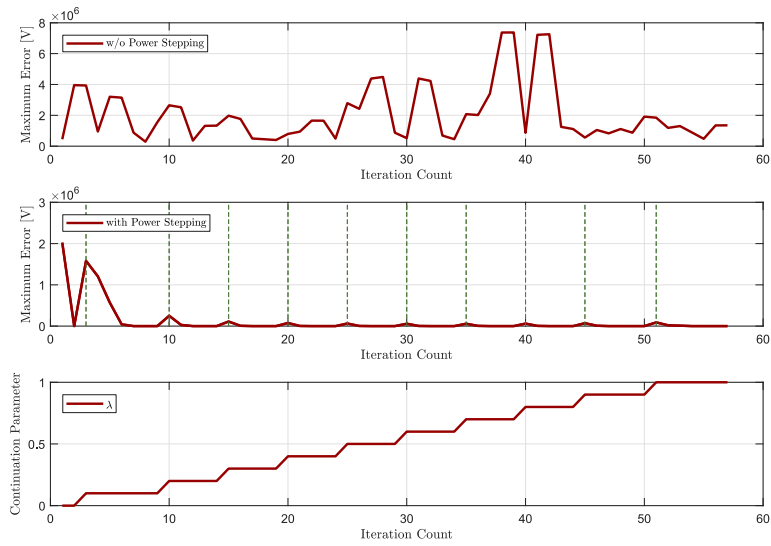


Figure 34: Convergence of 145 bus test case for three-phase power flow with (middle) and without (top) power stepping. For the power stepping case, the green dotted line represents the change in continuation factor λ whose evolution is shown in the bottom plot.

Taxonomy Feeder Test Cases and other Large Test Cases

Table 11 documents the results obtained from the SUGAR three-phase solver for standard taxonomy feeder cases and three large meshed test cases. The standard taxonomy feeder cases include both balanced and unbalanced three-phase test cases. The first two of the meshed test cases are high density urban meshed low voltage networks more commonly known as 342-Node Low Voltage Network Test Systems [76]. The third meshed test system is a high voltage 9241 node PEGASE transmission system that was extended to a balanced three-phase model from the positive sequence model. All these cases were simulated in SUGAR three-phase solver to validate the solver accuracy by comparing the obtained results against those produced from standard distribution power flow tool GridLAB-D. Slight differences (less than 1e-2) in the results were observed for cases between SUGAR and GridLAB-D and can be attributed to the default values used for unspecified parameters (e.g. neutral conductor resistance) in GridLAB-D.

Table 11: SUGAR three-phase results for taxonomy feeders and large cases

Cases	#Nodes	Iter. Count	Deviation from GridLAB-D	
			Max. ΔV_{mag} [pu]	Max. ΔV_{ang} [°]
GC-12.47-1	36	3	9.10E-06	6.6E-04
R1-12.47-1	2455	5	8.73E-04	9.94E-03
R2-12.47-3	2311	5	6.56E-04	1.32E-02
R3-12.47-3	7096	5	1.94E-03	3.89E-02
R4-12.47-1	2157	5	6.81E-04	9.61E-03
R5-12.47-5	2216	5	5.44E-05	4.20E-03
Network Model 1	1420	3	3.38E-03	2.14E-03
Network Model 2	1420	3	3.83E-03	6.00E-03
case9241pegase*	12528	5	NA [#]	NA [#]

* 9241 bus PEGASE transmission test case was extended to three-phase model

[#]The following case did not run in GridLAB-D

4.3 Combined Transmission and Distribution System Analyses

There is a growing adoption of variable and intermittent sources of generation especially wind and solar in the power systems across the globe. This high levels of penetration of renewables will result in much narrower operational margin than what's available today, thereby significantly affecting the reliability of the grid. In order to ensure that the reliability of the grid is not affected, interdependencies between the transmission grid and distribution grid (wherein a significant fraction of solar is likely to be installed) will have to be clearly understood while enabling control based on the knowledge of the operating state for both the transmission as well as the distribution grid. This was apparent when a transmission system operator in PJM coordinated with the Sturgis, Michigan distribution grid to avoid a blackout by utilizing 6 MW of distributed generation back in 2013 [77]. To securely and reliably enable control actions such as this, the operators and planners of the grid may require new simulation capabilities that will navigate through the invisible boundaries that exists today between the transmission and distribution grid analyses and solution methodologies. The existing simulation framework for power system analyses is incapable of capturing these interdependencies between the transmission and distribution grids. No standard tool exists in the industry today that can jointly model the transmission and distribution grids while ensuring robust steady-state solution for the same. This lack of simulation capability was highlighted in an ARPA-E workshop to identify paths to large-scale deployment of renewable energy resources, where one speaker noted that the "tools are not graceful in considering penetration levels at which much of the thermal fleet could get de-committed," and that "studies do not co-simulate impact of renewable injection into receiving AC systems" [6]. Another speaker noted that the tools for simulating increasingly coupled transmission and distribution systems "are not well integrated" [7].

In this Section we demonstrate that our equivalent circuit framework can jointly model the transmission and distribution (T&D) grid without loss of generality and ensure robust convergence for the same. Moreover, as shown in this report, the circuit simulation techniques developed within this report are directly applicable to both the power flow and the three-phase power flow problem and thereby allow us to extend the same to be used in the case of combined transmission and distribution analyses for robust convergence.

4.3.1 Background

The existing research literature in the field of combined transmission and distribution simulation is limited primarily due to the use of disparate methods for the transmission and distribution analyses formulation and algorithms [65]-[68]. Amongst these, the most common methodology for combined T&D simulation is to model the transmission network via positive sequence model and the distribution network via three-phase model and to couple the two. The assumption here is that the three-phases of the transmission network are balanced at the point of interconnection (POI). In general, most of these methods tend to couple the transmission and distribution systems via an interface and then solve the two via disparate methods [66]-[68]. For instance, [66] models the transmission grid via PowerWorld and the distribution grid via GridLab-D. The integrated simulation is then performed by running individual sub-circuits in their respective tools and then by exchanging variables via a communication port. Similar approaches are also used in [67]-[68]. Such approaches result in inheritance of legacy robustness issues from the positive sequence as well as the three-phase solvers, wherein a failure of either tool to solve a sub-circuit (transmission or distribution test case) results in complete breakdown of the framework. Moreover, due to the use of disparate tools/methods for solving the individual transmission and distribution test cases, it is difficult to develop methods that are generic and can guarantee convergence for both transmission and distribution systems. A more novel master-slave approach toward solving the combined simulation is proposed in [65], wherein the combined problem is solved in a distributed way. In this method, the problem is split into a transmission power flow and several distribution power flow sub-problems that are then solved via separate algorithms to capture the different features of transmission and distribution grids. However, the methodology has mostly been tested on unrealistically small sized systems with no claims of robust convergence for the individual sub-systems.

Another approach for combined simulation of transmission and distribution systems is to model the complete three-phase network of the transmission system and then to couple that with three-phase networks of the distribution systems [69]. This approach does have to assume that the transmission grid operation is balanced, and thus allows for simulation of unbalanced transmission grids. However, the primary limitations to this approach are the general lack of three-phase data for the transmission network, and the lack of research toward ensuring robust convergence of three-phase transmission networks.

4.3.2 General Methodology

To robustly solve for the combined transmission and distribution network, we propose the use of equivalent circuit approach that is previously discussed in Section **Error! Reference source not found.** In this approach, we represent the coupled transmission and distribution grid as an aggregated equivalent circuit, and we use exact or relaxed NR methods to solve for the set of non-linear equations defined by that aggregated circuit. The aggregated equivalent circuit for the transmission system is the positive sequence network of the same and assumes balanced operation of the grid, whereas the equivalent circuit of the distribution system models each phase of the distribution grid individually, thereby allowing for analysis of unbalanced operation of the grid. In the equivalent circuit approach, we can couple the transmission grid model to the distribution grid model without modifications and solve them as such. As the entire grid can be thought of as a circuit, coupling of the two circuits for combined simulation is fundamental to the circuit analysis domain. To couple the two models, we model the positive sequence currents consumed by the distribution grid with current controlled current sources connected to the edge of the transmission system. Similarly, we model the three-phase voltages at the sub-station level of the distribution system by voltage-controlled voltage sources that are functions of the voltages at the transmission edge of the system.

4.3.3 Coupling port for Combined Transmission and Distribution Circuit

In this sub-section, we develop the port that is used to couple the transmission and distribution sub-circuits for combined transmission and distribution simulations. The positive sequence transmission network and three-phase distribution network are coupled at the POI to run combined simulation via the circuit shown in Figure 35.

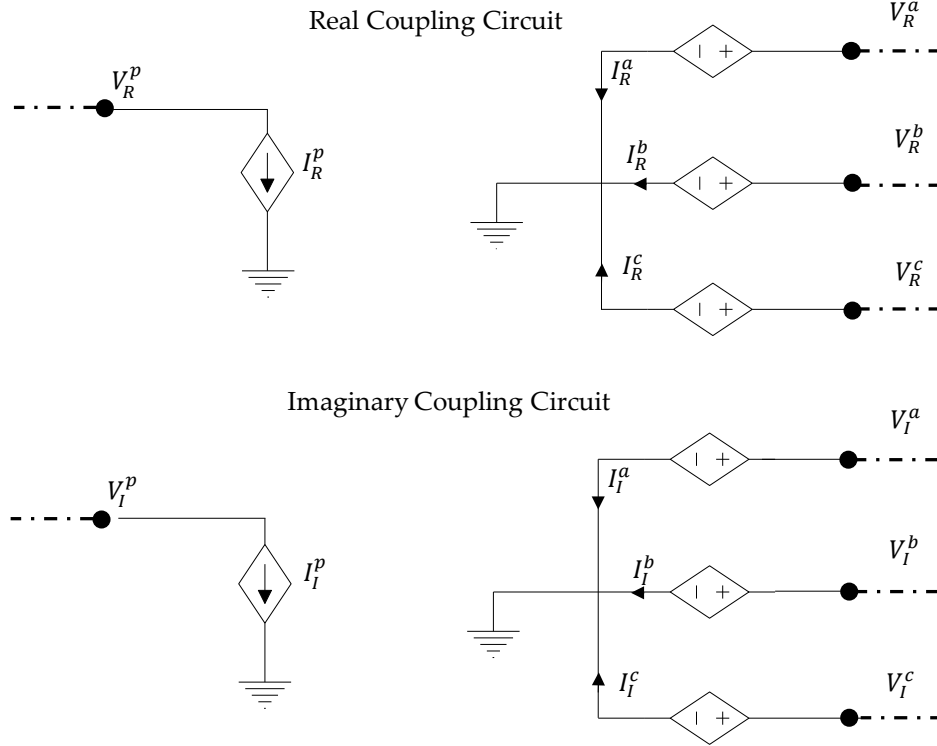


Figure 35: Coupling port for combined transmission and distribution analysis.

To derive the positive sequence currents (I_R^p , I_I^p) and three-phase voltages ($V_R^a, V_I^a, V_R^b, V_I^b, V_R^c, V_I^c$) required to model the port we make use of symmetrical components [62]. The positive sequence power flow problem for the transmission grid is assumed to have balanced operation, and therefore, the zero (I_R^z , I_I^z) and negative sequence (I_R^n , I_I^n) components of voltages and currents are ignored in the calculation of distribution grid currents consumed by the transmission grid. To calculate the transmission grid currents (I_R^p , I_I^p) from three-phase distribution grid currents, (102) is used.

$$\begin{bmatrix} I_R^z \\ I_I^z \\ I_R^p \\ I_I^p \\ I_R^n \\ I_I^n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & \alpha^2 & 0 & \alpha & 0 \\ 0 & 1 & 0 & \alpha^2 & 0 & \alpha \\ 1 & 0 & \alpha & 0 & \alpha^2 & 0 \\ 0 & 1 & 0 & \alpha & 0 & \alpha^2 \end{bmatrix}^{-1} \begin{bmatrix} I_R^a \\ I_I^a \\ I_R^b \\ I_I^b \\ I_R^c \\ I_I^c \end{bmatrix} \quad (102)$$

Similarly, the distribution end voltages as a function of positive sequence transmission POI voltages are calculated via:

$$\begin{bmatrix} V_R^a \\ V_I^a \\ V_R^b \\ V_I^b \\ V_R^c \\ V_I^c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & \alpha^2 & 0 & \alpha & 0 \\ 0 & 1 & 0 & \alpha^2 & 0 & \alpha \\ 1 & 0 & \alpha & 0 & \alpha^2 & 0 \\ 0 & 1 & 0 & \alpha & 0 & \alpha^2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ V_R^p \\ V_I^p \\ 0 \\ 0 \end{bmatrix} \quad (103)$$

Importantly, if unbalanced operation is expected at the high voltage transmission system level, then one must construct the three-phase equivalent circuit model of the transmission grid and couple it directly with the three-phase equivalent circuit model of the distribution grid at the POI. This can be done via an equivalent circuit approach via following the methodology set forth in this report and in [31]-[35]. However, the analysis of an unbalanced three-phase transmission network is beyond the scope of this project work.

We explore two approaches for combined simulation of transmission and distribution (T&D) grids in this report:

- i. Combined T&D simulation on a single machine.
- ii. Combined T&D simulation on distributed cores with parallel computing.

4.3.4 Combined T&D Simulation on a Single Core

In this approach, we first derive the equivalent circuits for the individual transmission and distribution grid models. We then couple the two using the coupling port model given in Section 4.3.3 to develop the aggregated equivalent circuit. Once we have developed the aggregated circuit for the combined T&D system, we stamp each element in the coupled network into the system matrix. This system matrix is like the one that we described for positive sequence power flow or three-phase power flow. Once the system matrix is created for the aggregated circuit, NR method is applied to find the solution of the aggregated circuit with stamps in the solution matrix being updated at each NR iteration. Figure 36 shows the coupled network for a combined transmission and distribution network.

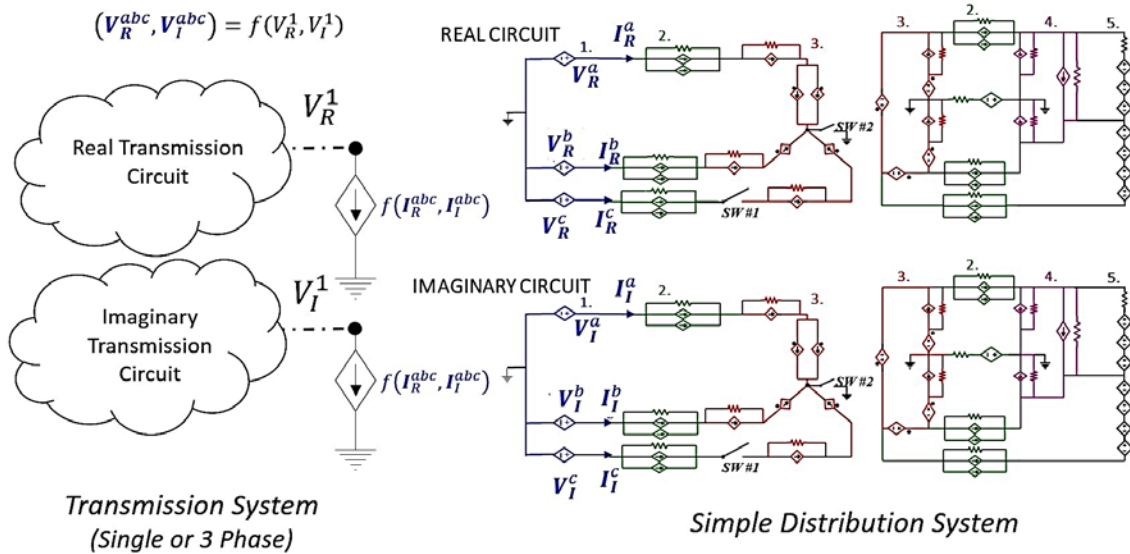


Figure 36: General framework for performing combined transmission and distribution simulation using equivalent circuit approach.

Here, we discuss results from the simulations of combined T&D networks on a single machine and demonstrate how our approach can overcome challenges faced by the existing methods.

4.3.4.1 Experiment 1

In the first experiment, the transmission grid is represented via a 9241 node PEGASE test system, which is then coupled to a distribution grid model that is further represented by a taxonomy feeder test case (R5-35.00-1) at the point of interconnection (POI). For the purposes of this experiment, the original distribution test case is modified to include distributed energy resources (DERs) in roughly 20% of the system nodes that contain electrical loads. The net capacity of DERs at each node is kept variable and is modified throughout the experiment.

The goal of this experiment is two-fold:

- i. To demonstrate that higher capacity of distribution loads can be supplied with higher penetration of DERs.
- ii. To demonstrate that more resilient grid voltages can be obtained by higher penetration of DERs during normal as well as contingency operation.

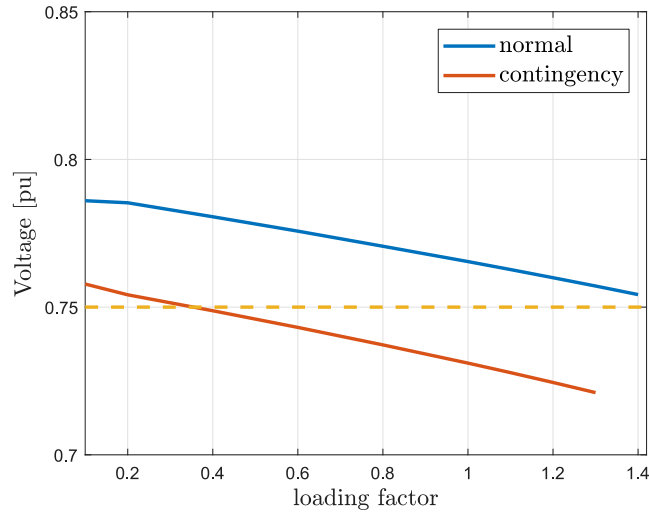


Figure 37: POI voltages under normal and contingency operation with changing distribution load.

To obtain the base maximum loading for the combined T&D system, we first develop the PV curve for the voltages at the POI by varying the loading factor of the distribution feeder, as shown in Figure 37. We repeat this analysis on the coupled system by taking a generator off-line (N-1 contingency) on the transmission grid that is in close vicinity of the POI. As seen in the Figure 37, for the base case with no DERs, the voltages after the contingency has occurred have dipped below 0.75 pu for majority of the loading factors and the likelihood of a system collapse is higher with increasing loading of the distribution feeder.

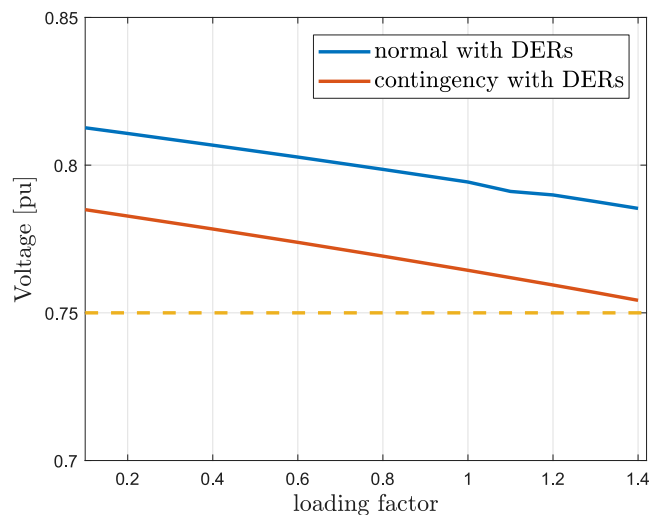


Figure 38: POI voltages under normal and contingency operation with changing distribution load and with DERs in the system.

To supply the full load in the distribution feeder such that the POI voltages remain above 0.75 pu during all loading factors, we increase the penetration of DERs in the system. We simulate the contingency and normal cases with the increased DERs and show the results in Figure 38. With the increased penetration of DERs in the system, the voltages are above 0.75 pu during normal as well as contingency scenarios for all loading factors up to 1.4x while being able to supply greater than rated load of the distribution feeder without system collapse.

4.3.4.2 Experiment 2

A similar experiment is performed on a larger more realistic test case. In this experiment, the 78k+ nodes Eastern Interconnection testcase of the U.S. transmission grid is modeled via positive sequence network. The 8000+ nodes taxonomy feeder three-phase test system is then coupled to a weak point of this transmission network where the voltages are highly sensitive to load currents. The primary goal of this experiment is to evaluate the minimum penetration of DERs required to supply the full load of the distributed grid while ensuring that the sub-station voltages at the POI remain above 0.75 pu.

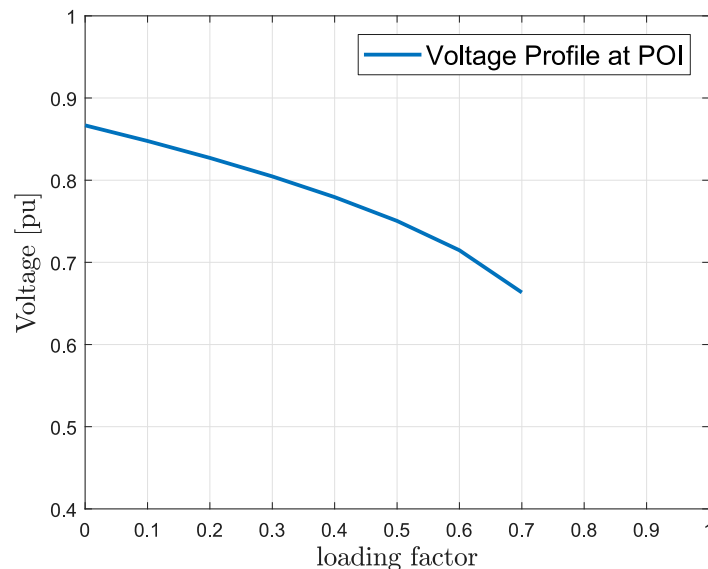


Figure 39: Voltage in pu at the point of interconnection with increasing loading factor of the distribution feeder.

To first evaluate the maximum transfer capacity at the POI prior to voltage collapse, we gradually increase the loading factor of the distribution feeder until the system collapses. As seen in Figure 39, the system can only supply a fraction of the rated capacity (0.7 loading factor) prior to voltage collapse without any penetration of DERs.

As a remedial action, the penetration of DERs in the system is increased until the transmission grid can supply the full load of the distribution system while keeping the

voltages at the POI above 0.75 pu. As in the prior experiment, the DERs in the system are added to roughly 20% of the total system nodes that contain electric loads. A scaling factor is used to increase the penetration of DERs in the simulation. Figure 40 shows that with 20% penetration of distribution generation in the distribution grid, the transmission network can supply the full load while maintaining grid voltages above 0.75 pu at the interconnection sub-station.

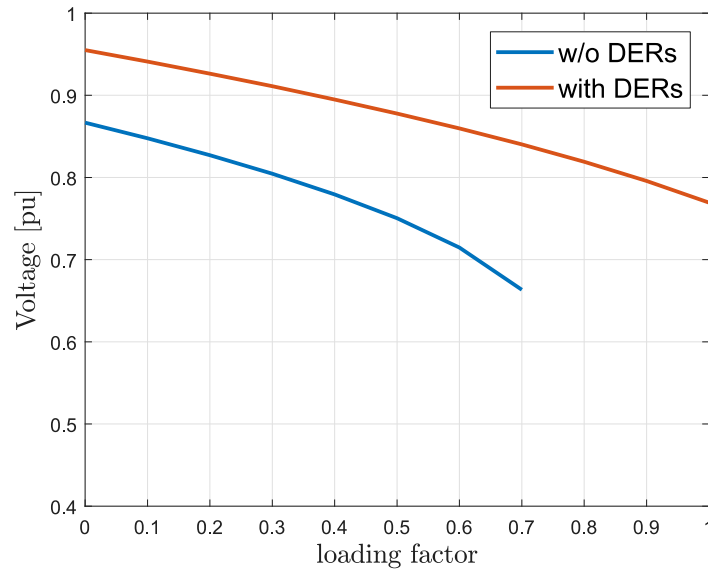


Figure 40: Voltage in pu at the point of interconnection with increasing loading factor of the distribution feeder i) with DERs and ii) without DERs.

4.3.4.3 Experiment 3

In this experiment the combined T&D framework is used to demonstrate the flow of power from the distribution network into the transmission network; i.e., reverse flow of power. This reverse flow of power is demonstrated by gradually increasing the penetration of DERs in the distribution feeder until the power flow direction is reversed. The results for this experiment are shown in Figure 41. The left vertical axis in the figure shows the active power transfer across the POI whereas the horizontal axis shows the penetration of DERs in the system as a function of its scaling factor. It is shown that as the net penetration of DERs increase in the distributed feeder, the net active power transfer across the POI decreases at first. At around >1.2 times the rated capacity of DERs, the direction of flow of power is reversed with power flowing from the distribution feeder into the transmission network. On the right vertical axis of the figure, the voltage in pu at the POI is shown as a function of the variable DERs in the distribution feeder. As expected, the net increase in DERs result in a voltage magnitude increase at the POI. Akin to prior experiments, DERs are added to roughly 20% of the distribution feeder nodes that carry electric load.

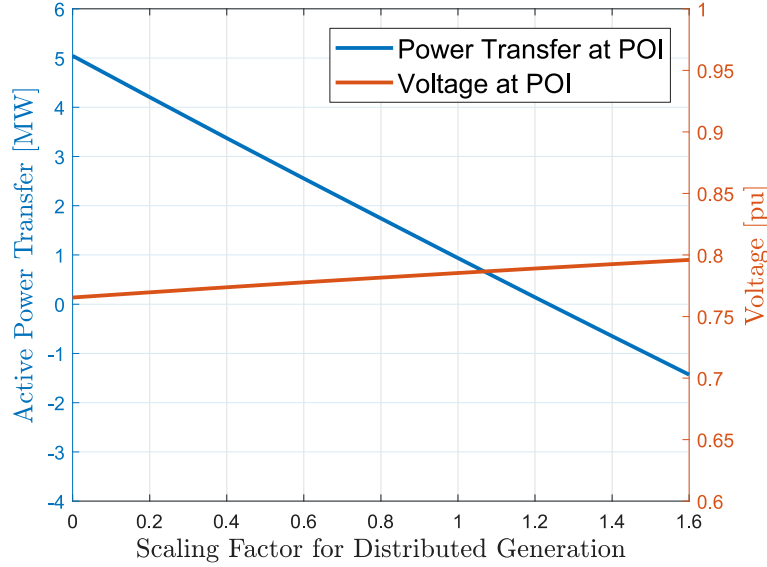


Figure 41: Reverse power flow observed during increasing DERs in the distribution feeder.

4.3.5 Combined T&D Simulation in Parallel with Distributed Cores

The experiments in the prior sub-section detail the equivalent circuit approach for combined T&D analysis on a single machine. However, while solving large combined T&D systems with hundreds of distribution networks connected to a single transmission network, the computational capacity and the system memory of a single machine may be insufficient. Therefore, beyond a certain sized integrated system, the combined T&D simulation becomes computationally impractical on a single machine due to the large size of the solution matrix. Therefore, to address this limitation, we explore the use of a parallel simulation framework with the use of distributed cores or machines. In the proposed approach, the large integrated equivalent circuit with multiple transmission and distribution networks are “torn” into multiple sub-circuits using the theory of diakoptics [78], first proposed by Kron. In the past, significant research has been carried out [79]-[81] for solving large circuits in parallel for solution matrices that have a special bordered block diagonal (BBD) structure. Interestingly, the solution matrix of the combined T&D problem due to the hierarchical nature of the coupling between the various networks is inherently in BBD form, and therefore, the developed theory for parallel simulation in circuit simulation domain can be directly applied to our problem. One must note that the primary purpose of the following discussion within this section is not to develop parallel methods for power system simulation, but rather to introduce fundamental concepts and simple examples corresponding to the proposed equivalent circuit framework that in future may garner interest and further enable the available research in the field of parallel circuit simulation to be applied directly to this problem [82], [86] and [89].

In the following sections, we discuss the “tearing” of large combined T&D system into multiple sub-circuits through domain-based decomposition [86]. We then briefly introduce the Gauss Seidel Newton (GSN) algorithm that can be used to solve a combined T&D problem in a parallel framework.

Importantly, one of the key prerequisites for a robust parallel simulation framework of a large T&D circuit is the ability to solve each individual sub-circuit robustly. In our case, this relates to solving the power flow and three-power flow equivalent circuits robustly. In a large simulation problem wherein, we may have hundreds, or even thousands of distribution networks connected to a single transmission network, it is of utmost importance that we can ensure robust convergence to the correct physical solution for each of the individual networks. Otherwise, it may cause severe bottlenecks in the overall problem convergence leading to divergence or even convergence to erroneous results. Our equivalent circuit framework with circuit simulation methods can ensure robust convergence for both the power flow and three power flow circuits, thereby extending the same robust properties to the parallel simulation framework.

4.3.5.1 Background

There has been extensive research towards the use of parallel simulation techniques for obtaining the DC and transient solution of very large integrated circuits [79]-[86]. The theory of diakoptics [78] and bordered block diagonal matrices [80] are integral to these solution methodologies and are developed within that work. We briefly discuss these key concepts in following sub-sections and demonstrate how they can be extended to solve combined T&D problems in the equivalent circuit framework.

4.3.5.2 Diakoptics

Diakoptics, or the “methods for tearing” [78], involves taking a large problem and dividing it into the set of sub problems, which can then be solved independently prior to being coupled together again to provide an exact solution for the problem. The aim of this technique is to tear the network either through domain-based decomposition [86] prior to the construction of the solution matrix or through the direct partitioning of the solution matrix with no prior domain knowledge. In the combined T&D problem, the distribution feeders are known to be weakly coupled to the transmission network often at a single point of interconnection. This allows for the application of domain-based decomposition to “tear” the integrated T&D network into a set of sub-networks with lines at POIs being the cut-set branches as shown in Figure 43. To numerically demonstrate the following, consider an aggregated T&D network with the following function form:

$$\mathcal{F}(\mathbf{V}_R, \mathbf{V}_I) = 0 \quad (104)$$

This large T&D network is torn into m independent sub-circuits that consist of the internal variables $(\mathbf{V}_R^{int}, \mathbf{V}_I^{int})$ that are only functions of circuit elements within the sub-

circuit and the external variables ($\mathbf{V}_R^{ext}, \mathbf{V}_I^{ext}$) that are functions of circuit elements in the other sub-circuits [79]. The decomposed sub-circuits have the following function form:

$$\mathcal{F}_{int}(\mathbf{V}_R^{int}, \mathbf{V}_I^{int}, \mathbf{V}_R^{ext}, \mathbf{V}_I^{ext}) = 0 \quad (105)$$

$$\mathcal{F}_{ext}(\mathbf{V}_R^1, \mathbf{V}_I^1, \dots, \mathbf{V}_R^m, \mathbf{V}_I^m, \mathbf{V}_R^{ext}, \mathbf{V}_I^{ext}) = 0 \quad (106)$$

for $int = 1, \dots, m$, and $\{\mathbf{V}_R^{int}, \mathbf{V}_I^{int}\} \in \mathbb{R}^{n_i}$ are the internal nodal voltages of sub-circuits, and $\{\mathbf{V}_R^{ext}, \mathbf{V}_I^{ext}\} \in \mathbb{R}^{n_e}$ are the external nodal voltages.

4.3.6 Bordered Block Diagonal (BBD) Matrix Structure

In many fields of engineering and science, block bordered structured matrix-based problems arise. In the circuit simulation field, BBD matrix structure is common for representing the system matrix for the VLSI circuits, wherein partitioning (through node tearing or branch tearing) of the circuit was beneficial for parallel analyses. The ability to represent these large circuit's solution matrices in the BBD form enables the use of vastly researched parallel sparse solver techniques [86] to obtain the system solution. The hierarchically structured combined T&D network, wherein a limited number of transmission networks supply bulk power to numerous local distribution feeders, is inherently representable in BBD form. For instance, consider Figure 42 in which three distribution networks are connected to a large transmission network. There exists a natural weak coupling between the different networks in the figure shown via the flow of current.

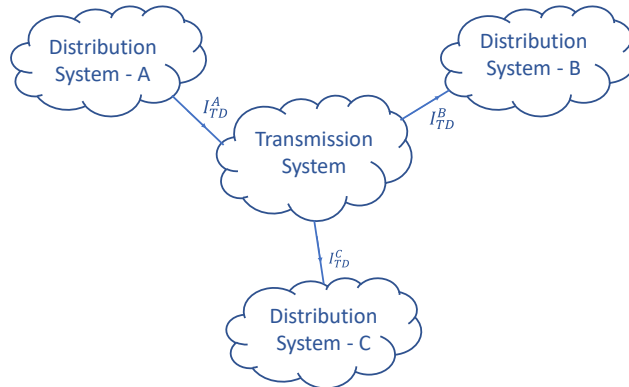


Figure 42: Weakly coupled transmission and distribution network.

This integrated network can be divided into a set of sub-systems ($\mathcal{S}$) by the branch tearing technique at the coupling points between the transmission and distribution network, as shown in Figure 43.

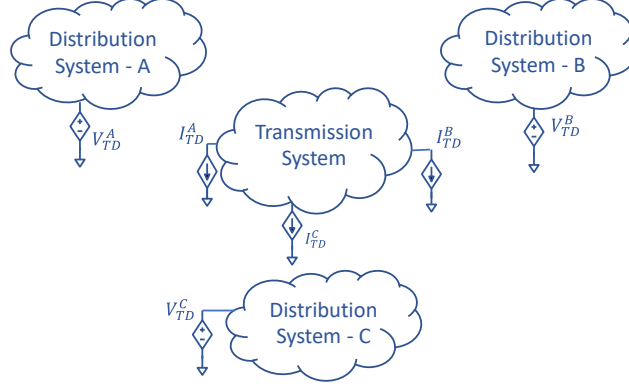


Figure 43: “Torn” transmission and distribution sub-systems.

T				tt'
	D_A			td_A
		D_B		td_B
			D_C	td_C
TT'	TD_A	TD_B	TD_C	

Figure 44: Bordered Block Diagonal structure for combined transmission and distribution system.

The decomposed network is further representable in BBD form, as shown in Figure 44. The block diagonal terms in the matrix (T, D_A, D_B, D_C) represent the system Jacobian elements for the decomposed sub-circuits $(\mathcal{S} = \{\mathcal{F}_T, \mathcal{F}_{D_A}, \mathcal{F}_{D_B}, \mathcal{F}_{D_C}\})$ that are functions of sub-circuit's internal parameters $\{\mathbf{V}_R^{int}, \mathbf{V}_I^{int}\}$ whereas off-diagonal terms in the vertical right of the matrix i.e. (tt', td_a, td_b, td_c) are system Jacobian elements that are functions of sub-circuit's circuit external variables $\{\mathbf{V}_R^{ext}, \mathbf{V}_I^{ext}\}$. Remaining elements in the bottom of the matrix represent the Jacobian elements of the coupling sub-circuit (representing the port circuit equations) with respect to sub-circuits internal and external variables. Mathematically, these elements are given by:

$$T = \left\{ \frac{\partial \mathcal{F}_T}{\partial \mathbf{V}_R^{i,int}}, \frac{\partial \mathcal{F}_T}{\partial \mathbf{V}_I^{i,int}} \right\}, i = 1, \dots, \dim(T) \quad (107)$$

$$tt' = \left\{ \frac{\partial \mathcal{F}_T}{\partial \mathbf{V}_R^{i,ext}}, \frac{\partial \mathcal{F}_T}{\partial \mathbf{V}_I^{i,ext}} \right\}, i = 1, \dots, \dim(T) \quad (108)$$

$$TT' = \left\{ \frac{\partial \mathcal{F}_C}{\partial \mathbf{V}_R^{i,ext}}, \frac{\partial \mathcal{F}_C}{\partial \mathbf{V}_I^{i,ext}} \right\}, i = 1, \dots, \dim(Y_C) \quad (109)$$

$$D_x = \left\{ \frac{\partial \mathcal{F}_{D_x}}{\partial \mathbf{V}_R^{int}}, \frac{\partial \mathcal{F}_{D_x}}{\partial \mathbf{V}_I^{int}} \right\}, i = 1, \dots, \dim(D_x) \text{ \& } x = \{A, B, C\} \quad (110)$$

$$td_x = \left\{ \frac{\partial \mathcal{F}_{D_x}}{\partial \mathbf{V}_R^{ext}}, \frac{\partial \mathcal{F}_{D_x}}{\partial \mathbf{V}_I^{ext}} \right\}, i = 1, \dots, \dim(D_x) \text{ \& } x = \{A, B, C\} \quad (111)$$

$$TD_x = \left\{ \frac{\partial \mathcal{F}_C}{\partial \mathbf{V}_R^{i,ext}}, \frac{\partial \mathcal{F}_C}{\partial \mathbf{V}_I^{i,ext}} \right\}, i = 1, \dots, \dim(Y_C) \text{ \& } x = \{A, B, C\} \quad (112)$$

where set $\{A, B, C\}$ represents different sub-circuits of the different phases of distribution system. In the following section, we discuss Gauss the Seidel Newton (GSN) method, that we use to solve the combined T&D parallel problem that is representable in this BBD form.

4.3.6.1 Gauss-Seidel-Newton Approach

We make use of the Gauss-Seidel-Newton (GSN) method [83], [88] to solve the set of sub-systems given by decomposed combined T&D sub-circuits. The subsystems are chosen such that the number of internal nodes ($\mathbf{x}^{int}$) for each sub system are far less than the number of external coupling nodes ($\mathbf{x}^{ext}$). The GSN algorithm is a two-step algorithm. Within the inner loop, the set of independent sub-systems ($\mathcal{S}$) are solved in parallel using block NR algorithm until convergence or for a limited number of iterations. In this inner loop, the external coupling variables ($\mathbf{x}^{ext}$) are kept constant for each sub-circuit, whereas the internal variables ($\mathbf{x}^{int}$) are solved for iteratively. In the outer loop, the external coupling variables from each sub-system are distributed to other sub-systems via a Gauss step, and the inner loop of NRs are performed again. This iterative algorithm is then repeated until the error difference of external coupling variables communicated between the consecutive outer loops (epochs) are within a certain tolerance.

While applying GSN to combined T&D simulation problem, solving of individual sub-systems within the inner loop equates to running independent instances of power flow and three-phase power flow in parallel. One necessary condition for convergence of the GSN algorithm is that each of these independent power flows and three-phase power flows for the individual sub-systems that are solved at every Gauss step produce a correct solution; absence of which could result in divergence or breakdown of the complete algorithm. We have previously demonstrated in earlier sections that the use of circuit simulation methods along with other adapted circuit theoretic methods can ensure convergence for any feasible or infeasible power flow or three-phase power flow circuit, which makes these methods well suited for the proposed GSN algorithm. Algorithm 4-4 details the implementation of GSN algorithm for combined T&D algorithm.

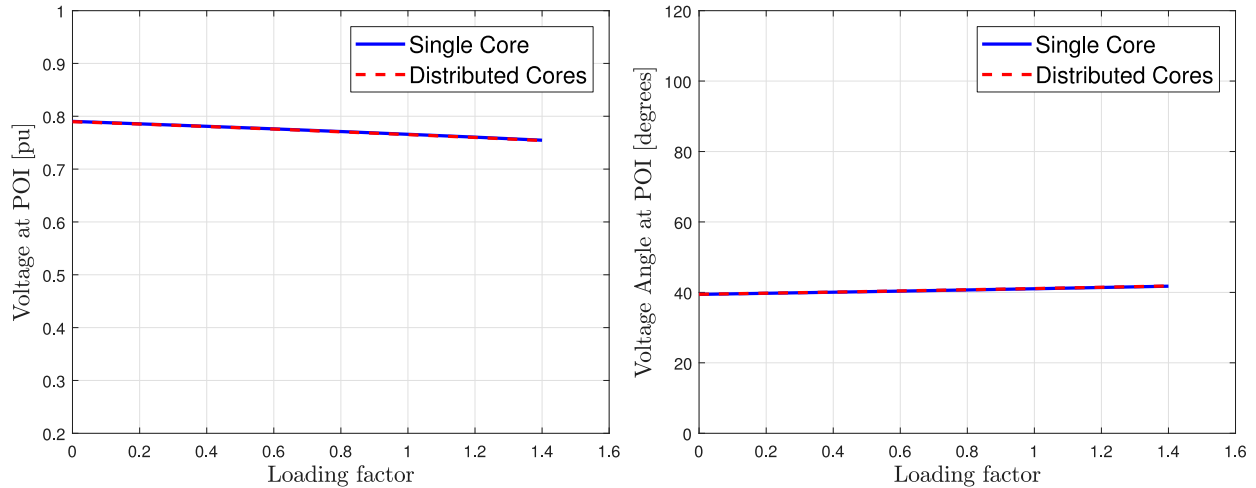


Figure 45: Comparison of combined T&D simulation algorithms: i) Single machine setup using NR (in blue), ii) Parallel simulation on distributed cores using GSN (in red).

4.3.6.3 Combined Transmission and Distribution Analysis on a Large System

In order to simulate a very large combined T&D system, 50 distribution feeders, each representing roughly 8000 nodes, were coupled to a large realistic transmission network at different locations. The network representing the Eastern Interconnection of the U.S. with roughly 85k+ nodes is used to model the transmission network, and the openly available taxonomy feeder test cases [84] are used to model the 50 distribution networks. This problem represents a solution matrix size of roughly 3 million rank with a total of ~3x4,00,000 distribution nodes and ~85000 transmission nodes. We simulated this case using GSN until completion.

The complete simulation took 11 Gauss steps wherein within each Gauss step, NR simulations run in parallel for each of the block sub-circuits took around 5-10 iterations. In the final solution, the POI voltages were all found to be within the acceptable range of 0.8-1.2 pu and the complete simulation took less than a couple of minutes to converge with Tx-stepping method enabled. The Figure 46 represents the evolution of the sub-station voltages at the POI during the Gauss-step in the outer loop of the parallel combined T&D simulation.

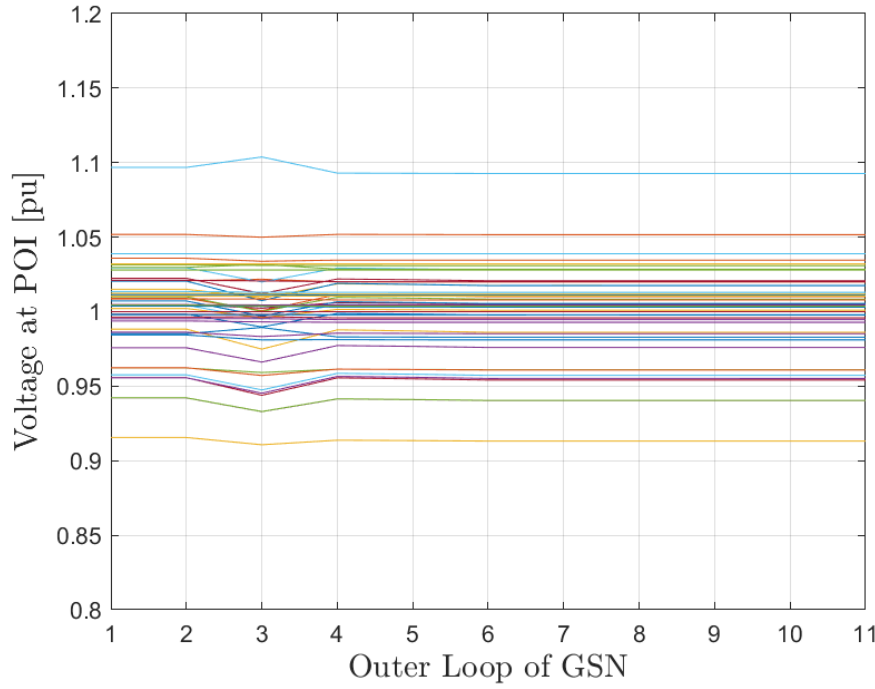


Figure 46: Voltages at the POI in the outer loop of GSN.

4.3.7 Notes on Convergence

Suppose that the system of non-linear equations that represents the large combined interconnected transmission and distribution network is given by:

$$JV = I \quad (113)$$

where matrix J has the form given in Figure 44. To further explore the convergence properties, this matrix J can be split into two components given by:

$$J = M - N \quad (114)$$

In general, for the Gauss-Seidal-Newton (GSN) algorithm to guarantee convergence for the decomposed matrix J the spectral radius of the iteration matrix ($\rho(M^{-1}N)$) needs to be less than unity. However, a less strict condition that requires the solution matrix to be point-wise strictly diagonal dominant is often sufficient i.e.

$$\sum_{i \neq j}^n |a_{ij}| \leq |a_{ii}|, \text{ for all } i \quad (115)$$

where a_{ij} is a value in the matrix for i^{th} row and j^{th} column.

Since the solution matrix J is representable in BBDF form, a much milder condition can guarantee convergence for the partitioned system. If the combined system is represented as an aggregated equivalent circuit and is partitioned into sub-circuits at multiple “tearing” nodes, then the existence of a capacitance at a sub-set of these “tearing” nodes with a large enough value can guarantee convergence for the partitioned system via GSN algorithm [89]. This is a much milder sufficient condition for convergence when

compared against the strict diagonal dominance condition for a general matrix that requires a large value capacitor from each node in the system to ground. However, modifications (adding a high value capacitance from the “tearing” node to ground) such as these are often unwarranted as they change the inherent structure of the problem, and therefore, other convergence techniques should be explored.

One such method is presented in [90] and ensures convergence for the power flow network-based problems via GSN by partitioning the matrix into BBDF such that the spectral radius of the iteration matrix corresponding to the partitioned system is ensured to be less than one. The work in [90] partitions the solution matrix J into M and N such that $J = M - N$, where M is a block diagonal matrix capturing the interactions between the internal variables of each block sub-circuit and N is the off-diagonal matrix that captures the communication between the variables of other sub-circuits. To ensure convergence by GSN, the method introduces a diagonal matrix $\bar{E}$, such that the matrices M and N are modified as follows:

$$M = D + \alpha \bar{E} \quad (116)$$

$$N = \alpha \bar{E} - E \quad (117)$$

where, $J = D + E$. It is shown in [90] that by choosing the value of $\alpha = \frac{1}{2}$, the algorithm can ensure convergence for the partitioned system.

Furthermore, it should be noted that other methods such as the use of distributed Schur’s complement [93] can also be used to extract the exact solution of the linearized matrix J_l for $J_l V = I_L$ at each step of NR given that the linearized matrix J_l is in BBDF.

5.0 RESULTS AND DISCUSSIONS

With the theoretical background of an Equivalent Circuit Formulation based solver now established, this report now details of the implementation of an ECF based solver.

This implementation was designed with the following goals in mind:

- Cleanly designed and written code for reusability and extendibility
- Efficient, high-performing code with a focus on readability while allowing code optimizations in the inner loops
- Parallelizable code to support the Monte Carlo analyses that form the primary objective of this portion of the project

5.1 SUGAR C++ (an ECF based solver)

The readability, reusability, and most importantly, efficiency, of a software such as SUGAR depends heavily on its architecture and design. The better a software is broken up into operative sub-components that interact via well-defined interfaces, the better the software will operate. Additionally, continuous future development is only possible when these components and their interfaces are logical, easily understood, and well documented. Furthermore, it is not necessary to reinvent the wheel. As many components as possible should be part of high-performing, widely used, and well tested third-party libraries.

5.1.1 Concepts and Abstractions

Before describing the software design in depth, we discuss abstractions and concepts that comprise an ECF based solver from an algorithm perspective. This enables us to better understand the required software abstractions, their expected performance, and their possible-restrictions.

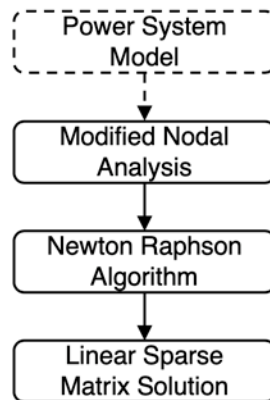


Figure 47: Concepts hierarchy of an ECF based solver.

Figure 47 provides an overview of algorithms that are employed to solve a PF problem using the ECF.

The lowest level algorithm within this hierarchy is a linear sparse matrix solution algorithm. Linear sparse matrix solvers can be classified into direct and iterative solvers.

Direct sparse matrix solvers are a well-developed field [97]. Sparse matrix solution methods include the Cholesky-, QR- and, LU-factorization algorithms. LU factorization is the oldest solution method and represents the default algorithm for general square matrices [98]. LU factorization algorithms can be categorized into left-looking, right-looking, and multi-frontal algorithms. The challenge of any LU factorization algorithm is to effectively determine permutations p and q of a matrix $A^{n \times n}$ such that the factorization $A(p, q)$ is reasonably stable and L and U are close to optimally sparse [97]. This requires an algorithm to find a tradeoff between maximized matrix conditioning and minimal matrix fill-ins.

It is difficult to determine exact time and memory requirements as well as scaling properties of these algorithms as they highly depend on a problem's properties, such as matrix-sparsity, matrix-structure, etc. However, [98] describes the performance of right-looking LU factorization algorithms to take $O(n + |A| + f)$ time, where n is the matrix size, $|A|$ is the number of non-zeros in A and f is the number of floating point operations. With a few exceptions, this reduces to $O(f)$ [98]. To determine memory complexity of these algorithms, we will take an empirical approach and study memory requirements of the KLU [99] algorithm of different problems in the results section.

Both of the evaluated LU factorization algorithms (UMFPACK [100] and the KLU [99]) are divided into two stages: a symbolic factorization stage that only requires the matrix structure as input, and a numeric factorization that finds numerical results to a given problem. This enables us to reuse symbolic factorizations during the next-higher algorithm in this hierarchy (the Newton-Raphson algorithm), as NR solves multiple sparse matrix problems that only differ numerically.

The Newton-Raphson algorithm is a numerical solution algorithm for nonlinear problems. It requires an initial guess for a problem's solution and employs a gradient based method to iteratively converge to a solution of a nonlinear system of equations. Barring some exceptions, a good initial guess leads to quadratic convergence of this algorithm.

To solve a general system of nonlinear equations the NR algorithm requires partial differentials of all dependent system variables to each other (the Jacobian). ECF based solvers hierarchically build a problem's Jacobian in terms of equivalent circuit analogies using equivalent circuit-based models and MNA. Therein, each model symbolically stores equations to determine its corresponding Jacobian entries for each NR iteration. This enables a modularized and efficient algorithm design.

The highest-level algorithms within an ECF solver are limiting or homotopy methods that aim to improve convergence of the nonlinear solver. Herein, limiting methods require a solution vector data-pass for each NR-iteration. Hence, their implementation is performance critical, making them perfect candidates for early code optimizations. Homotopy methods on the other hand, alter a system after an NR loop finalizes. Their implementation is less critical to inner-loop solver performance, however, designing

these algorithms to require minimal steps is critical to the overall performance of an ECF solver.

5.1.2 Module and Interface Design

The SUGAR C++ implementation is split into three core components: A nonlinear equivalent circuit solver infrastructure, a set of solver control algorithms, and a set of data structures. These data structures are the parsed power flow models, their ECF representations, and structures that assemble the element models to models of power systems. Additionally, a set of utility functions and packages fulfill I/O tasks or tasks that are commonly used in many parts of the code.

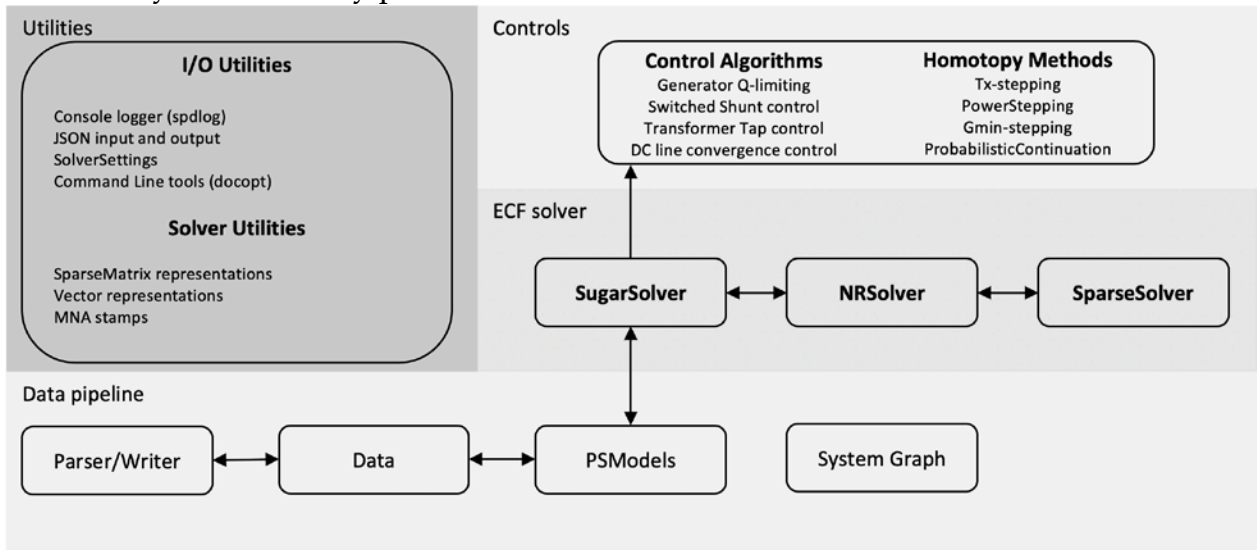


Figure 48: Global design overview of SUGAR C++ solver.

Figure 48 gives an overview of the solver’s components. The “ECF solver” component comprises of three objects: the *SparseSolver* that serves as an interface to sparse matrix solver libraries and is utilized to solve a single linearized equivalent circuit problem; the *NRSolver* that solves one Newton-Raphson loop at a time and is controlled by the *SugarSolver*, which in turn enables the integration of control algorithms via a single well-defined interface; the *SugarSolver* that relays the Power System model (*PSModels*) to the *NRSolver* and keeps track of solutions.

The data pipeline classes enable Power System models to be instantiated from and written to different file types. Parsers for PSSE .raw file formats or MatPower files exist. The *PowerFlowData* class acts as a container for this input data. From a *PowerFlowData* object a *PSModels* object is instantiated. The *PSModels* class encapsulates all ECF models and provides interfaces for solver and control algorithms. All of this is enabled by a set of utilities that serve either I/O purposes, such as a library to log solver states and progress to the console or a file, a command line input library, or a json parser to that

enables solver inputs via a json based input file. Other utilities include sparse matrix and vector representation classes and helper functions that predefine MNA stamps.

5.1.3 ECF solver

The “ECF solver” is the core of this implementation. Here, we focus on a clean optimized implementation of the solver classes that define interfaces for control but do not implement controls themselves. The ECF solver consists of three stages.

5.1.3.1 SparseSolver

The innermost stage is the *SparseSolver* class that defines an interface for sparse matrix solver libraries. Its purpose is straight forward. It receives a sparse matrix problem defined by a sparse system matrix and a right-hand side vector, relays this problem to a sparse matrix solution library, and returns the solution vector to the next stage, the *NRSolver*. The sparse solver utilizes either the UMFPack [100] or KLU [99] library to solve sparse matrix problems. Both are part of the *SuiteSparse* package.

5.1.3.2 Matrix and Vector Representations

Sparse matrix solvers require specific matrix formats that commonly compress storage. Many different formats with specific advantages and disadvantages exist. Generally, more complex compression schemes make it more difficult to create and modify a matrix. This is why our implementation splits creating a system matrix into two stages with two different matrix formats.

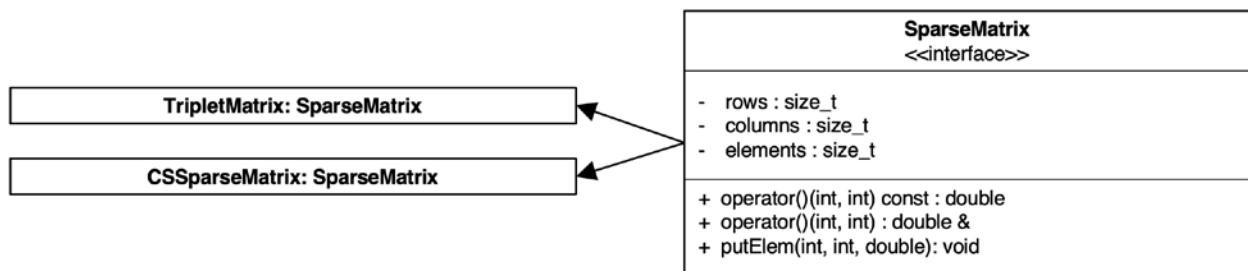


Figure 49: The two *SparseMatrix* classes share a common interface within SUGAR C++.

Figure 49 shows the common interface of the two available sparse matrix classes of the SUGAR C++ implementation. Herein, the *TripletMatrix* stores each matrix entry as a triplet of row index, column index, and matrix entry. Its underlying data structure is a C++ map. This enables stamping while the matrix structure is still unknown. The *CSSparseMatrix* structure stores a sparse matrix in either compressed row or compressed column format. Both formats are used as inputs to the sparse matrix packages. However, to handle them efficiently, the matrix structure needs to be known in advance. Looking at the inner loop from a data flow perspective, clarifies how both matrix types are used.

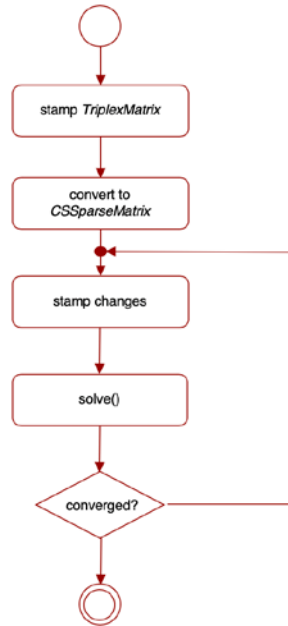


Figure 50: Sparse matrix conversion during Newton-Raphson loop.

Figure 48 shows the Newton-Raphson loop from a matrix manipulation perspective. First, a matrix is filled with non-changing entries before entering the loop, where the complete structure of the matrix is still unknown. The trick to define the full system matrix structure is to add non-changing entries as zero elements and later fill them during the loop. After the full matrix structure is defined and the slow and storage heavy *TripletMatrix* is converted to a compressed representation. The corresponding *CSSparseMatrix* that is further filled with nonlinear elements and passed to the solver library. The initially converted matrix is stored, so that subsequent iterations are able to reuse only efficiently stored matrices.

It is worth noting that a similar interface exists for vector representations that include a *PairArray*, which is the pendant to the *TripletMatrix*, and a *DenseArray* that is used after conversion. However, while this separation simplifies our algorithm, splitting vector representations is far less critical. Vectors of *SparseMatrix* problems are generally dense and much smaller in size than their corresponding matrices. For simplicity of the following discussion, we will refer to sparse matrices and vectors without mentioning their exact representation.

5.1.3.3 NRSolver Class

We now look at the nonlinear solver from an algorithm perspective. The *NRSolver* class implements the inner NR-loop, which solves a nonlinear ECF problem using the Newton-Raphson algorithm. The *NRSolver* implements the innermost loop. Hence, it is crucial in defining the overall performance the solver. We designed this inner loop to be insulated

from most control algorithms. This enables an efficient, clean implementation that optimizes performance.

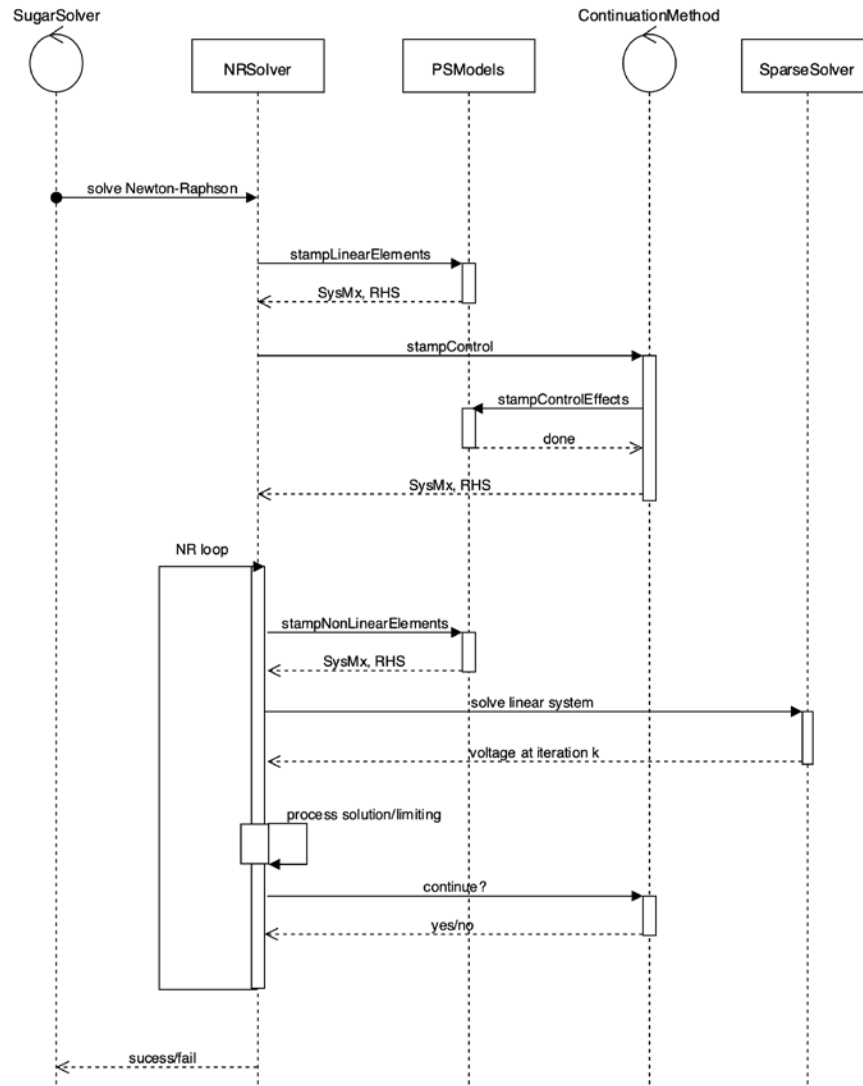


Figure 51: Activity diagram of the *NRSolver*'s inner loop implementation.

Figure 51 shows the activity diagram of the *NRSolver*'s inner loop implementation. The algorithm is started by a function call from the *SugarSolver*. It stamps the full system matrix in three distinct stages. First, the linear components are stamped and stored for reuse. Next, the control algorithm dependent matrix entries are stamped. Further, both sub matrices are composed to a non-changing matrix that remains unchanged within a NR loop. Hence, this matrix is reused every time the loop reaches its top. After the innermost loop is entered, all nonlinear elements are stamped on top of the composed non-changing matrix, defining the full matrix problem. Note that a control algorithm's influence on this loop so far is limited to the *stampControl()* method.

The full matrix problem is then solved via a function call to the *SparseSolver* that, after a successful solve, returns a solution vector. This solution vector is further postprocessed.

Postprocessing algorithms require the current and the previous solution vectors. They include limiting and damping methods and the calculation of the difference between the solutions to evaluate convergence.

The efficiency of the limiting, damping, and difference algorithms is vitally important to the total solver performance as they require looping through data inside the innermost loop. Hence, all algorithms are implemented inside the *NRSolver*. This enables their implementation within a single data pass, which keeps them as high-performing as possible. The postprocessing algorithms are at the bottom of the inner loop. Now, three reasons to exit the NR loop exist:

- The difference between solutions is smaller than the tolerance: the problem converged.
- The maximum number of iterations is reached.
- The system diverged and the outside control algorithm (via a function of the *ContinuationMethod* interface) decides to stop.

Finally, the *NRSolver* returns a success/fail flag and the final number of iterations. In case of convergence, the solution vector is further extracted and stored by the *SugarSolver* class.

Importantly, a single NR loop is commonly not sufficient to solve a full PF problem. Usually, control actions that are modeled in outer loops (e.g. Generator Q-limiting) are necessary. To improve the performance of the solver we aim to reuse as much data as possible. Therefore, when the *SugarSolver* invokes the *NRSolver* loop again, the initial linear matrix that was created during the prior loop is reused and the inner loop starts with stamping the control dependent matrix parts.

5.1.3.4 **SugarSolver**

The *SugarSolver* defines how control algorithms interact with the *NRSolver* by implementing an outer control loop structure that includes call-back functions for generic control algorithms. This design enables insulation of potentially complex control implementations from the solver that using this design remains clean, readable, and efficient. Additionally, this design choice defines a rigid interface for control algorithms, which greatly simplifies their implementation.

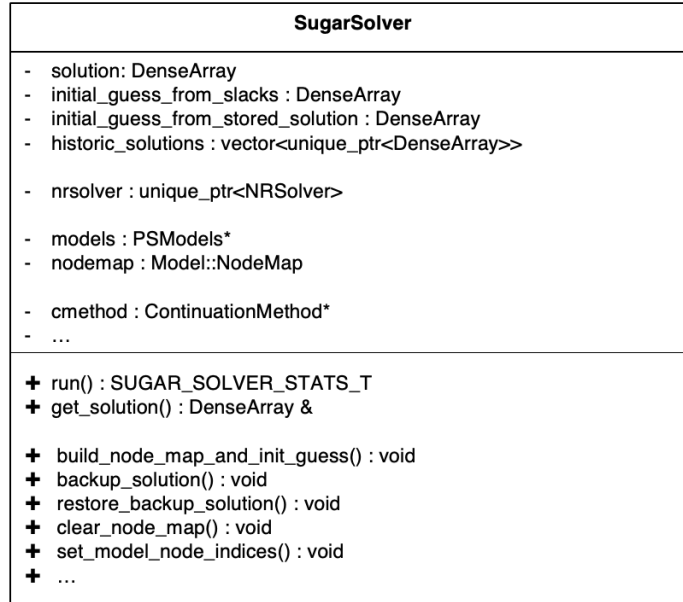


Figure 52: UML class diagram of the *SugarSolver*.

Figure 52 shows a class diagram of the *SugarSolver* including its most important components. The *SugarSolver* controls the execution of inner Newton-Raphson loops. It initializes these loops with initial guesses. Hence it stores possible initial guesses as member variables and keeps track of sub-problem solutions that can further be used for future initial guesses of ensuing inner loops.

Additionally, the *SugarSolver* references a *PSModels* object that contains all ECF models for stamping. As discussed above, prior to stamping, the elements have to be assigned MNA matrix row indices. The *SugarSolver* initializes MNA node indices within the models via the *set_model_node_indices()* function and further stores this node assignment in a *NodeMap* class that maps each PF node to corresponding matrix rows. A final crucial member of the *SugarSolver* class is a pointer to a control algorithm that is accessed via a *ContinuationMethod* interface.

After receiving a *PSModels* structure, setting its MNA node assignments, and receiving a control algorithm, a simulation is started via the *SugarSolver*'s *run()* function. When the simulation finalizes, the *run()* function returns a structure of statistics of the simulation run including number of total Newton-Raphson iterations, number of control loop iterations, and success or failure of the simulation. Finally, the simulation's result can be obtained by calling the *get_solution()* function.

to determine and fulfill its function. These are the solver state including the previous solutions, and the models that are altered during control actions. After initialization, the control loop is started. A first control action is possible during the *prepare()* call. Subsequently, the inner simulation loop is invoked. After its completion, the controller is notified of the inner-loop's success or failure. Additionally, the stored solutions are updated by the *SugarSolver*: In the case of a successful inner loop, the new solution is stored as possible initial guess for the future. If the inner loop does not converge, the *SugarSolver* notifies the controller of the failure and restores the last obtained solution as the initial guess for the next inner loop. The outer control loop finalizes when the control algorithm concludes. This is determined by a call to the control algorithm's *finished()* function.

It is important to highlight that the *SugarSolver* does not rely on the control algorithm to store solutions and use them as future initial guesses. If this was not the case, control algorithms would have to implement this on its own. This would lead to multiple copies of the same code, which is a common source of software bugs. Importantly, this design choice restricts control algorithms to take only a single step back (to the previous solution) in the case of non-convergence, which in practice has proven not to be a limitation.

5.1.3.5 Outer Loop Controls

Next we discuss the implementation of different outer-loop control algorithms. The interface of a control algorithm is defined by the *ContinuationMethod* class.

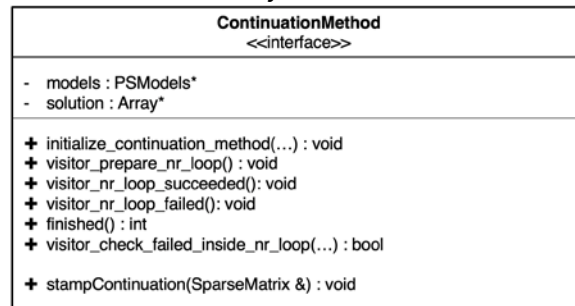


Figure 54: Class diagram of the ContinuationMethod control interface.

Figure 54 shows a class diagram of this interface. Each control algorithm implements a specialization of the *ContinuationMethod*'s member functions. These functions are then called by the *SugarSolver* as visitors within the outer control loop of **Error! Reference source not found.** This structure enables the implementation of two kinds of algorithm: Homotopy or continuation methods, and outer loop controls, such as generator Q-limiting, transformer tap adjustments, or switched shunt adjustments.

Homotopy Methods implemented as ContinuationMethod Classes

First, we discuss the implementation of homotopy methods. Currently implemented methods within SUGAR C++ are the Tx-stepping method and the probabilistic

continuation method that was designed specifically for Monte Carlo Simulations. The Tx-stepping method is implemented to include adaptive step sizes, which greatly speeds up convergence. Its parameters are tuned to solve representations of real Eastern Interconnection PF cases and contingencies thereof as efficiently and robustly as possible, without a good initial guess.

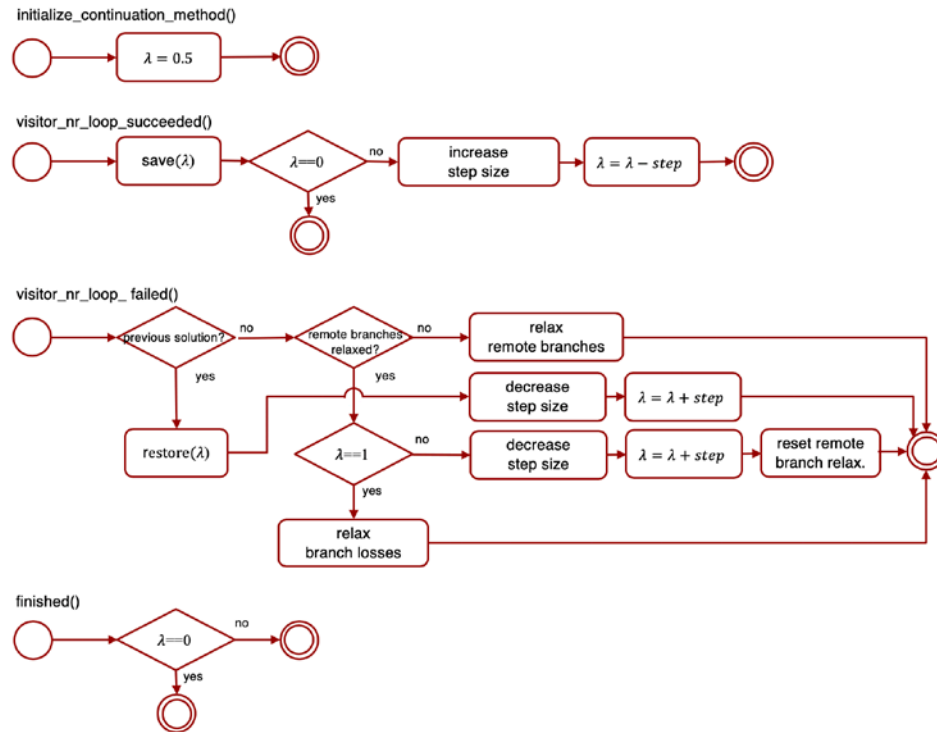


Figure 55: Core algorithm of the adaptive Tx-stepping implementation.

Figure 55 shows how the sub-algorithms of the Tx-stepping method are implemented as *ContinuationMethod* visitor functions. Here, λ is the Tx-stepping homotopy factor. It ranges from 1 to 0, where 1 represents a completely relaxed system and 0 represents the original problem. The adaptive step size implementation of Tx-stepping initializes λ to be 0.5, which represents a 50% relaxed state. Importantly, this is after a solution of the original problem ($\lambda = 0$) was attempted.

Next, as defined by the outer control loop algorithm, a solution attempt with λ at 0.5 is started. In case we find a solution to this 50% relaxed problem, the *visitor_nr_loop_succeeded()* function is invoked. It first stores the value of λ that now corresponds to a solved system with a solution that can be used as initial guess for the next step. As mentioned before, only the homotopy factor λ needs to be stored. The *SugarSolver* class automatically stores the solution vector during the outer control loop algorithm. Further, the *visitor_nr_loop_succeeded()* function increases λ 's step size by a factor that was optimized for solution speed and then increases λ by the updated step size.

In the case of non-convergence of the inner loop, the *visitor_nr_loop_failed()* function is called. Here, a more complex set of possible control actions is implemented. If a prior solution exists, λ is restored to the homotopy factor of this solution. Furthermore, the step size of λ is decreased and λ is updated to take a smaller step in the next solution attempt.

If no prior solution exists, other Tx-stepping parameters are changed in order to find a first solution as quickly as possible. Initially, the relaxation (admittance) of the auxiliary remote-controlled generator lines is adapted. If this fails, λ is scaled towards a more relaxed system with decreasing step size. At each of the steps of λ the auxiliary line relaxation is reset to its original, causing the relaxation to be repeated for each step of λ . When λ is 1, the system is completely relaxed in terms of the homotopy factor. Then, the final parameters that can be adapted to find an initial solution are the homotopy admittances of each transmission line and transformer in the system. The number of possible parameter-combinations for initial solution attempts provides a sufficient search space that almost guarantees finding a relaxation that leads to an initial solution. However, for most problems an initial solution is found very quickly without requiring parameter changes other than the changes of λ .

Notably, the structure of the outer control loop with its predefined visitor-functions simplifies this implementation of Tx-stepping to simple sequences of logical steps. Similarly, other continuation methods are implemented.

Device Controls implemented as Continuation Methods

Outer loop device controls, such as the generator Q-limiting algorithm, controls of transformer taps, and controls of discretely switched shunts, are implemented using the *ContinuationMethod* interface as well. However, it should not go unmentioned that while these outer discrete control loops are a de-facto standard in current commercial PF software, they are problematic. Modeling discrete discontinuities in this way can cause oscillations and lead to non-convergence of a solver.

control example: Generator Q-limiting

We will discuss the generator Q-limiting algorithm as an example of these outer discrete control algorithms. Other algorithms of that kind are implemented similarly enough that explaining one of them in detail will suffice. The generator Q-limiting algorithm, in addition to implementing the *ContinuationMethod* interface to interact with the *SugarSolver*, has access to the *PSModels* object including all generators of the system.

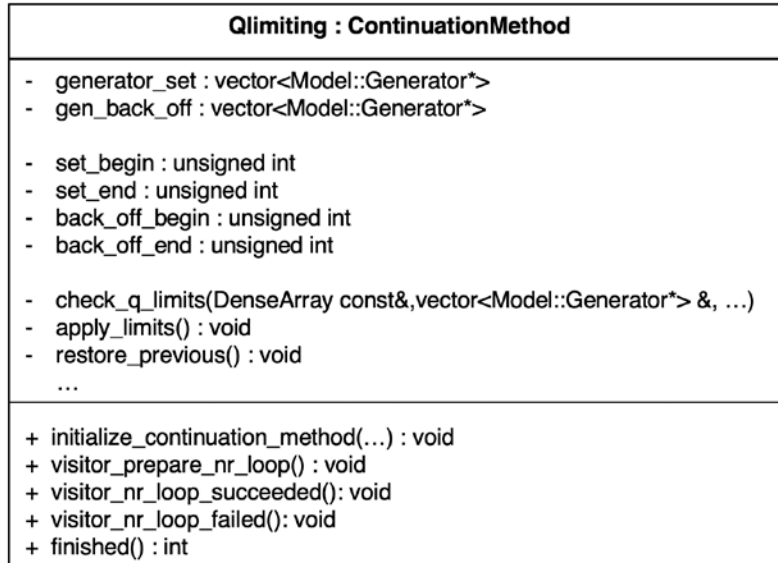


Figure 56: UML class diagram of the *QLimiting* class that implements the *ContinuationMethod* interface.

Figure 56 shows a reduced UML class diagram of the *Qlimiting* class. To fully focus on explaining the structure of the algorithm, we omit the sets of other components that hit reactive power limits in this diagram. Other components that are evaluated are continuously modeled switched shunts, FACTS devices, and VSC DC lines.

The core of the Q-limiting algorithm is implemented within three private functions. The *check_q_limits()* function finds all generators that hit their limits given the current solution vector and stores them for future use. The *apply_limits()* function applies limits (converts PV-generators to constant PQ models) to all or a subset of generators that hit their limit. The subset of generators to be limited is defined by the *limit_begin* and *limit_end* indices that are adapted to find subsets of generators for which the simulation converges. In an extreme case, both indices point at a single generator at a time. The *restore_previous()* function restores the previous state in case of a failed limiting attempt. Importantly, every function also applies to generators that back off from their limits.

Figure 57 shows how the private control functions of the Q-limiting algorithm are utilized by the *ContinuationMethod* interface's visitor functions. One can see that once the core of the control algorithm is implemented, mapping the control actions to the interface is straight forward and only requires minor logic.

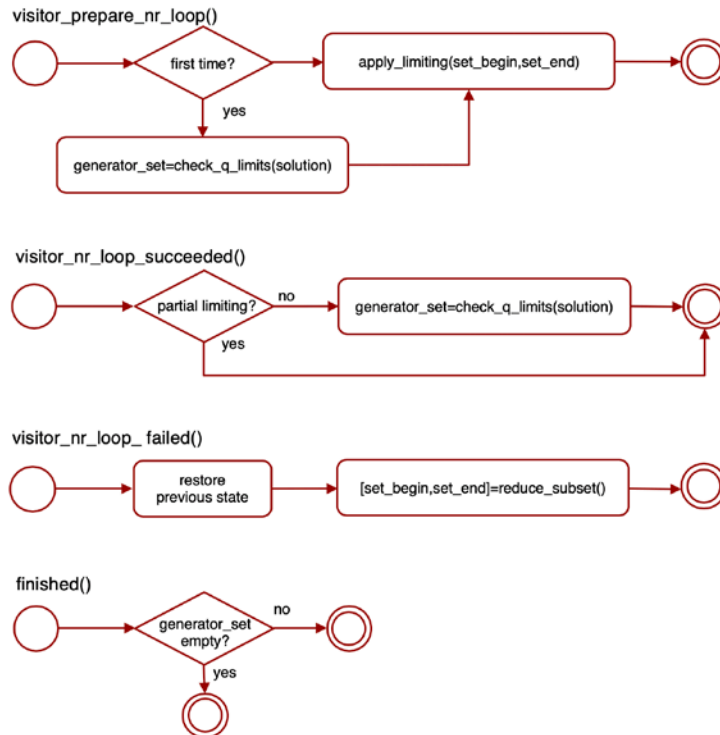


Figure 57: Control algorithms implemented by QLimiting.

To explain Figure 57 that shows the control algorithm's visitor functions in more detail, we walk through one control loop iteration of the Q-limiting algorithm. The algorithm starts with the *visitor_prepare_nr_loop()* function that is called at the top of the control loop. In this implementation, the first call to this function is differentiated from subsequent calls. The first call invokes the *check_q_limits()* function to find an initial set of generators to be limited. Subsequently, it applies these limits. In ensuing calls, only the limits of a defined generator subset (*set_begin*, to *set_end*) are applied.

Next, the altered system is re-solved. After a successful inner loop, the search for generators that hit their limit as well as generators that back-off from their limit, is re-initiated. If the inner loop diverged, previously limited generators are reset to their prior state and the set of generators to be limited is reduced to a subset of the previous step. Finally, the *finished()* function is called at the bottom of the control loop to determine if the algorithm has finalized. It returns "false" if there are remaining generators to be limited, and "true" otherwise, which continues the loop.

A note on how a reduced set of generators is established in case of a failed attempt: In case the algorithm is parameterized to limit as many generators at once as possible, the set is cut in half. Dependent on the sorting of the generator vector that is an additional parameter, then either the generators with the highest Q limits, with the smallest Q limits, or generators that exceeded their limits by the highest or lowest amounts are prioritized.

In a final stage, when the subset is reduced to a single generator, every generator limit is applied individually.

Finally, it is worth mentioning that this implementation includes an option that is advantageous when composing the Tx-stepping algorithm and the Q-limiting algorithms to a single algorithm. This combined algorithm is applied to aid convergence of very complex systems with many remote generators. In fact, only generators that are controlling remote voltages are considered in this case. Likewise, the “partial limiting” option that only limits generators that were initially found to hit a limit is used for this composed algorithm.

Device Control Algorithms

As we have mentioned, it is possible to compose multiple control algorithms together to composite control algorithms. Importantly, the *ContinuationMethod* interface is general enough to allow composite algorithms to be implemented utilizing the same interface as simple control algorithms. One such composite algorithm is the previously mentioned Tx-stepping algorithm for complex systems with many remote-controlled voltage magnitudes, in which generators controlling these magnitudes are limited during the Tx-stepping algorithm to aid convergence.

Importantly, implementing atomic control algorithms first and further composing them as composite algorithms that still implement the *ContinuationMethod* interface keeps the solver design clean even with a large number of implemented control algorithms. An alternative scenario that does not define such an interface is prone to become difficult to maintain after a certain number of implemented algorithms. Within such an implementation, the danger of unwanted interactions between the algorithms increases rapidly.

Composed algorithms

An example of a composite control algorithm, that comprises of an adaptive number of sub-algorithms is the *DeviceControl* algorithm. It includes possible control of generator Q-limits, transformer taps, discrete switched shunts, and an algorithm to converge PF after DC line parameter changes.

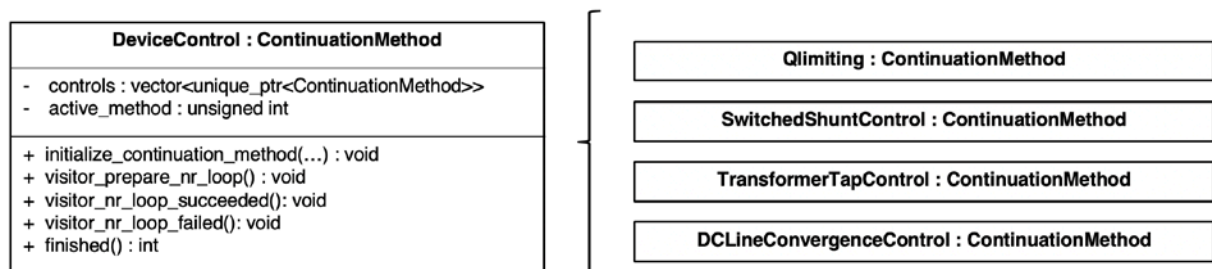


Figure 58: UML class diagram of the *DeviceControl* class.

Figure 58 shows the UML class diagram of the *DeviceControl* algorithm. One can see that it stores every sub algorithm in a vector of algorithms and keeps track of the currently active algorithm via an index. The *DeviceControl* algorithm's visitor functions relay the function call to the currently active sub algorithm. This creates a generic structure in which algorithms can be composed freely by adding them to the *controls* vector. The sequence in which they are added determines the sequence they are invoked. The *DeviceControl* control algorithm finalizes when all sub-algorithms finalize.

5.1.4 Data pipeline

We have discussed the “ECF solver” structures and their corresponding control algorithms. Now, we will introduce the data structures that enable simulation of Power System problems. The C++ SUGAR implementation can read two commonly available formats of Power System models: PSSE .raw files, and Matpower .m files. Additionally, an internal file binary file format (.sug) can be used. The data that is defined in these files is parsed into the *PowerFlowData* class. This raw data is further refined to equivalent circuit models of each simulation component. These components compose the *PSModels* class that represents the ECF equivalent system model of a problem.

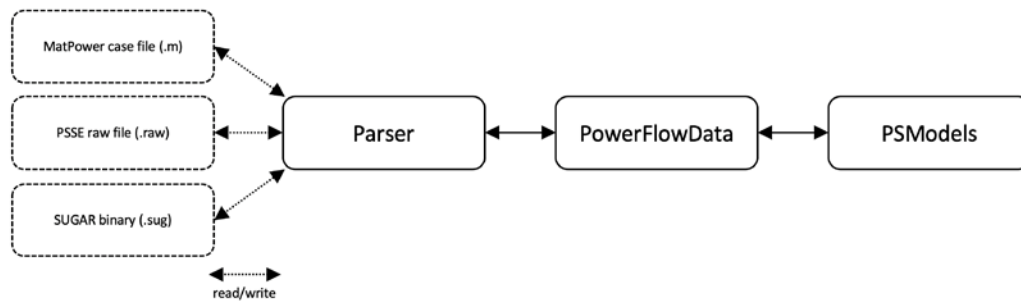


Figure 59: Relationships between model data structures of the SUGAR C++ implementation.

Figure 59 gives an overview of the model data structure of SUGAR C++. We will further discuss details of each component, starting with the *Parser*.

5.1.4.1 Parser

The data parser reads and writes PF case data to files. It implements an interface that enables the extension to additional file types without code changes outside of the *Parser* class.

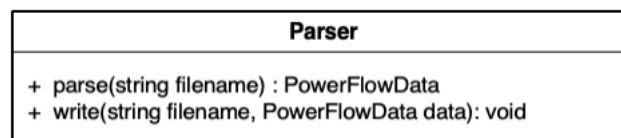


Figure 60: UML diagram of the *Parser*.

As we can see from Figure 60, its interface is straight forward. It includes a *parse()* function that returns a *PowerFlowData* object, and a *write* function that writes a *PowerFlowData* object to a file. Importantly the *Parser* enables the implementation of multiple sub parsers, that all implement the same interface and are instantiated by the *Parser* object, which means it utilizes the factory software design pattern.

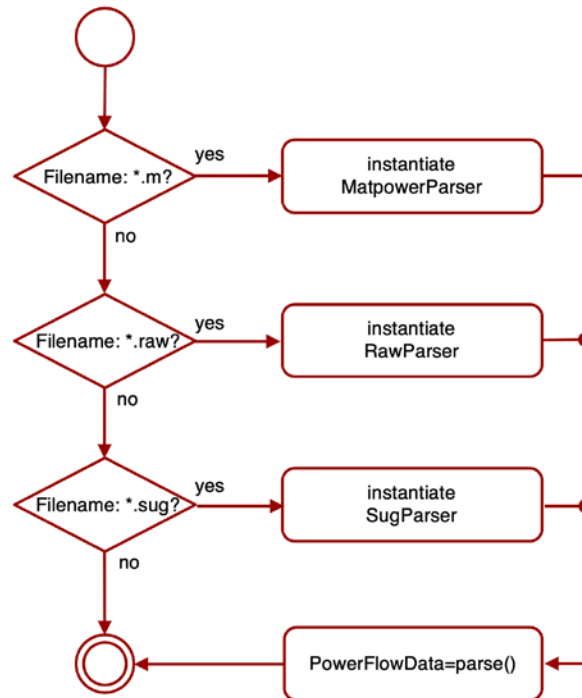


Figure 61: Algorithm of the SUGAR C++ parser that applies a factory pattern to instantiate the correct algorithm.

Figure 61 shows an activity chart of the *parse()* function's algorithm. After the *parse()* function is called, a solver object that specializes in parsing a certain file type is instantiated based on the ending of the input file that is parsed. Then the data is parsed and a *PowerFlowData* object is returned. The *write()* function of the *Parser* is implemented similarly.

PSSE ".raw"-file parser

As an example of a specialized parser, we discuss details of the PSSE .raw file parser that reliably parses ".raw"-files from file version 30-34. Interestingly, parsing every ".raw"-file version with one parser proves to be a complex endeavor, since the file structure between .raw file versions changes erratically. For example, in the bus definitions of version 30 and 31, shunts are defined immediately after the bus type. These positions shift for other versions. Additionally, the position of the owner definition switches place. In versions after 32 it is defined in front of the solution, while in versions before 32 it comes after the solution. Erratic changes like these continue for many elements between .raw file versions. This is exacerbated by the fact that a version number

was only included into files after version 32. When a version number is defined, it is commonly faulty within files in general. This leaves a parser with the only option to infer the file version from the file structure. Despite these challenges, our parser is heavily tested on a multitude of cases and operates reliably for even the most complex ones.



Figure 62: Class diagram of the .raw file parser.

Figure 62 shows the UML class diagram of the *RawParser*. It can be seen that a parse function for each raw file section is implemented. Correspondingly, write functions for each of these sections exist.

The parser works in the following way. It splits the “.raw”-file into its sub-sections and infers the file version from their structure. Next, it parses each section by tokenizing it line by line and assigning the parsed values to a *PowerSystemData* object. Finally, the filled *PowerSystemData* object is returned.

The Matpower file parser operates similarly. It first splits the “.m” case file into sections and further tokenizes the data, before filling the parsed data into a *PowerSystemData* object. Importantly, this is only possible because the data defined in a Matpower case file is a subset of the data that can be defined as part of a “.raw”-file. While Matpower case files exhibit less complexity compared to .raw files, their structure is not as rigidly defined, which slightly increases difficulties in tokenization.

5.1.4.2 PSModels – Class

The *PSModels* class loads the data from the *PowerFlowData* class to create corresponding ECF models that can further be used for simulation. It serves as a container for ECF models that together comprise a system model. Additionally, it implements functionality such as sanity checks of the data, and interfaces to apply continuation methods, control algorithms, or contingencies.

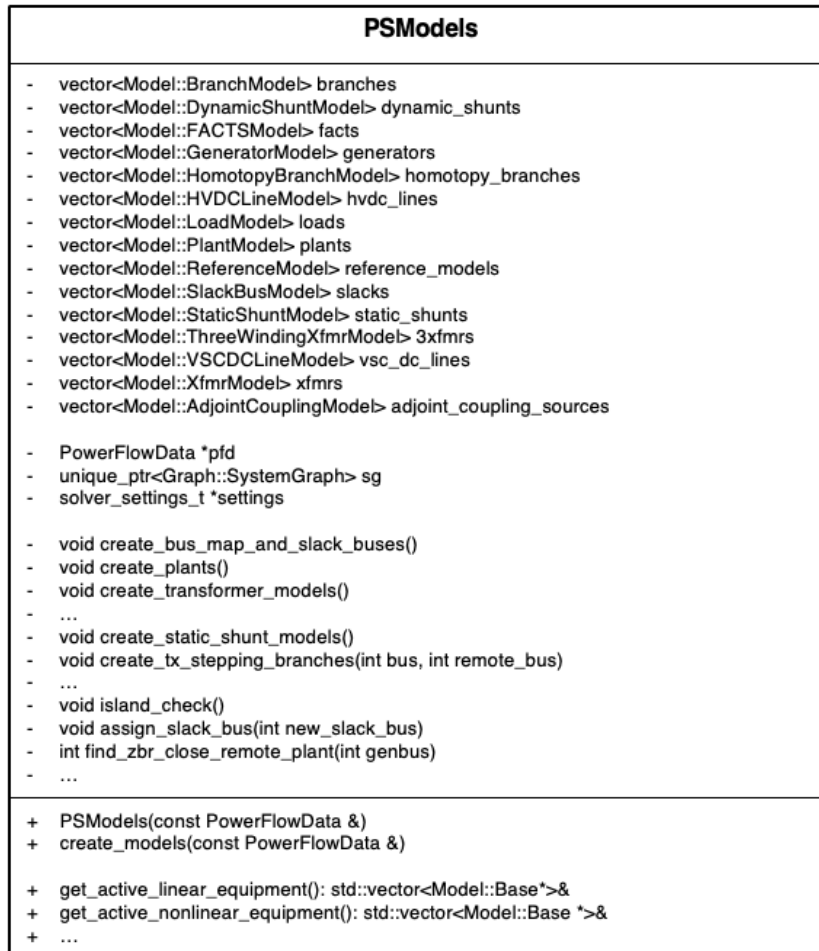


Figure 63: UML class diagram of the *PSModels* class.

Figure 63 shows an UML class diagram of the *PSModels* class. One can see that each class of ECF models is stored in separate vectors. This simplifies access to these models for control algorithms, post simulation analysis, and several other utility functions. Importantly, the functions *get_active_linear_equipment()* and *get_active_nonlinear_equipment()* are utilized by the NRSolver to obtain vectors of linear and nonlinear models to be stamped during a NR loop. Noteworthy, separating models into vectors for each type instead of storing all models together in a single vector using an interface speeds up the stamping process dramatically. This is because the reduced overhead of virtual function call tables.

A *PSModels* class can directly be instantiated using a *PowerFlowData* object. Alternatively, PF data is loaded into the class via the *create_models()* function. To create ECF models, specialized “create” functions for each model class are implemented. Some ECF models, like static shunts, transmission lines, or load models are straight forwardly created from the supplied data. Other elements involve additional calculations.

Many of these calculations require a graph representation of the system in order to be implemented efficiently. An example for such an algorithm is the obligatory sanity check

that each system island has an assigned slack bus. The *SystemGraph* object provides this graph representation and is accessible for the *PSModels* class via a reference. The implementation of the *SystemGraph* class is discussed in Section 5.1.6.

Finally, the *PSModels* class contains a reference to the *solver_settings* object. Solver settings that affect models and their creation include a system wide loading factor, a flag that enables ignoring bus labels, and other variables such as a minimum electrical distance between plant models. Interestingly, ignoring bus labels is sometimes useful due to an ambiguity that results when a generator is defined on a bus that is set to be a load-type bus. Usually, this is the case when a generator hit its reactive power limit and is interpreted as a constant real and reactive power device when the case-file is written. However, this situation is ambiguous since it could also indicate that a generator's status flag is set incorrectly. The user is informed about this situation via a warning message. To further enable simulation of both possible scenarios, we only instantiate generators that are located on generator-type busses by default but allow overwriting this behavior by an "ignore bus labels"-flag.

5.1.4.3 Power System Analysis Models

The *PSModels* class aggregates all models within the SUGAR C++ implementation to a system that in general poses a nonlinear equivalent circuit simulation problem. All models implement the same interface for stamping their contributions to the system matrix and have their MNA nodes assigned.

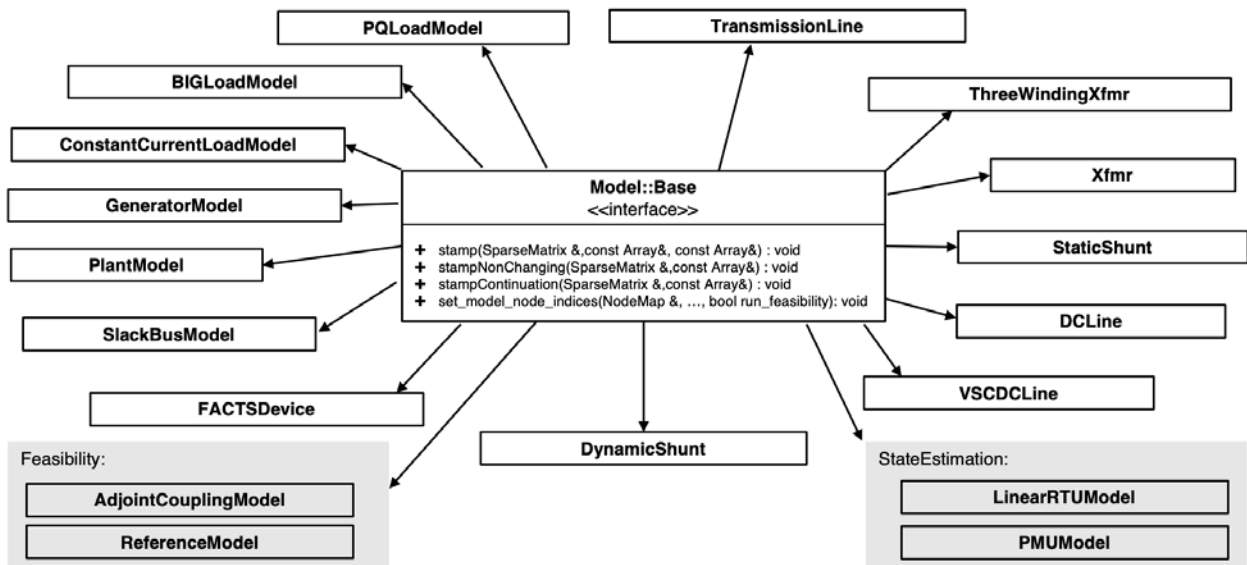


Figure 64: UML diagram of PF, feasibility PF, and SE models with their common interface *Model::Base*.

Figure 64 provides an overview of all models implemented in SUGAR C++. These are the PF models, additional models for the PF feasibility optimization algorithm, and the State Estimation algorithm. Notably, when an optimization-based simulation is started,

each PF model additionally stamps its corresponding adjoint equivalent circuit contributions.

The implemented PF models correspond to all models that are defined in PSSE “.raw”-files. We have validated each model’s correct implementation with versions of the PSSE and the PowerWorld simulators. This validation process can become quite complicated, considering the amount of different input parameter settings that are possible for some models. This is especially true for transformer parameters.

Transformer Model

Transformer models of PSSE “.raw”-files have three parameters that establish which quantities are used to define a transformer’s tap ratio, phase angle, and losses. Together, these 3 parameters specify 18 different combinations of parameters that are converted into the transformer’s electrical model parameters. Our implementation has cross validated all 18 scenarios with PSSE and PowerWorld.

Three Winding Transformers

Three winding transformers of “.raw”-files are defined in delta-configuration using the same set of parameters as two winding transformers. Our implementation converts them to a star configuration and models each star component as a two-winding transformer model. Importantly, each winding has its own state that can be on or off. We create the auxiliary star node as a set of two auxiliary matrix entries and do not assign an artificial bus number to this node. Notably, a three winding transformer only defines one magnetizing loss element that is modeled on its primary winding.

Plant Model

Plant models are composite models that allow multiple generators to control a single voltage magnitude of a bus different from their own. They are the SUGAR C++ implementation’s solution to sets of “.raw”-file generators that control a single remote bus voltage magnitude. Instantiating such a model requires multiple auxiliary algorithms that are now presented.

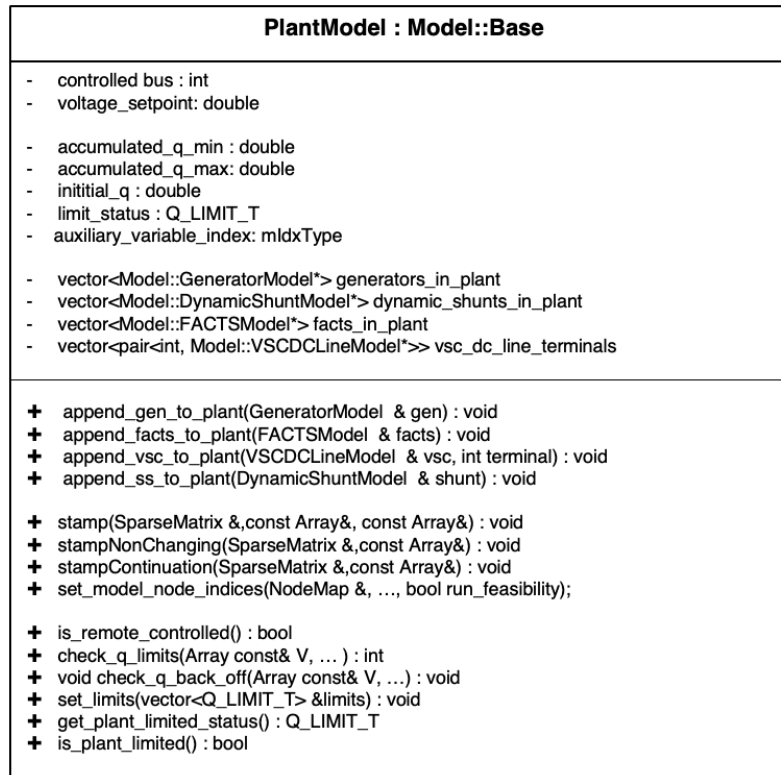


Figure 65: UML class diagram of a Plant model.

Figure 65 shows an UML class diagram of a Plant model. The purpose of this model is to control a bus (the *controlled_bus*) to a voltage magnitude (the *voltage_setpoint*). To enable this control, a set of voltage controlling models cooperates. These models include generators, dynamically modeled shunts, FACTS devices, or VSC-DC lines (to be more precise a certain terminal of a VSC-DC line). The plant model stores pointers to every cooperating model in order to have access to these models. Furthermore, functions to append models of each type to a single plant are implemented.

Importantly, since all models control exactly one bus voltage magnitude, only one auxiliary variable is added to the system of equations. This auxiliary variable defines the reactive power that is necessary to control the bus's voltage magnitude to its setpoint. To split the total reactive power into each model's contribution, participation factors are defined. The sum of all participation factors is normalized to one, defining fractions of reactive power contributions to each element. Unfortunately, assigning these factors can become arbitrarily complex, especially when prioritizing reactive power delivery of certain models over others. This is possible since, in reality, all controls operate independently from each other. For example, it is valid for an operator to choose to change the setpoint of a switched shunt, or FACTS device up to their limits before generators adapt their contribution. Nevertheless, this macro modeling of reactive power control represents a compromise that works sufficiently given the provided information.

Creating a plant model additionally requires algorithms that check for validity of the created model. In order for a plant model to work properly, a clear control path between each generator and the controlled bus has to exist. A clear path is defined by a connection that does not include other voltage-controlled buses. We check for this, utilizing a shortest path algorithm of the SystemGraph. An additional requirement for a controllable plant is its electrical distance from neighboring plants with different voltage magnitude setpoints. It is easily imagined that two voltage magnitude setpoints that are electrically very close to each other result in extremely high branch currents that are likely unphysical and cause numerical stability problems for the solver. To evade this possibility, plant models must adhere to a minimum electrical distance to each other. A breadth-first search algorithm for electrically close plants is used to avoid such a scenario. If our implementation encounters two plant models that are electrically too close to each other, it merges both plants to a single one. This is a valid assumption, since electrically close nodes will result in minimal voltage differences.

It is worth mentioning the following data peculiarity that is a common occurrence: Sometimes multiple models are set to control a single magnitude but define different magnitude setpoints. In this case, our algorithm prioritizes generator setpoints. For other discrepancies, the first encountered setpoint during model creation is used. Additionally, a warning message drawing attention to this ambiguity is printed.

Finally, we discuss a plant model's interfaces. From Figure 65 we see that a plant model implements the *Model::Base* interface that defines functions for matrix stamping and MNA node distribution. Additionally, interface functions that are utilized by the *QLimiting* control algorithm are implemented. Noteworthy, when a single generator of the plant hit its limits, the distribution factors of the remaining generators have to be reassigned. The plant is only considered to be reaching its limit when each of its elements is limited. When backing off from a limit, a plant assigns every single generator for back-off at once.

Switched Shunts

The instantiation algorithm of switched shunt models is notable. Switched shunts can be modeled in multiple ways. When inserted as dynamical switched shunts, they are treated similarly to generator models. In our C++ implementation, they are part of *Plant* models and are limited via the Q-limiting algorithm. Discrete switched shunts, however, are modeled as static shunts within a Newton-Raphson loop. They require an outer loop control algorithm to be adjusted.

Adjoint and Coupling Models

For optimization-based algorithms, each element model additionally stamps its adjoint counterparts. The PF feasibility optimization algorithm requires two additional types of models to be instantiated: A *ReferenceModel* is the adjoint counterpart of a

SlackBus model that sets the adjoint reference, and an *AdjointCouplingModel* couples primary and adjoint circuits via feasibility sources.

5.1.5 SugarAlgorithms

The aforementioned *SugarAlgorithms* class defines a global interface to the available algorithms and sub-algorithms of our SUGAR C++ implementation.



Figure 66: UML class diagram of the *SugarAlgorithms* class.

Figure 66 provides a UML class diagram of the *SugarAlgorithms* class. It defines algorithms for Power Flow, and State Estimation simulations. Furthermore, Monte Carlo simulations based on both simulation types can be started. The Power Flow feasibility optimization uses the same algorithm as a Power Flow simulation with a *run_feasibility* flag signals the additional creation of adjoint circuits. Furthermore, *SugarAlgorithms* defines auxiliary algorithms for State Estimation that create synthetic SE cases (*createStateEstimationSample()*) and calculate performance metrics of SE solutions (*calculateStateEstimationMetrics()*).

Functions to write output files are defined in the *SugarAlgorithms* class. They include a *save()* function that writes power system case files dependent on the file ending of the given file name (".m", ".raw", or ".sug), and specialized write functions that output ".csv" based files of a system's line loadings and generator set points.

Finally, sub-algorithms to create outer control loops are defined, standardizing their creation process. These algorithms return a pointer to the control algorithms interface class only, simplifying their handling. Notably, the *createNR()* function defines a simple Newton-Raphson loop that only applies limiting algorithms, if requested, without any outer loop controls.

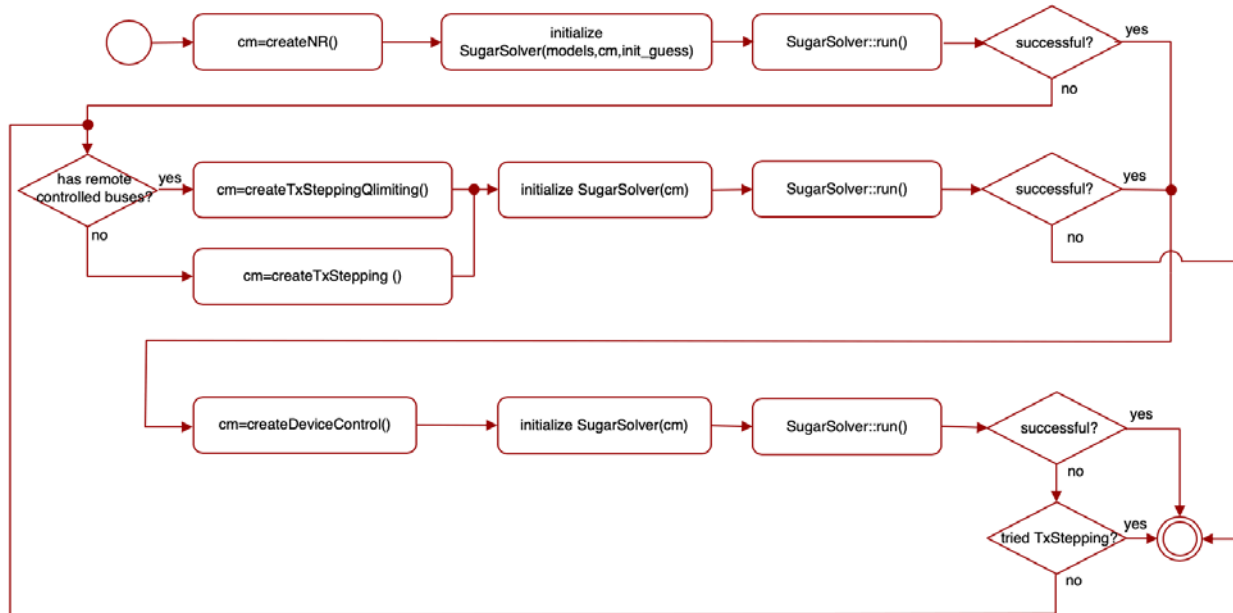


Figure 67: Flow diagram of the *runPowerFlow()* function of the *SugarAlgorithms* class.

Figure 67 shows a diagram of the *runPowerFlow()* algorithm. Visibly, these algorithms become complex quickly, therefore, we use these base-algorithms as building blocks for more evolved algorithms. For example, an algorithm running multiple artificial State Estimation cases calls the the *runPowerFlow()* algorithm to create a base solution and further creates SE cases utilizing the *createStateEstimationSample()* algorithm and subsequently the *runStateEstimation()* and the *calculateStateEstimationMetrics()* function in a loop.

5.1.6 Network Graph Library

Several algorithms within the SUGAR C++ implementation require a graph representation of the system to be implemented efficiently. Some of them are system island exploration, search for corresponding slack buses, and plant model minimal electrical distance enforcement.

The decision of implementing a lightweight graph library was made after exploring several third-party library options that did not fit the purpose. In fact, after careful design, the *SystemGraph* library's performance is close to the performance of optimized graph libraries.

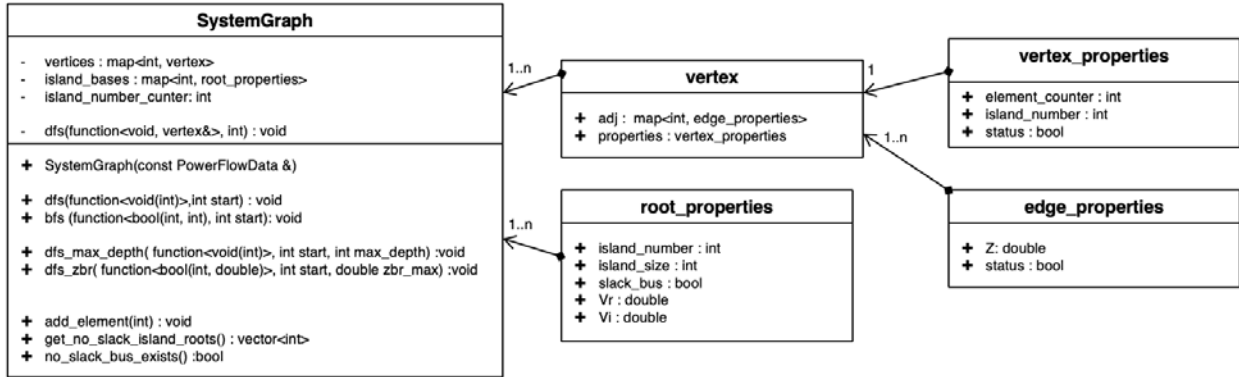


Figure 68: UML diagram of structures and classes that represent the *SystemGraph* library implementation.

Figure 68 shows a UML diagram of the *SystemGraph* library's data structures. We decided to utilize an adjacency list-based data structure that is optimal for sparse graphs like a power system's network graph. Compared to an adjacency matrix-based implementation, it is much more space efficient without having disadvantages during graph traversal. Our adjacency lists are stored in map containers of the C++ standard library.

The *SystemGraph* stores a map of vertices, that is indexed by corresponding system bus numbers. Furthermore, each vertex stores a map of edges and *edge_properties* and includes a *vertex_properties* structure. Herein, *edge_properties* include a measure of electrical distance (the magnitude of the complex impedance) and a status flag. Furthermore, *vertex_properties* include an *element_counter* that defines the number of connected models, the island number of a vertex, and a status flag. Importantly, these are the properties utilized in the graph-based algorithms of the current implementation. These properties are easily extended, since adding additional ones does not require any changes in the graph algorithms.

This graph structure can be traversed using search algorithms. A depth-first as well as a breadth-first algorithm were implemented using non-recursive algorithms with a focus on performance. They allow the implementation of more complex algorithms utilizing visitor function calls. The algorithm for minimal electrical distance utilizes the *dfs_zbr()* function that includes the electrical distance calculation directly within the search, which simplifies its implementation and improves performance. A note on performance, theoretically both search algorithms scale with $O(|V| + |E|)$ and require $O(|V|)$ memory, where $|V|$ is the number of vertices and $|E|$ is the number of edges.

Importantly, for graph construction, a highly performant implementation is necessary as well. This becomes clear when looking at system sizes of 80k buses and over 100k branches that are partly defined in different data structures. We implement this initialization, with one data pass for each bus and branch type. After initialization we utilize a depth-first search algorithm to explore system islands.

The *island_bases* data structure stores roots for each island, where a root is an island's slack bus if at least one slack bus exists (signaled by the *slack_bus* flag of the *root_properties* data structure). Utilizing the *island_bases* data structure, our implementation efficiently finds islands that do not have a slack bus assigned and fixes the problem. Additionally, traversing the vertices based on islands provides an efficient way of building the slack bus voltage-based initial guess of the Tx-stepping algorithm.

None of the algorithms implemented in this library cause a noticeable delay for any simulation. In a worst-case scenario of a PF simulation, an Eastern Interconnection planning case that defines multiple islands and requires few iterations to converge due to good available initial guesses, the library's share in total computational cost is around 2.6%. For a 118-bus case power flow run, this share of total computational cost drops to 0.29%.

5.1.7 Settings and Input File Format

This SUGAR C++ implementation enables the simulation of three mayor algorithms for Power Systems (PF, PF feasibility, and SE) and implements a probabilistic solution framework. All of these components are configured using multiple parameters. We organize these parameters in a json-based settings file that is further parsed into accompanying data structures for each solver component.

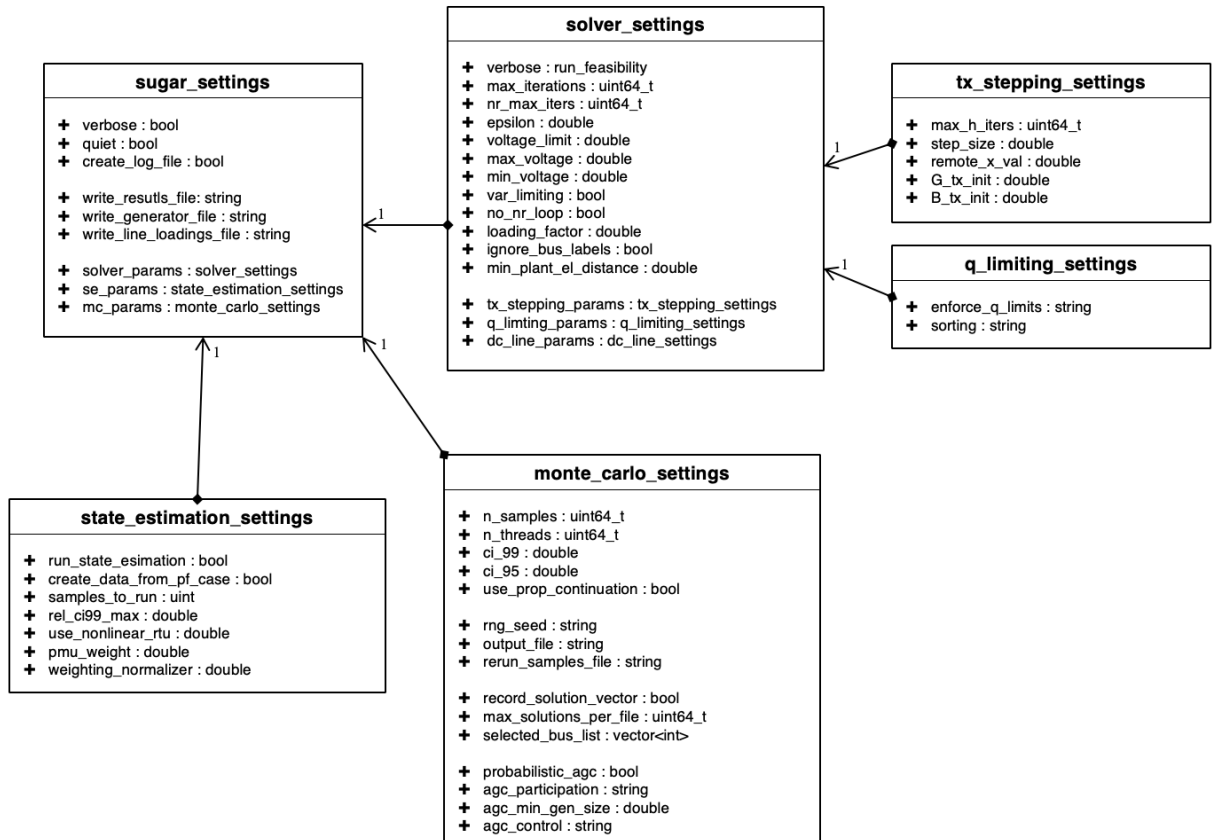


Figure 69: UML class diagram of input parameter data structures.

Figure 69 provides an overview of these data structures. They include global settings such as the verbosity of the solver, solver parameters, parameters of control algorithms, as well as parameters for composite algorithms such as State Estimation or probabilistic analysis.

5.1.8 Used Third-Party libraries

It is infeasible to design a high-performing software package without the efficient use of third-party libraries for tasks that are not unique to the developed software. The SUGAR C++ implementation utilizes libraries to solve sparse matrix problems, log progress to the console or to log-files, read and write data in json-format, and to generate random numbers.

5.1.8.1 Sparse Matrix solvers

We utilize libraries of the SuiteSparse package to solve sparse matrices. Specifically we implemented *SparseSolver* interfaces for the UMFPack [100] and the KLU [99] solvers. Both libraries implement LU factorization algorithms.

KLU

The KLU library implements a sparse left-looking factorization algorithm and relies on permutations to block diagonal matrix forms. Additionally, it implements algorithms to reduce matrix-fill ins during factorization. Herein, the KLU set of algorithms is designed to specifically perform well when solving “SPICE-like” problems [99]. In the SUGAR C++ implementation, KLU generally outperforms UMFPACK (see Section 5.1.9), which is most likely due to this characteristic.

UMFPACK

The UMFPACK library uses an Unsymetric MultiFrontal method and a direct sparse LU factorization algorithm [100]. Compared to that from KLU, this algorithm is more general. UMFPACK relies on BLAS (Basic Linear Algebra Subprograms). Hence, for optimal performance it is important that it is linked to a BLAS library that is optimized for a system’s architecture. Noteworthy, for probabilistic analysis with multiple threads solving matrices in parallel, it is imperative that the BLAS library is configured to run sequentially. Otherwise, the expected speedup might not materialize.

5.1.8.2 JSON-Parser

To read and write json-input based files we utilize the open source “JSON for modern C++” library [101]. It is high performing and provides a simple-to-use interface with predefined getters for standard data types. Especially convenient is its ability to directly map JSON containers to C++ standard library containers. To read and write more complex data structures, one can implement two interface functions

```
void to_json(json&, const data_structure &)  
void from_json(const json &, data_structure &).
```

where *data_structure* is the type of the respective data structure. SUGAR C++ utilizes this library to parse the solver settings as well as to read probabilistic input files and write statistics of the probabilistic results as json-based output files

5.1.8.3 Logger

Logging the progress of a command line tool is vital, since this is the only way to track a tool's progress. Hence, the speed of the logger library is performance critical to the whole simulator, especially when the solver is set to verbose mode and logs inner loop information. The SUGAR C++ implementation uses the spdlog library [102] that is a very fast, thread-safe, header-only library for C++. Benchmarks promise that logging more than 5 million strings per second is possible when the library is accessed by a single thread.

5.1.9 Performance Evaluation of SUGAR C++

The total simulation time of nonlinear simulators depends on the problem and its initial conditions. Hence, the total simulator runtime is not a good measure to evaluate the performance of an implementation. This is especially true if homotopy methods are employed to solve a problem due to the unavailability of good initial conditions.

5.1.9.1 Computational Cost

To evaluate performance of the solver, we will first look at how the inner solution loop performs. This is defined by the per-iteration solve time of the simulator, which is a cost incurred for each nonlinear-solver iteration. The time to build the initial nonchanging matrix can be seen as a fixed cost. However, for linear algorithms or algorithms that need few iterations to converge, this cost might be substantial. We will now look at these timing measurements for different sized systems and for different solver settings.

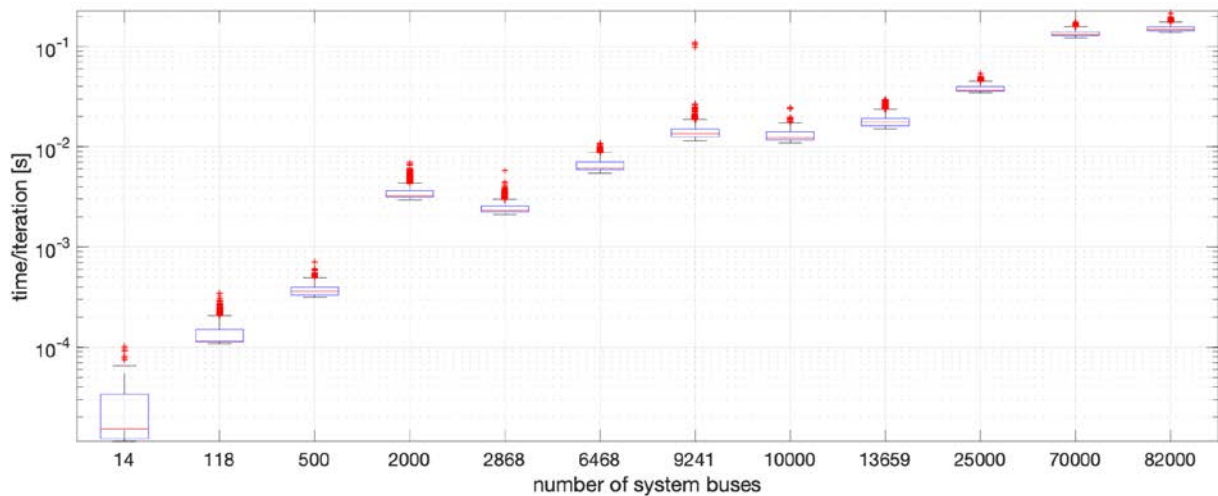


Figure 70: Per iteration solution times of the SUGAR C++ implementation for openly available Power Flow cases using KLU.

Figure 70 shows per iteration times of the solver on different size, openly available power flow cases [103], [104]. The time shown includes the total time for nonlinear matrix stamping, sparse matrix solution, solution vector post processing, and final condition evaluation. In Figure 70, the KLU sparse matrix solver was used. We will take a closer look at the time contributions of each component using a profiler. Importantly, these inner loop times are the time values where small improvements can magnify to big overall performance improvements of the solver.

In contrast to these inner loop times, the initial linear matrix creation time is a one-time cost, since these matrices are reused for each iteration during the analysis.

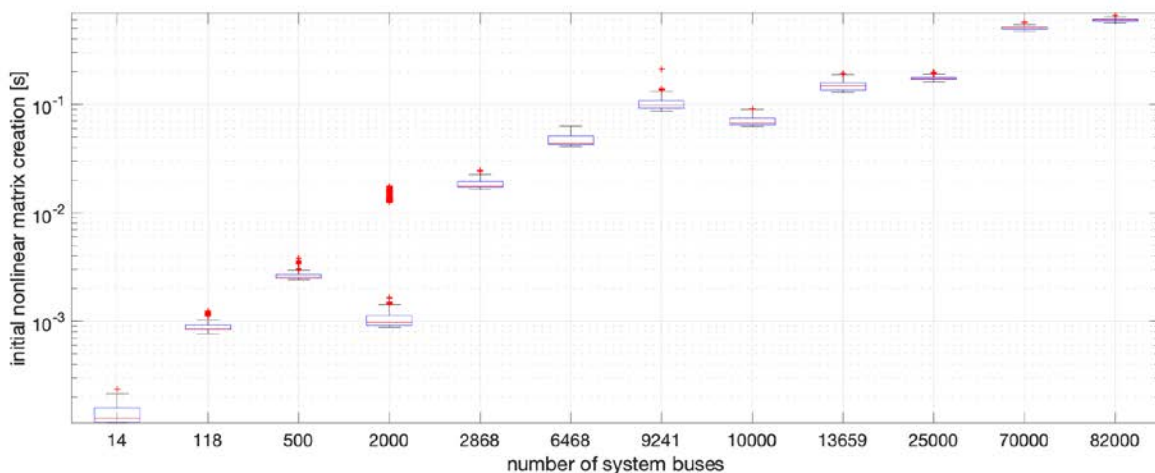


Figure 71: Initial Power Flow matrix creation time of selected power system cases.
This is a one-time cost.

Figure 71 shows the initial linear matrix creation times. Compared to the per iteration times these costs are higher. However, since the linear matrix is only created once, the total impact of linear matrix creation on the total solution time is generally negligible. However, these times do give us a good measure for the speed of the stamping algorithm.

5.1.9.2 Performance Profile

Next, we discuss the relative impact of different solver components on the total simulation cost. To create performance profiles of our software, we used the “callgrind” tool [105] of the profiling software Valgrind. First, we will look at cases where a good initial guess is available and the convergence is reached within a few iterations. Then we will look at cases where Tx-stepping is needed for them to converge. This increases the amount of total iterations to problem convergence and adds calculations of the outer control loop (Tx-stepping) to the total solver performance.

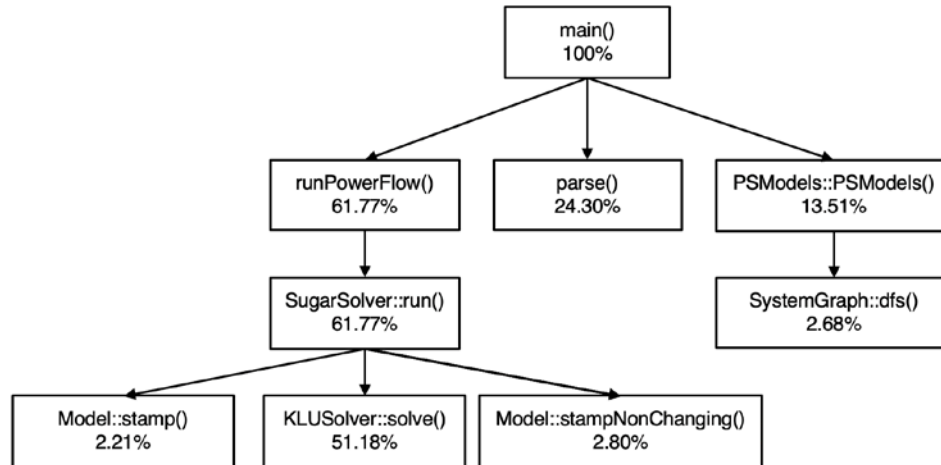


Figure 72: Performance of the SUGAR C++ solver for a real Eastern Interconnection case and good initial conditions

Figure 72 shows the solver's component's contributions to total performance when running PF on a real Eastern Interconnection case (over 78k buses) with good initial conditions. Herein, generator Q-limits are taken care of within the simulation. For this scenario we see that parsing the ".raw" text file actually takes up 24.3% of the total simulation time. Creating the ECF models takes another 13.51%. The nonlinear solver algorithm runs for 61.77% of the time. Here, 51.18% is spent solving linear sparse matrices, and 2.21% stamping nonlinear elements. Interestingly, for simulations that converge quickly, building the linear matrix still requires 2.8% of the total simulation time. Interestingly, this is using the KLU sparse matrix solver library that outperforms UMFPACK for this type of matrix. Using the same test set up, we found UMFPACK to incur 79.5% of the total simulation cost. From Figure 72 we see that sparse matrix solution is the bottleneck of the total simulator performance in our solver. For simulations that take more iterations and outer loops with control algorithms, this situation becomes even more clear.

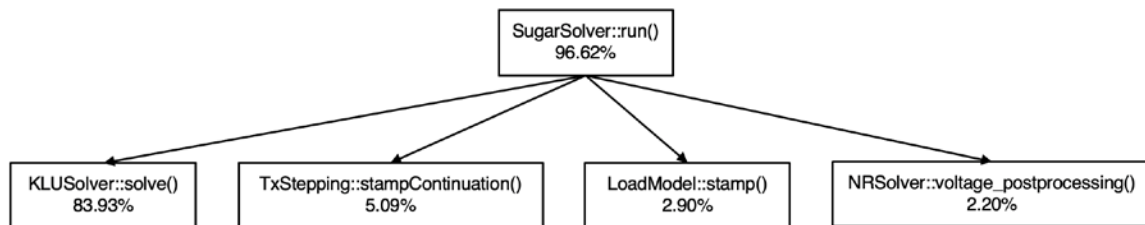


Figure 73: SUGAR C++ profile of a real eastern interconnect PF simulation using Tx-stepping.

Figure 73 shows the profiler output for the Eastern Interconnection case without a good initial guess. The solver needs more iterations, including multiple Tx-stepping steps to find a solution. Hence, the inner loop performance becomes more important. This inner

loop performance is limited by sparse matrix solving, which takes up 83.93% of total simulation time. Other functions such as the *stampContinuation()* method that stamps the Tx-stepping system changes, the nonlinear stamps, and the voltage postprocessing together incur around 10% of total computational cost.

All previous performance studies show the solver's performance for PF problems. When solving problems involving optimization algorithms within the ECF, we deal with matrixes that are double in size as compared to PF. However, since our SE algorithm is linear, only a single sparse matrix solution is required.

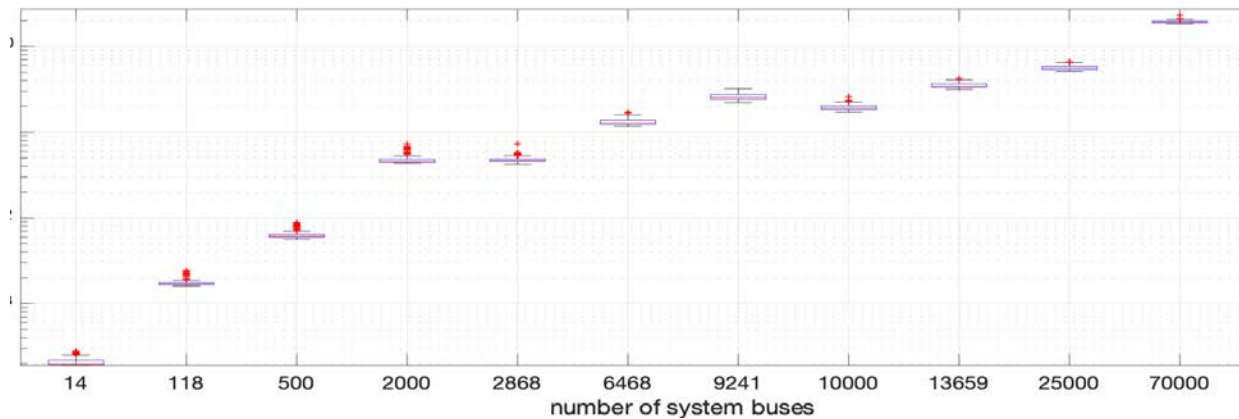


Figure 74: Solution times of synthetic State Estimation samples that are crated from openly available power flow data.

Figure 74 shows convergence times for synthetically created state estimation cases that were solved using our linear algorithm. We notice that we solve the biggest SE cases in the order of one second. This computational effectiveness suggests the possibility of probabilistic SE studies. We will present such studies in the Results Section.

5.1.9.3 Memory Profile

Finally, we study the memory profile of our implementation using the “valgrind” tool for memory profiling “massif”. Within this tool we use the “--stacks=yes” flag to enable profiling of program stack and heap.

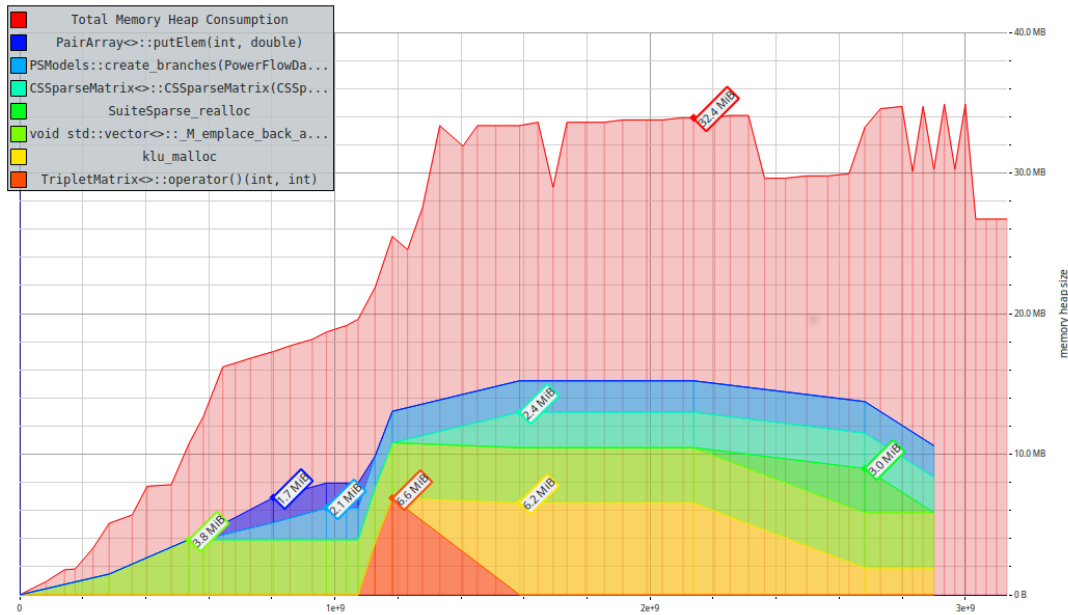


Figure 75: Valgrind massif memory profiler output for the 9241-bus PEGASE case.

The massif-visualizer [106] prepares a graphical representation of a program's memory-profile. Figure 75 shows such a profile for a Power Flow study of a 9241-bus test case. We see that the uncompressed system matrix (*TripletMatrix*) is the single biggest structure during this simulation. However, this data structure is short lived. It is destroyed with the creation of the compressed system matrix (*CSSparseMatrix*) (2.4 MB). After the uncompressed system matrix, the KLU solver is the second biggest data structure and requires 6.2 MB of memory. The peak memory use of this simulation was found to be 32.4MB. In addition to the already mentioned structures, this includes input-data, ECF models, additional sparse sub-matrices, the system graph, and stored intermediary solution vectors.

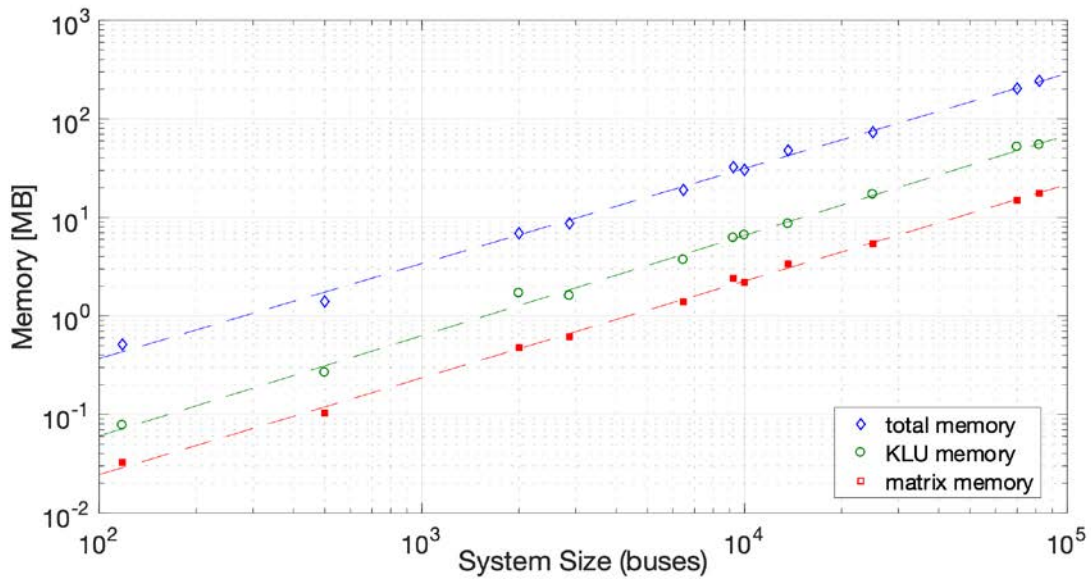


Figure 76: A double-logarithmic plot of requirement for different sized systems for total required memory, KLU memory (symbolic and numeric factorizations), and system matrix size.

Figure 76 shows the memory requirements of SUGAR C++ for different sized Power Flow cases in double logarithmic form. Interestingly, we find a close to linear relationship between system size and required memory for all three studied variables. This unexpected finding is due to increased sparsity of increasingly big test systems. Additionally, we find that for the studied systems the memory requirements of the KLU algorithm are on average 2.87 times the size of the system matrix, and the total memory requirements of the solver are on average 14 times the size of the corresponding system matrix.

5.2 Monte Carlo Framework

To create probabilistic results for the implemented ECF algorithms, we implemented a Monte Carlo based numerical probabilistic tool that bases on our ECF solver implementation. Our main goal for this MC framework implementation was to run efficient probabilistic simulations on the available server infrastructure.

5.2.1 Properties of Monte Carlo Algorithms

Simple random sampling Monte Carlo algorithms are known to converge with an error decrease of $1/\sqrt{n}$, where n is the number of samples. This means that for a tenfold increase in accuracy, one hundred times more samples have to be run. This is comparatively slow. Hence, algorithms exist to improve on this convergence rate. However, the MC algorithm is the only general algorithm for probabilistic studies and will be used in this implementation. Monte Carlo simulations offer almost complete data parallelism, which implies extremely good weak-scaling properties. These problems are commonly referred to as “embarrassingly parallel” problems.

5.2.2 Design Considerations

In order to design a high performing algorithm, multiple options arise. One option is to use bulk parallel programming techniques that efficiently scale across multiple compute nodes: These require post processing of Monte Carlo sample results on a file-system level or communication of results using message-passing libraries such as MPI [107]. Alternatively, shared memory programming offers the efficient utilization of single compute nodes utilizing a thread-level parallel paradigm. To create shared memory parallel programs, multiple libraries exist; e.g. the thread building blocks [108] or openMP [109] libraries. Since C++11, the “thread support library,” which is part of any C++ standard library, offers features to implement memory parallel programs. These features include direct interfaces to system threads, locks and atomic data types for synchronization, or higher-level structures such as asynchronous execution scheduling featuring asynchronous interactions with futures and promises.

In our implementation we aim for efficient use of a single compute node while keeping minimal overhead and maximum control of the programming model. Hence, we decided on an implementation using C++ thread support library functions. As we will see, this design choice also enables us to fully control data reuse and sharing through the C++ copy and move construction feature. Note that this design choice does not limit potential speed-up across multiple machines. In fact, this is trivially achieved by starting multiple thread parallel MC cores on different machines while utilizing different RNG streams to guarantee independent random samples.

5.2.3 Implementation

The Monte Carlo algorithm design is straight forward. We utilize parallel worker threads for computation, while printing intermittent updates in the main thread. However, careful synchronization is required when simulation results are stored in a shared data structure and when final conditions are determined based on all previous results.

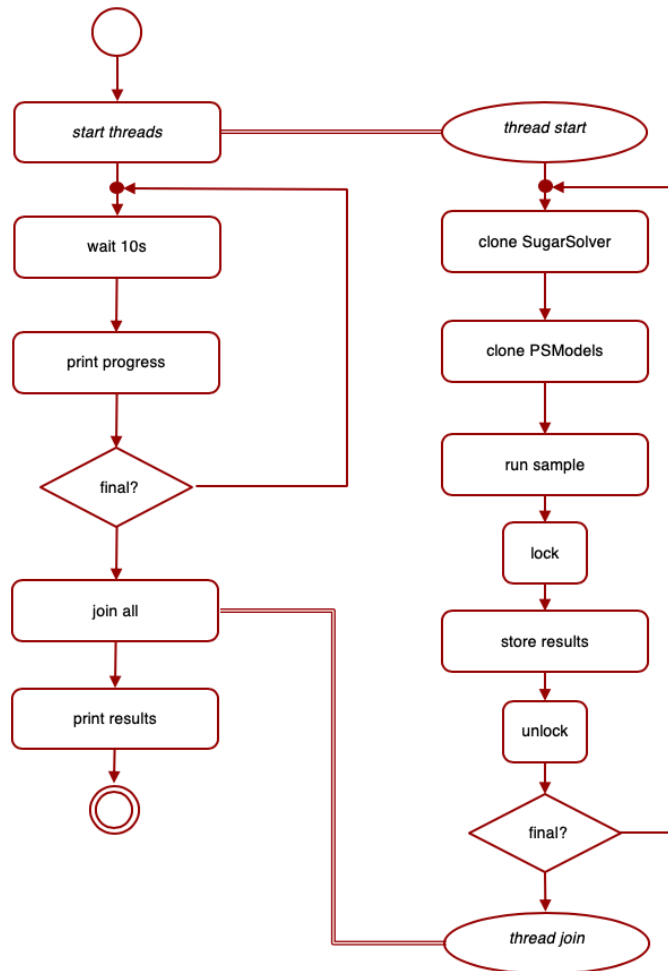


Figure 77: The Monte Carlo algorithm.

Figure 77 presents the *MonteCarlo* algorithm in a flow diagram. The algorithm starts parallel threads that have access to a *SugarSolver* and a *PSModels* structure. The worker-thread clones the *SugarSolver* that include an initial guess and shared matrix structures that do not change between samples. It then creates a probabilistic sample by cloning *PSModels* utilizing the specialized *clone_probabilistic()* function. Cloning of these data structures enables us to use move and copy to determine which data is shared and which data needs to be recreated for every probabilistic sample.

Next the worker thread starts the requested ECF based algorithm. After the algorithm's completion, the worker thread stores the solution in a data structure that is shared between all threads. A write lock needs to be acquired to access this data structure. After writing and establishing that this latest result did not meet a final condition of the Monte Carlo algorithm, the worker releases the write lock and returns to the top of the loop to calculate the next sample. When a final condition is met, the thread exits and notifies all other threads to suspend calculations. After all threads have joined the main thread, the results are printed and the Monte Carlo algorithm finalizes. Final conditions are defined by the maximum sample count or when the probabilistic algorithm reaches a confidence threshold on a simulation's parameter.

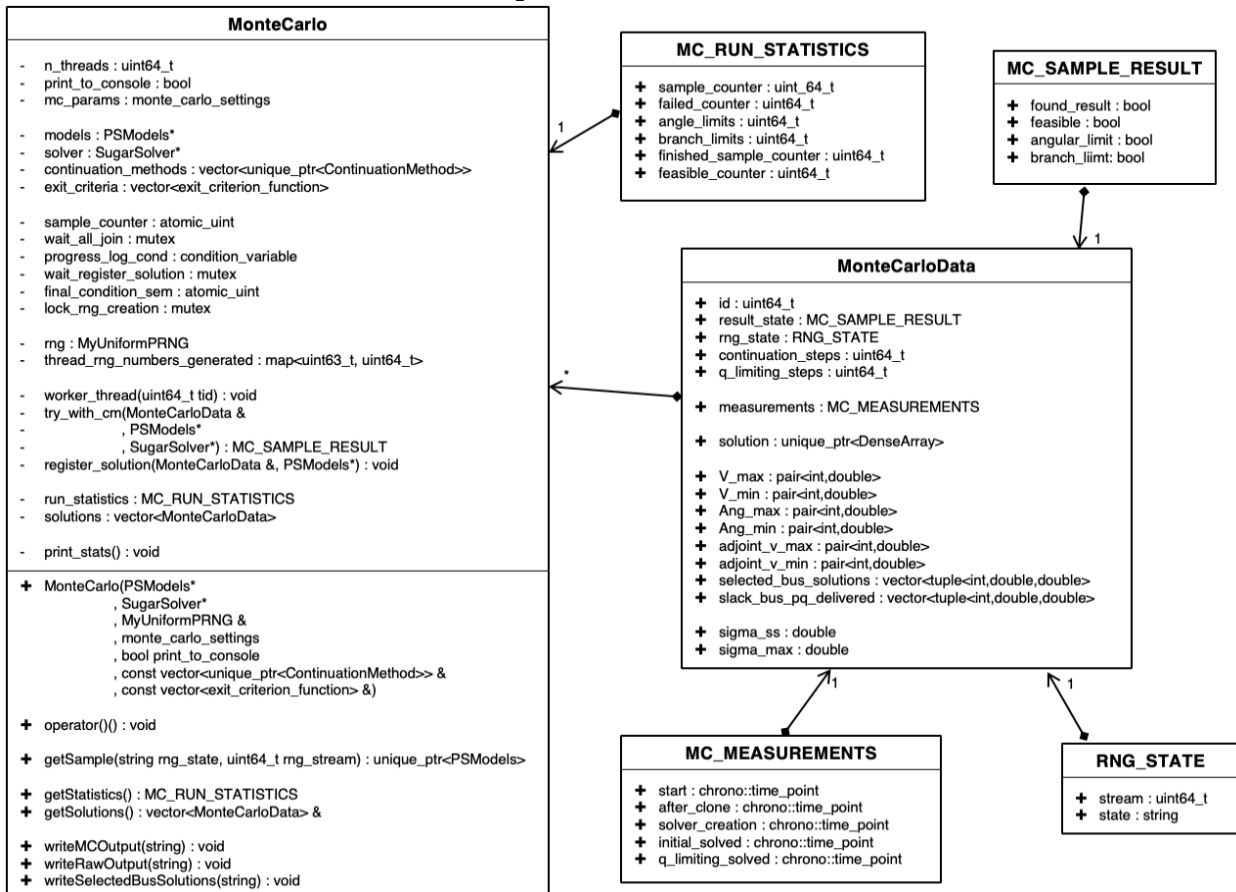


Figure 78: UML class diagram of the parallel *MonteCarlo* solver and related data structures.

Figure 78 shows a UML class diagram of the *MonteCarlo* solver and its related data structures. One can see that the *MonteCarlo* class implements synchronization using mutexes and atomic counters that partly serve as semaphores. Additionally, a condition variable is used to stall and reawake the main thread.

A vector of *MonteCarloData* objects serves as common solution store. In addition to the raw solution vector, multiple statistics of a simulation are stored. Importantly,

performance timing measurements that are stored in the MC_MEASUREMENTS struct introduce substantial overhead and thus can be enabled and disabled during compile time.

Each ECF model that implements probabilistic variables inherits from the *Probabilistic* interface that defines functions for handling probabilistic model. Furthermore, each probabilistic variable is defined using the pointer to a *Distribution_I* interface that can include different distributions. Currently, Normal and Uniform distributions are implemented in this framework. Importantly, each model implements copy and move constructors and a randomize function for probabilistic sample creation.

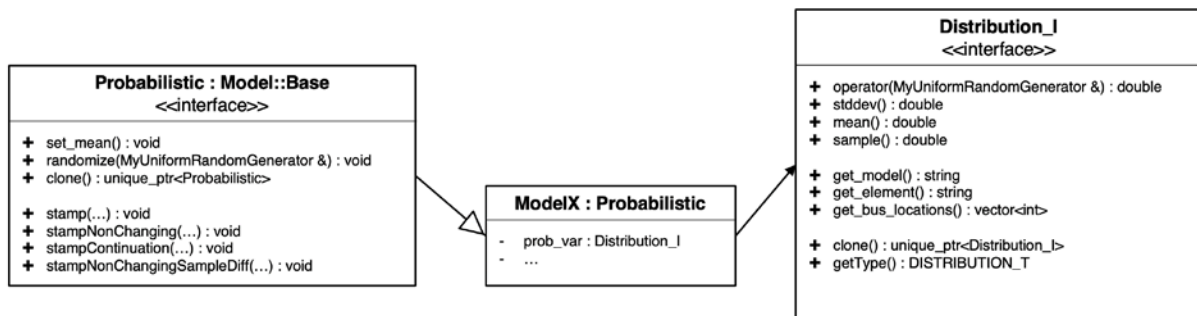


Figure 79: UML class diagrams of the Probabilistic model interface and the class interface for defining and sampling a variable's distribution

Figure 79 gives an overview of these interfaces. Importantly, additional functions of the *Distribution_I* interface enable the identification of model, element, and location data that is connected to the distribution object's variable. This information is crucial to create inter-model correlations.

5.2.4 Probabilistic Input Data

The SUGAR Monte Carlo framework reads probabilistic input data in json-format. Variable's uncertainties can be defined on a global, zonal, area, or bus level.

```

{
  "probabilistic": {
    "global": {
      "load": {
        "P": {
          "distribution": "normal",
          "stddev": 0.1
        }
      },
      "zonal": [
        {
          "id": 1,
          "xfmr": { }
        }
      ],
      "area": [],
      "bus": []
    }
  }
}
  
```

Figure 80: Example of json-based probabilistic input.

Figure 80 shows an example of a json-based probabilistic input. It shows the global definition of normally distributed real power loads with a standard deviation of 0.1 of their nominal values. Additionally, it defines network uncertainties for transformers in area 1. Uncertainties can be defined for each variable that has a mapping to a json-element defined.

A model calls helper functions that implement search algorithms for matching uncertainty definitions. These functions instantiate a probabilistic variable and return it to the model. Importantly, in this search more local definitions are prioritized over global ones.

5.2.5 Used Libraries

The only third-party library that was used for the implementation of this Monte Carlo framework was a pseudo random number generator. The importance of a fast, high-quality random number generator cannot be understated. We utilize the PCG RNG [95] library for that purpose. More precisely, we implement an interface to the “pcg32” RNG type.

5.2.6 Correlations within Monte Carlo Simulations

Two ways to include variable correlations into numerical simulations exist. If variable interactions are assumed to be linear, uncorrelated random numbers are transformed into correlated ones by multiplication with a Cholesky factorized correlation matrix. This is our current approach.

Alternatively, nonlinear variable interactions can be defined via Copula functions. However, this increases modeling complexity substantially. Many different copulas with multiple parameters exist. Defining realistic model parameters for these functions becomes increasingly difficult for these functions without a substantial amount of available data. A good overview of copula functions used in power systems modeling can be found in [110].

5.2.6.1 Design Considerations and Implementation

We distinguish between two types of correlations that are defined using slightly different information. Correlations of variables within models can be defined within the uncertainty definitions of the models. Since there are no interactions between models, local correlations are parsed and instantiated in a straight-forward manner as part of a model.

```

"branch": {
  "Gser": {
    "distribution": "normal",
    "stddev": 0.1,
    "correlation": {
      "Bser": 0.1
    }
  },
  "Bser": {
    "distribution": "normal",
    "stddev": 0.1
  }
}

```

Figure 81: Branch model that defines a localized correlation between its parameters “Gser” and “Bser”.

Figure 81 shows an intra-model correlation defined for variables of a branch model.

Alternatively, correlations between models can be defined. These correlations define interactions of a single, or two types of variables across the whole system, within a zone, or area.

```

"inter-model-correlations":
[
  {
    "from": {
      "model": "gen:wind",
      "element": "P",
    },
    "to": {
      "model": "gen:wind",
      "element": "P",
    },
    "value": 0.8,
    "exp-decay": 0.5,
    "power-decay": 0.0
  }
]

```

Figure 82: Defines wind power generation correlations between generators in a system.

Figure 82 defines positive wind power correlations across a system. Importantly, the “to” element can identify a different variable. For example, negative correlations between wind power availability and PV generation could be defined using two different elements. In addition to defining the base value of a correlation, we define parameters for functions that define the weakening of correlations with increased distance between variables. We can define exponential or simple power laws for this decay and approximate the distance between variables by their shortest path distance on the system graph.

This information is sufficient to create a global correlation matrix for a system. Each distribution is assigned a matrix row that is defined by a distribution’s position in a vector

of all distributions in a system that is initially created. Now, creating correlation matrix entries for localized correlations between two variables is implemented with a data-pass through all models. We simply use an interface function that returns correlations of probabilistic variables within the models.

Creating correlations between variables requires a different mechanism. To avoid multiple model-data passes for the search for matching variables, we add functions that return identifying information such as a probabilistic variable's element and parameter name directly into the *I_Distribution* interface. Now, a search through the vector of all system distributions suffices to create globally correlated variables. In addition, information about a distribution's local bus is available, which enables us to implement correlation-weakening effects with increasing distance between variables.

A system's correlation matrix does not change between MC samples and is reused within our framework. However, it is stored in a Cholesky factorized form to enable direct multiplication to transform uncorrelated samples to correlated ones. Importantly, distributions have to be shifted to a zero mean before creating correlated samples and shifted to their original positions afterwards.

While creating uncorrelated samples is possible within a single data pass, creating correlated random samples requires a three-step process. First, random samples for each variable are drawn. These random samples are stored in a vector of system distributions. A reference to this vector is passed to the *MyCorrelatedRNG* class (Figure 83). In a second step, correlated samples are created by multiplication of previously drawn uncorrelated samples with the Cholesky factorized correlation matrix. Lastly, the correlated random variables are reassigned to the distributions. Because only reference information was passed to the *MyCorrelatedRNG*, the PS models now see this information and a correlated system sample is created.

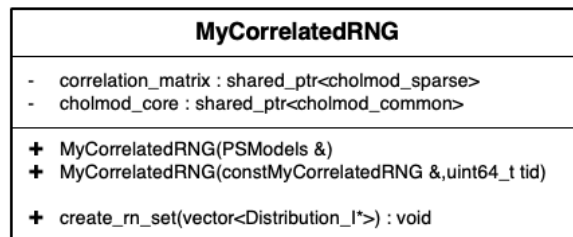


Figure 83: Class diagram of the correlated random number generator.

5.2.6.2 Third-Party Libraries

An additional third-party library is required to calculate the Cholesky factorization. We use the CHOLMOD library [111] that is part of *SuiteSparse* for this purpose.

5.2.7 Memory Profile of the Monte Carlo Framework

In contrast to the detailed memory profile in section 5.1.9.3 we employ a method with less overhead to study memory scaling properties of our MC framework. We investigate

how the maximum memory use changes with increasing parallelism for different size systems using the Linux standard tool “time -v”.

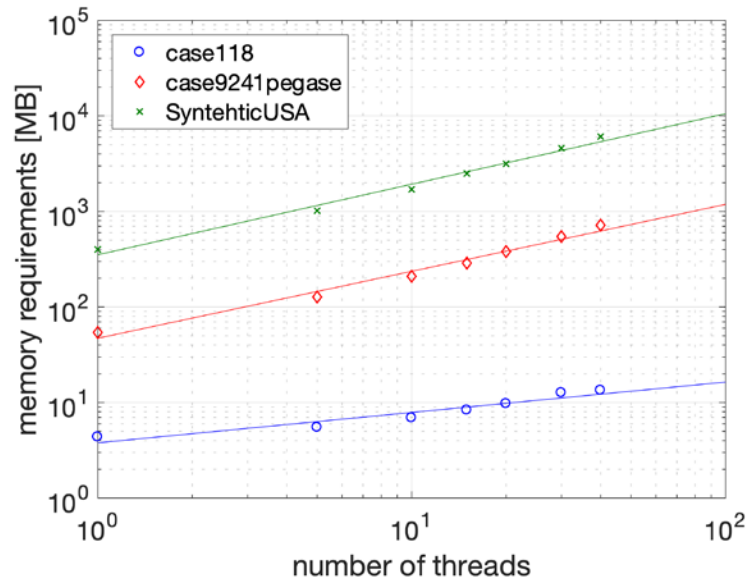


Figure 84: Memory scaling properties of the MC framework for three different size system with additional parallelism.

Figure 84 shows how additional parallelism increases memory requirements for a 118-bus system, a 9241-bus system and an 80k-bus system. Looking at the trendlines for each system we find sub-linear scaling with exponents of .32, .70, and .74, respectively.

6.0 CONCLUSIONS

The U.S. Department of Energy's Quadrennial Technology Review [87] underscores the need for "high-fidelity planning models, tools, and simulators and a common framework for modeling, including databases" for the future grid of tomorrow. Towards the direction of the common goal set forth in that report, this project developed a generic framework for modeling both the transmission as well as the distribution grid including novel methods that can solve robustly for the steady-state operating point of these network models. This is a significant advancement over state-of-art tools used today that use disparate tools and methods for transmission and distribution grid analysis and often require a good initial guess for obtaining the steady-state operating point of the grid. Within the scope of this project, to demonstrate the efficacy of the proposed framework while also validating the methodology, we developed a power system analyses tool SUGAR (Simulation with Unified Grid Analyses and Renewables) that can solve for the steady-state operating point of any transmission or distribution network from arbitrary initial conditions.

Appendix A. BIG Model: Linear Model for Aggregated Load in the Power Grid

A.1 Background

In traditional power flow and three-phase power flow analyses, non-linear models are used to characterize the aggregated electric load behavior. These models are often insensitive to system voltages (as in the case of PQ load model) and introduce strong non-linearities in the formulation (as in the case of PQ and ZIP load model). Interestingly, some of the challenges in terms of solution accuracy and convergence for both these analyses are often due to these models that are used to mimic the behavior of aggregated electric load in the system. For instance, consider the B.C. Hydro system wherein it was shown that decreasing the substation voltage by 1% decreased the active and reactive power demand by 1.5% and 3.4%, respectively [91]. PQ load models that make up the bulk of the aggregated load models today are purely based on constant power variables and are independent of the complex voltage magnitude or angle at the connected node and therefore cannot replicate this behavior. Improvements to the PQ load models (e.g. ZIP model, exponential) can better characterize the voltage sensitive load behavior by incorporating the voltage magnitude dependency; however, like in the case of the PQ load model, they introduce significant non-linearities in the formulation. Furthermore, ZIP and exponential load models cannot characterize load characteristics on a constant voltage node in the system (e.g. variable load connected to a generator node) as the load magnitudes for these load models are independent of the voltage angle information at the connected node. To address these drawbacks, we developed a linear load model [52]-[55] that can capture the true voltage sensitivities for the aggregated load in the system.

To develop a load model that better characterizes the true physics of the grid, we began with understanding the electric load behavior for the Carnegie Mellon University (CMU) campus by observing the measurement data for a randomly chosen 48-hour period as shown in Figure A-1. We can infer from the figure that the load current variation (I_R and I_I) can be attributed to two factors: 1) system voltage variation and 2) variation in actual load demand (i.e. devices turning on and off).

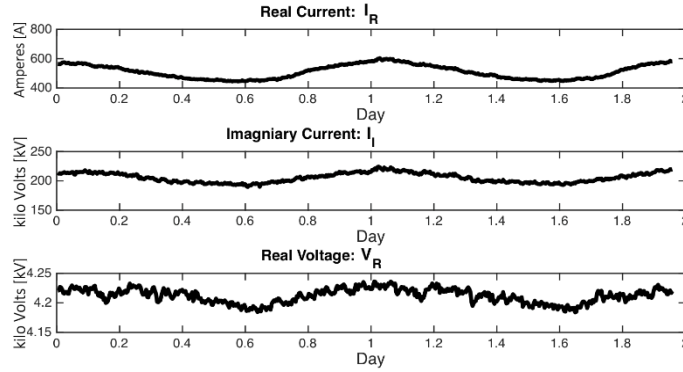


Figure A-1: CMU dataset - current (real and imaginary), and voltage over time (2 days).

Therefore, to accurately capture the load behavior, the developed load model template needs to characterize the base load as well as the voltage sensitivities of the base load correctly. We developed a circuit theoretic model called *BIG* load model in [52]-[53] to achieve exactly that. Furthermore in [54]-[55], we demonstrated the use of machine learning algorithms for fitting this *BIG* load model template. In the following sub-section, we will discuss the development of this circuit theoretic load model, following which we explore the major advantages of the proposed *BIG* load model.

A.2 Circuit Theoretic *BIG* Load Model

A first-order impedance model can be used to represent any phase and magnitude relationship between current and voltage phasors at a single frequency as shown in Figure A-2. This first-order load impedance can be represented as an equivalent circuit model via a conductance (G) and susceptance (B) in series or in parallel, and as such, would capture the load behavior wherein the current flowing into the load bus is directly proportional to the voltage across it. However, the aggregated loads can sometimes behave contrary to this behavior; for example, consider an aggregated load with a large percentage of induction motors that run to maintain a constant mechanical torque. Such loads are likely to exhibit a behavior wherein the current flowing into the load bus is inversely proportional to the applied voltage. This behavior is like that of a constant PQ load model, where the increase in voltage has no influence on the constant power P and would conceptually correspond to a decrease in current.

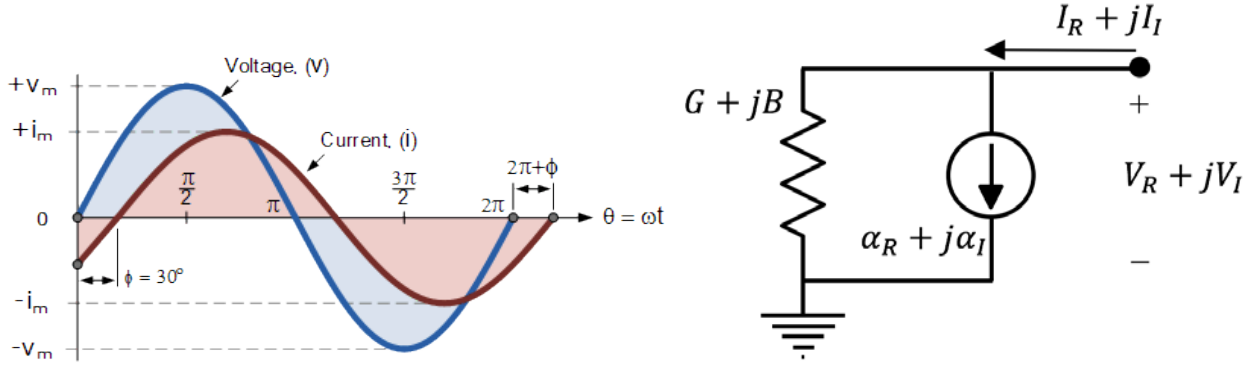


Figure A-2: First-order load model.

To begin from a circuit modeling perspective, we consider a load model for a generalized aggregated load in the system that could capture both positive and negative voltage sensitivities for the load current.

First, we derive the circuit to capture negative sensitivities of load currents with respect to system voltages. To do so, let us consider the governing equation for the PQ load model, which has negative sensitivities of load currents with respect to system voltages:

$$I_R^{PQ} + jI_I^{PQ} = \frac{PV_R + QV_I}{V_R^2 + V_I^2} + j \frac{PV_I - QV_R}{V_R^2 + V_I^2} \quad (118)$$

We can split the complex current function in (118) and linearize it to obtain the real and imaginary terms:

$$I_R^{PQ^{k+1}} = 2I_R^{PQ^k} + \frac{\partial I_R^{PQ}}{\partial V_R} V_R^{k+1} + \frac{\partial I_R^{PQ}}{\partial V_I} V_I^{k+1} \quad (119)$$

$$I_I^{PQ^{k+1}} = 2I_I^{PQ^k} + \frac{\partial I_I^{PQ}}{\partial V_R} V_R^{k+1} + \frac{\partial I_I^{PQ}}{\partial V_I} V_I^{k+1} \quad (120)$$

where the constant terms represent the values of real and imaginary currents known from k^{th} iteration and are represented by a constant current source. Note that partial derivatives for which the real and imaginary currents are directly proportional to the voltages across the respective split circuit models, i.e. real and imaginary, are represented as a conductance (G), while the partial derivatives for which real and imaginary currents are directly proportional to the voltages of other sub circuits are represented by a voltage controlled current source (B).

Furthermore, it can be shown that the respective partial derivatives defined in (119) and (120) have the following properties that further represent the negative sensitivities:

$$\frac{\partial I_R^{PQ}}{\partial V_R} = \frac{\partial I_I^{PQ}}{\partial V_I} \equiv G < 0 \quad (121)$$

$$\left| \frac{\partial I_R^{PQ}}{\partial V_I} \right| = \left| \frac{\partial I_I^{PQ}}{\partial V_R} \right| \equiv B \quad (122)$$

From (119) and (120) we can observe that the governing equation of a PQ load model, i.e. (118), can be translated to an equivalent circuit corresponding to a constant current source (I) in parallel with the susceptance (B) and a negative conductance (G) that compensates for the inverse relationship between the load current and voltage. With this model, as the voltage across the load increases, the current will decrease and vice versa. This model is now extended to even capture positive sensitivities of the load current to voltage change.

To capture both load type sensitivities with respect to voltage, we consider the complex governing equation of the generalized load current that is given by:

$$I_R + jI_I = \alpha_R + j\alpha_I + (V_R + jV_I)(G + jB) \quad (123)$$

where the complex admittance ($G + jB$) with positive G captures the constant impedance load behavior and is directly proportional to the voltage across the load, and the combined impedances capture the voltage sensitivities. Specifically, a negative conductance in conjunction with complex current ($\alpha_R + j\alpha_I$) will mimic the inverse current/voltage sensitivity relationship and positive conductance will represent the other. Both the positive and negative impedances capture the change in load due to the voltage variation with respect to the portion of the load that is parameterized by the current source.

The complex equivalent circuit and the split-circuit of the proposed susceptance (B), current source (I), and conductance (G) load model, BIG , defined by equations (124)-(125), is shown in Figure A-3.

$$I_R = \alpha_R + V_R G - V_I B \quad (124)$$

$$I_I = \alpha_I + V_I G + V_R B \quad (125)$$

It is worth noting that the BIG model is equivalent to the ZIP load model with the real power coefficient set to zero and a different “fixed complex current” term. Most importantly, the BIG load model is linear in a current/voltage formulation, while the ZIP model is nonlinear in both current-voltage and traditional PQV formulations. In addition, the BIG model can capture dependency of the load current with respect to the voltage angle.

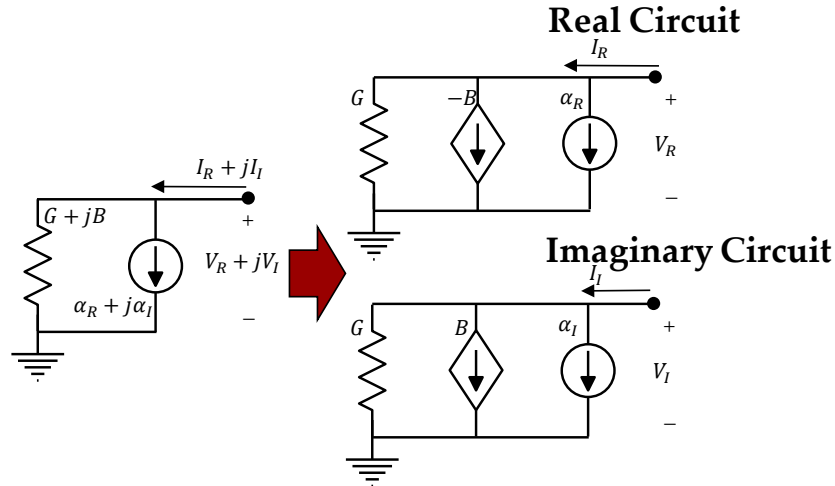


Figure A-3: *BIG* load model.

A.3 Contributions of *BIG* Load Model

The *BIG* load model has many advantages over existing aggregated load models. Among these, the four that are most impactful are as follows:

A.3.1 Linearity

The *BIG* load model is a linear load model as shown in [52] and results in linear network constraints for the network in the equivalent circuit approach. This contrasts with non-linear PQ and ZIP load models. The use of the *BIG* load model significantly reduces the network non-linearities in the proposed equivalent circuit framework thereby significantly reducing the probability of the solver to result in divergence or convergence to an erroneous solution. Importantly the *BIG* load model still results in non-linear network constraints for power flow and three-phase power flow analyses with the use of the 'PQV' formulation and therefore does not extend the same benefits as it does in the case of the equivalent circuit framework.

A.3.2 Captures Voltage Sensitivities

The *BIG* load model when parameterized using real measurement data captures the true sensitivities of the load currents (as a function of system voltage) at the given operating point. This further enables the use of linearized equivalent circuits for any further analysis around the operating point. Accurate sensitivity analysis with a linearized system requires the system sensitivities to be accurately defined as is done via the *BIG* load model. Figure A-4 shows the measured real and imaginary currents for the Carnegie Mellon campus and the constructed currents using the *BIG* load model segments [55]. In the figure, the *BIG* model load model is shown to capture the true

sensitivity of the measured load currents in the system. This contrasts with all existing aggregated load models in the literature.

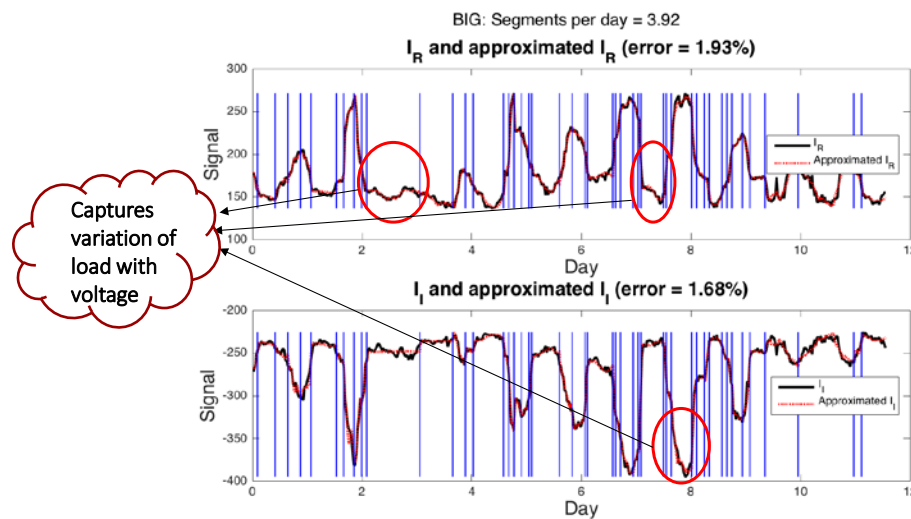


Figure A-4: *BIG* load model is shown to capture the voltage sensitivities if the measured CMU system load [92].

Another experiment that demonstrates the ability of the *BIG* load model to capture true voltage sensitivities is further discussed. In this experiment, a synthetic test grid is setup that includes physics-based load models for IMs, resistive heater load, and capacitors banks. Synthetic measurement data is then generated from this synthetic grid for a number of days with a range of voltage inputs. The *BIG* load model and *PQ* load model parameters are then fitted based on the synthetic measurement data using the same methods as described in [54]-[55]. Now to evaluate and further validate the sensitivity of the fitted load model to the system voltages, the source voltages are decreased and increased by 5% respectively, and new synthetic measurements are produced from the test grid. As it is seen in Figure A-5, the previously fitted *BIG* load models can still capture the load characteristics accurately with perturbed voltages whereas the *PQ* load model results in erroneous currents.

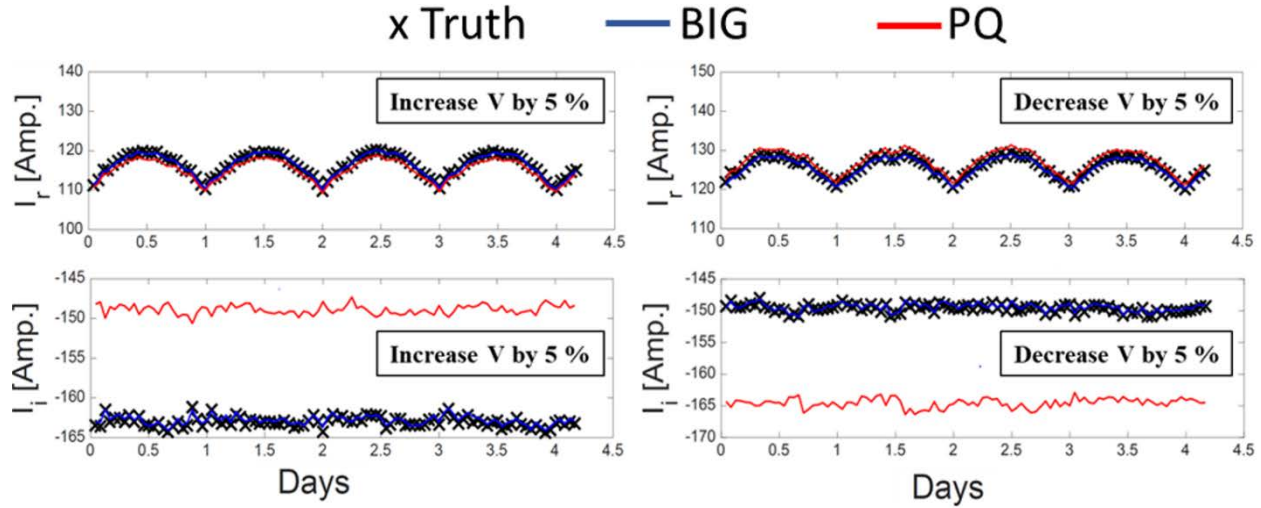


Figure A-5: Fitted *BIG* versus PQ load model with perturbed system voltages [92].

A.3.3 Captures Voltage Angle Information

Unlike PQ and ZIP load models that are either fixed in magnitude or voltage magnitude dependent only, the *BIG* load models are sensitive to both the voltage magnitude and voltage angle. This allows for modeling of aggregated load that is sensitive to only voltage angle i.e. complex voltage dependent load connected to constant voltage magnitude (PV buses) nodes in the system. To illustrate further, as an example behavior of such an aggregated load is shown in Figure A-6. The figure illustrates complex voltage characteristics of a PV node in a power flow case study where the variable load connected to the PV node is varied from 100 MW to 650 MW. The figure shows that the real power absorbed by the PV node is independent of the voltage magnitude at a given node and can be represented as a function of voltage angle (with respect to the reference) only. Therefore, existing load models such constant PQ and voltage magnitude dependent ZIP model will fail to model this behavior, whereas the proposed *BIG* load model will be able to accurately model it.

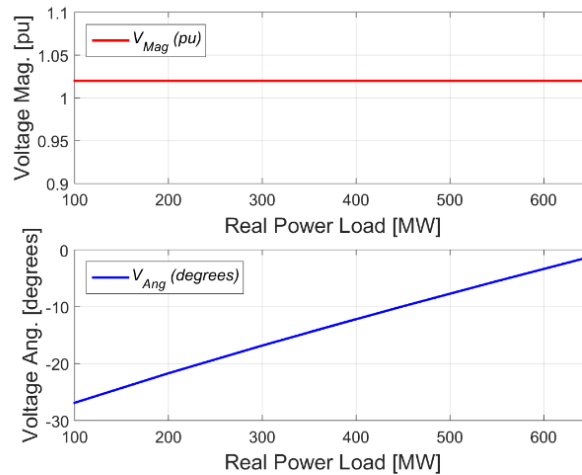


Figure A-6: Complex voltage profile on PV node with variable real power injection.

A.3.4 Generic Model for both Power-flow and Transient Analyses

Importantly, BIG load model unlike ZIP and PQ load model can be used for both time-domain transient analysis as well as frequency domain power flow and three-phase power flow analyses without any modification; hence further enabling the unification of the two analyses as discussed in Appendix B and in [50]. PQ and ZIP load models cannot be directly used for time-domain transient analysis due to the existence of time-average power terms.

Appendix B. Unified Power System Analyses and Models

B.1 Introduction

One of the key underlying challenges in the existing power system analyses is the lack of consistent solutions between the steady-state analysis (power flow and three-phase power flow) and the steady state obtained from time-domain transient analysis. In general, the expectation is for the final steady state of the transient response to match exactly the balanced power flow solution or the three-phase power flow solution. However, this is generally not the case in the existing framework for power system analysis where the nonuse of standardized modeling and solution methods between the steady-state and the transient analyses results in inconsistent and often erroneous results. This contrasts with what's observed in the field of circuit simulation [51], wherein standardization of models and algorithms guarantees consistent solutions between the steady state and time-domain transient analyses for electronic circuits with billion plus nodes.

The most notable cause for discrepancy in solutions between the steady-state and the transient analyses is the use of disparate models for aggregated loads and generators in the two analyses. In steady-state power flow and three-phase power flow analyses, power variables with time average magnitudes and phasor relationships are used to model the aggregated loads as well as the generators (PV/PQ models). These models are inherently incompatible with time domain analysis, where instead physics-based models or some form of approximation of the constant power models (e.g. constant impedance/current) are used to represent the same; hence, resulting in inconsistent solutions between the two analyses. To enable consistent solutions between the two analyses requires that either the network models are derived from the true physics of the equipment or are aggregated such that they can be represented in terms of voltages and currents in both time-domain analysis (transient or dynamic analysis) as well as frequency-domain analysis (for power flow and three-power flow) without loss of generality.

In the past, the use of real and reactive power variables to model the aggregated load and generation for positive sequence and three-phase power flow was necessary due to the inability of the exiting frameworks to include physics-based models directly and due to the lack of online measurement data for the power grid that could characterize the load behavior in terms of currents and voltages. However, real-time access to voltage and current measurements has significantly increased due to high penetration of Remote Terminal Units (RTUs) and Phasor Measurement Units (PMUs) in the grid. This has allowed for characterization of aggregated load using real measurement data with voltage and current as unknown variables as shown in the case of the *BIG* load model in the previous Appendix A. In this appendix as an alternative approach, we explore the

use of physics-based models that can be generically used in both the power flow as well as the time-domain transient analyses. Importantly, both these approaches: i) measurement-based empirical modeling (in Appendix A) and physics-based modeling (discussed within this appendix) are generically applicable to both the steady-state as well as time-domain transient analyses and hence result in consistent solutions between the two.

To demonstrate the use of physics-based models in the equivalent circuit approach for both the time-domain analysis and power flow analysis we consider a simple model of a three-phase squirrel cage induction motor (IM) developed in Section 4.1.5. This model is used to create a simple test network on which we run transient as well as power flow analyses. It is shown that in our approach, using physics-based models, consistent steady-state solution is obtained for both the analyses i.e. transient analysis as well as power flow analysis.

B.2 Validation of IM model

To first validate the physics-based model of the IM that is developed in Section 4.1.5, we run time-domain simulations on the test network, which simulates IM's starting characteristics from stand still to rated speed. We first run this on SUGAR transient analysis module and then compare the obtained results against those produced by the same test setup in MATLAB SimscapePowerSystems (SPS). Figure B-1 shows the response of IM's critical parameters during motor start-up and past that into the steady-state region for both the simulation tools. The evolution of motor state variables over time exhibit similar form and shape when simulated with both the SimscapePowerSystems and the SUGAR thus validating the developed model for IM for both transient and steady-state operating region.

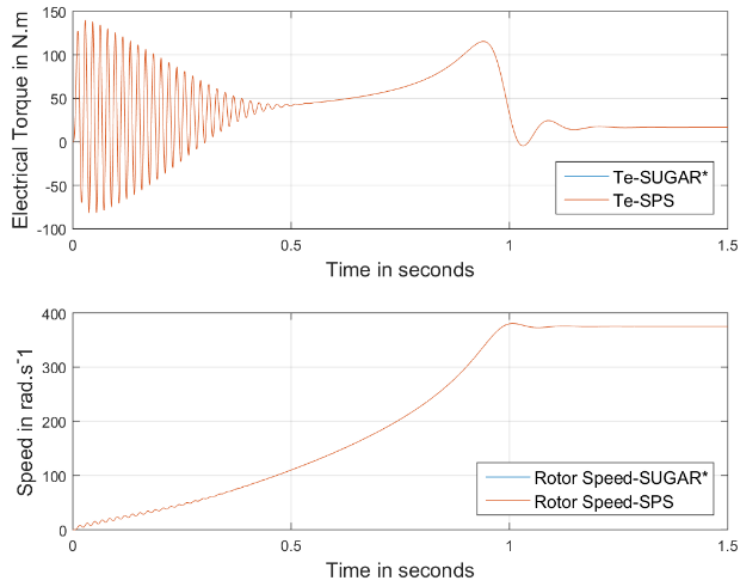


Figure B-1: Electrical torque and rotor speed comparison between SimscapePowerSystems (SPS) and SUGAR with SUGAR restricted to maximum of one N-R iteration.

B.3 Solution consistency between the steady-state and transient analysis

Next, we demonstrate that the simulation of this test network model results in a consistent solution between the steady-state obtained from the transient analysis as well as the steady-state obtained from the power flow analysis. For steady-state power flow as well as transient analyses, the model of IM developed in Section 4.1.5 is used without modifications. In the test network, the IM model is connected to an infinite bus through a transmission line. To run power flow analysis, the source frequency of the test network is set to system frequency. The results from the SUGAR steady state solver are then compared against the ones obtained from the SUGAR transient solver for the same test network. The time-domain transient analysis is run from $t=0$ to an approximate steady state condition at $t=1.5$ seconds as shown in Figure B-1.

Table B-1: IM results in equivalent circuit framework for steady-state (power flow) and time-domain transient analysis

Parameter	Unit	Steady State	Transient @ t=1.5 sec
Rotor Speed	rad.s ⁻¹	375.01	375.01
Electric Torque	N.m	16.64	16.64
Stator direct-axis current	Amps	-11.36	-11.36
Stator quadrature-axis current	Amps	13.09	13.09
Rotor direct-axis current	Amps	11.56	11.56
Rotor quadrature-axis current	Amps	-0.49	-0.49

The comparison of results is documented in Table B-1 and are a perfect match between the two analyses to at least three significant digits. This as a concept demonstrates that equivalent circuit framework can be used in conjunction with physics based models or measurement based empirical models to result in consistent solutions between the steady-state analysis (power flow or three-phase power flow) and the time domain transient analysis.

7.0 REFERENCES

- [1] Quadrennial Energy Review: Energy Transmission, Storage, and Distribution Infrastructure, April 2015.
- [2] "The disaster that could follow from a flash in the sky," *Economist*, July 13th, 2017.
- [3] W. F. Tinney and C. E. Hart, "Power flow solutions by Newton's method," *IEEE Transactions on PAS*, Vol. 86, No. 11, pp. 1449-1460, Nov. 1967.
- [4] D. Shiromohammadi, H. W. Hong, A. Semlyen, G. X. Luo, "A compensation-based power flow method for weakly meshed distribution and transmission networks," *IEEE Transactions on Power Systems*, Vol. 3, No. 2, May 1988.
- [5] P. A. N. Garcia, J. L. R. Pereria, S. Carneiro Jr., M. P. Vinagre, F. V. Gomes, "Improvements in the Representation of PV Buses on Three-Phase Distribution Power Flow," *IEEE Transactions on Power Systems*, Vol. 19, No. 2, April 2004.
- [6] J. Bebic, "Power Systems Operation and Planning Practices and Challenges," *ARPA-E Interconnected- Highly Utilized Grid Workshop*, Sept. 2016.
- [7] S. Grijalva, "Planning and Operational Challenges in Interconnected DER-based Grids," *ARPA-E Interconnected- Highly Utilized Grid Workshop*, Sept. 2016.
- [8] H.W. Dommel, W.F. Tinney, and W.L. Powell, "Further developments in Newton's method for power system applications," *IEEE Winter Power Meeting, Conference Paper No. 70 CP 161-PWR New York*, January 1970.
- [9] W. Murray, T. T. De Rubira, and A. Wigington, "Improving the robustness of Newton-based power flow methods to cope with poor initial conditions," *North American Power Symposium (NAPS)*, 2013.
- [10] A. G. Exposito, E. R. Ramos, "Reliable Load Flow Technique for Radial Distribution Networks," *IEEE Transactions on Power Systems*, Vol. 14, No. 3, August 1999.
- [11] J. Liu, M. M. A. Salama, R. R. Mansour, "An efficient power flow algorithm for distribution systems with polynomial load," *International Journal of Electrical Engineering Education*, 39/4.
- [12] Whei-Min Lin, Tung-Sheng Zhan, Ming-Tong Tsay, "Multiple -Frequency Three-Phase Load Flow for Harmonic Analysis," *IEEE Transactions on Power Systems*, Vol. 19, No. 2, May 2004.
- [13] Mary B. Cain et al., "History of Optimal Power Flow and Formulations," *Federal Energy Regulatory Commission*, December 2012.
- [14] A. Trias, "The holomorphic embedding load flow method," *IEEE PES General Meeting*, 2012.
- [15] S. S. Baghsorkhi and S. P. Suetin. (Mar. 2015). "Embedding AC power flow with voltage control in the complex plane: The case of analytic continuation via Padé approximants."

- [16] M. K. Subramanian, Y. Feng, and D. Tylavsky, "PV bus modeling in a holomorphically embedded power-flow formulation," in *Proc. North Amer. Power Symp. (NAPS)*, Manhattan, KS, USA, Sep. 2013, pp. 1–6.
- [17] I. Wallace, D. Roberts, A. Grothey, and K. I. M. McKinnon. (Jul. 2016). "Alternative PV bus modelling with the holomorphic embedding load flow method."
- [18] M. S. Srinivas, "Distribution load flows: a brief review," in *Proc. IEEE PES Winter Meeting*, Singapore, 2000.
- [19] V. Ajjarapu, C. Christy, "The Continuation Power Flow: The Tool for Steady-State Voltage Stability Analysis," *IEEE Transactions on Power Systems*, Vol. 7, No. 1, February 1992.
- [20] C. Liu, C. Chang, J. A. Jiang and G. H. Yeh, "Toward a CPFLOW-based algorithm to compute all the type-1 load-flow solutions in electric power systems," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 3, pp. 625-630, March 2005.
- [21] Ken Kundert, "Simulation of Analog and Mixed Simulation Circuit," Cadence Design Systems, Inc.
- [22] Jan Ogrodzki, *Circuit Simulation Methods and Algorithms*, CRC Press, May 2018.
- [23] L.W. Nagel and D.O. Pederson, "SPICE: Simulation Program with Integrated Circuit Emphasis," Memorandum No. ERL-M382, Electronics Research Laboratory, University of California Berkeley, 12 April 1973.
- [24] Laurence W. Nagel., "SPICE2: A Computer Program to Simulate Semiconductor Circuits," Memorandum No. ERL-M520, University of California, Berkeley, May 1975.
- [25] McCalla, William J., *Fundamentals of Computer-Aided Circuit Simulation*, Kluwer Academic Publishers, Boston (1988).
- [26] Kundert, Kenneth S., "Sparse Matrix Techniques and their Application to Circuit Simulation," in *Circuit Analysis, Simulation and Design*, ed. Albert E. Ruehli, North-Holland Publishing Co. (1985).
- [27] Saleh, Resve, "Nonlinear Relaxation Algorithms for Circuit Simulation," *Electronics Res. Lab.*, University of California, Berkeley, (April 1987).
- [28] Thomas Linwood Quarles, "Analysis of Performance and Convergence Issues with Circuit Simulation," Memorandum No. ERL-M89/42, University of California, Berkeley, April 1989.
- [29] "IEEE 4-Node Test Feeder," IEEE PES Power System Analysis, Computing, and Economics Committee, Distribution System Analysis Subcommittee.
- [30] L. DeMarco, T. J. Overbye, "Low voltage power flow solutions and their role in exit time -based security measures for voltage collapse," *27th Conference of Decision and Control*, Austin, Texas, Dec. 1988.

- [31] D. Bromberg, M. Jereminov, L. Xin, G. Hug, L. Pileggi, "An Equivalent Circuit Formulation of the Power Flow Problem with Current and Voltage State Variables," *PowerTech Eindhoven*, June 2015.
- [32] M. Jereminov, D. M. Bromberg, A. Pandey, L. Xin, G. Hug, L. Pileggi, "An equivalent circuit formulation for three-phase power flow analysis of distribution systems," *T&D Conference and Exposition*, 2016 IEEE PES.
- [33] M. Jereminov, D. M. Bromberg, L. Xin, G. Hug, L. Pileggi, "Improving Robustness and Modeling Generality for Power Flow Analysis," *T&D Conference and Exposition*, 2016 IEEE PES.
- [34] A. Pandey, M. Jereminov, G. Hug, L. Pileggi, "Improving Power Flow Robustness via Circuit Simulation Methods," *IEEE PES General Meeting*, Chicago, 2017.
- [35] A. Pandey, M. Jereminov, G. Hug, L. Pileggi, "Robust Convergence of Power Flow using Tx Stepping Method with Equivalent Circuit Formulation," *Power Systems Computation Conference*, 2018.
- [36] Vaccaro, A., Cañizares, C. A., and Bhattacharya, K.: "A range arithmetic-based optimization model for power flow analysis under interval uncertainty", *IEEE Transactions on Power Systems*, vol. 28, no 2, p. 1179-1186, 2013.
- [37] Pirnia, M., Cañizares, C. A., and Bhattacharya, K.: "Revisiting the power flow problem based on a mixed complementarity formulation approach", *IET Generation, Transmission & Distribution*, vol. 7, no 11, p. 1194-1201, 2013.
- [38] Sundaresh, L., and Rao, P. N.: "A modified Newton–Raphson load flow scheme for directly including generator reactive power limits using complementarity framework", *Electric power systems research*, vol. 109, p. 45-53, 2014.
- [39] Tapia, R., Fuerte-Esquivel, C. R., Espinosa-Juarez, E., and Sandoval, U.: "Steady-state model of grid-connected photovoltaic generation for power flow analysis", *IEEE Transactions on Power Systems*, vol. 33, no 5, p. 5727-5737, 2018.
- [40] B. Stott, "Review of Load-Flow Calculation Methods," *Proc. IEEE*, vol. 62, no. 7, pp. 916–929, 1974.
- [41] Zhao Jinqun, Jiang Xiaodong, Zhang Boming. "Study on PV-PQ Bus Type Switching Logic in Power Flow Computation," *Proceedings of the CSEE*, Vol. 25, No. 1, pp. 54-59, 2005.
- [42] Y. Kataoka, "A smooth power flow model of electric power system with generator reactive power limits taken into consideration," *Proc. - IEEE Int. Symp. Circuits Syst.*, no. 3, pp. 5286–5289, 2005.
- [43] Turitsyn, K., Sulc, P., Backhaus, S., and Chertkov, M.: "Options for control of reactive power by distributed photovoltaic generators", *Proceedings of the IEEE*, vol. 99, no 6, p. 1063-1073, 2011.

- [44] A. Agarwal, A. Pandey, L. Pileggi, "Continuously Differentiable Analytical Models for Implicit Control within Power Flow," arXiv preprint arXiv:1809.01569, Nov. 2018.
- [45] Russell Kao, "Piecewise Linear Models for Switch-level Simulation," Technical Report: CSL-TR-92-532, Computer Systems Laboratory, Stanford University, June 1992.
- [46] J. Katzenelson, "An algorithm for solving nonlinear resistor networks," in *The Bell System Technical Journal*, vol. 44, no. 8, pp. 1605-1620, Oct. 1965.
- [47] Private communication between Larry Pileggi, Amritanshu Pandey, Barry Rawn from Electrical and Computer Engineering Department in Carnegie Mellon University and Tochi Nwachuku from Nigeria Infrastructure Advisory Facility.
- [48] S. Cvijic, P. Feldmann, M. Ilic, "Applications of homotopy for solving AC power flow and AC optimal power flow," *IEEE PES General Meeting*, San Diego, July 2012.
- [49] Mehta, H. D. Nguyen, K. Turitsyn, "Numerical Polynomial Homotopy Continuation Method to Locate All the Power Flow Solutions," *IET Generation, Transmission and Distribution*, Vol. 10, No. 12, August 16.
- [50] A. Pandey, M. Jereminov, X. Li, G. Hug, L. Pileggi, "Unified Power System Analyses and Models using Equivalent Circuit Formulation," *IEEE PES Innovative Smart Grid Technologies*, Minneapolis, USA, 2016.
- [51] L. Pillage (Pileggi), R. Rohrer, C. Visweswariah, *Electronic Circuit & System Simulation Methods*, McGraw-Hill, Inc., New York, NY, USA, 1995.
- [52] M. Jereminov, A. Pandey, H. A. Song, B. Hooi, C. Faloutsos, L. Pileggi "Linear load model for robust power system analysis," *IEEE PES Innovative Smart Grid Technologies*, Torino Italy, September 2017.
- [53] A. Pandey, M. Jereminov, X. Li, G. Hug, L. Pileggi, "Aggregated Load and Generation Equivalent Circuit Models with Semi-Empirical Data Fitting," *IEEE Green Energy and Systems Conference (IGESC 2016)* November 2016.
- [54] H. A. Song, B. Hooi, M. Jereminov, A. Pandey, L. Pileggi, C. Faloutsos, "PowerCast: Mining and forecasting power grid sequences," *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer 2017.
- [55] B. Hooi, H. A. Song, A. Pandey, M. Jereminov, L. Pileggi, and C. Faloutsos, "StreamCast: Fast and Online Mining of Power Grid Time Sequences.," *Proceedings of the 2018 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics*.
- [56] L. B. Goldgeisser and M. M. Green, "Using Continuation Methods to Improve Convergence of Circuits with High Impedance Nodes," in *IEEE International Symposium on Circuits and Systems*, 2000, Geneva Switzerland.

- [57] Y. Wang, L.C.P. DaSilva, W. Xu., Y. Zhang, "Analysis of ill-conditioned power flow problems using voltage stability methodology," *IEEE Proceedings in Generation, Transmission, and Distribution*, Vol. 148, No. 5, September 2001.
- [58] P.C. Krause and C.H. Thomas, "Simulation of Symmetrical Induction Machinery," in *IEEE Transactions on Power Apparatus and Systems* Vol. PAS-84, No.11, November 1965.
- [59] "Electric Motors use 45% of Global Electricity, Europe Responding," *cleantechnica.com*, June 16th, 2011.
- [60] W. H. Kersting, *Distribution System Modeling and Analysis*, CRC Press, Boca Raton, Florida, USA, 2002.
- [61] R.J. Lee, P. Pillay, and R. G. Harley, "DQ Reference Frames for the Simulation of Induction Motors.," *Electric Power Systems Research*, 8(1984-1985) 15-26.
- [62] Charles L. Fortescue, "Method of Symmetrical Co-Ordinates Applied to the Solution of Polyphase Networks". Presented at the 34th annual convention of the AIEE (American Institute of Electrical Engineers) in Atlantic City, N.J. on 28 June 1918. Published in: *AIEE Transactions*, vol. 37, part II, pages 1027–1140 (1918).
- [63] K. Iba, H. Suzuli, M. Egawa, and T. Watanabe, "Calculation of the critical loading with nose curve using homotopy continuation method," *IEEE Trans. Power Syst.*, vol. 6, no. 2, pp. 584–59, May 1991.
- [64] H. D. Chiang and T. Wang, "Novel Homotopy Theory for Nonlinear Networks and Systems and Its Applications to Electrical Grids," in *IEEE Transactions on Control of Network Systems*.
- [65] H. Sun, Q. Guo, B. Zhang, Y. Guo, Z. Li, and J. Wang, "Master–slave splitting based distributed global power flow method for integrated transmission and distribution analysis," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1484–1492, May 2015.
- [66] K. Kalsi et al., "Integrated transmission and distribution control," Pacific Northwest Nat. Lab., Richland, WA, USA, PNNL22157. [Online]. Available from: http://www.pnnl.gov/main/publications/external/technical_reports/PNNL-22157.pdf.
- [67] K. Anderson, J. Du, A. Narayan, and A. El Gamal, "GridSpice: A distributed simulation platform for the smart grid," in *Proc. 2013 Workshop Modeling Simulation Cyber-Phys. Energy Syst.*, 2013, pp. 1–5.
- [68] B. Palmintier et al., "Integrated distribution-transmission analysis for very high penetration solar PV (final technical report)," Nat. Renew. Energy Lab., Golden, CO, USA, NREL Tech. Rep. NREL/TP-5D00-65550, Jan. 2016.
- [69] Q. Huang, V. Vittal. "Integrated Transmission and Distribution System Power Flow and Dynamic Simulation Using Mixed Three-Sequence/ Three-Phase Modeling," *IEEE Trans. on Power Systems*, vol. 32, no. 5, pp. 3704-3714, Sept. 2017.

- [70] E. L. Allgower and K. Georg, Numerical continuation methods: an introduction, Vol. 13, Springer Science & Business Media, 2012.
- [71] M. Jereminov, D. M. Bromberg, A. Pandey, M. R. Wagner, L. Pileggi, "Adjoint Power Flow Analysis for Evaluating Feasibility," arXiv:1809.01569, Sept. 2018.
- [72] C. Y. Ku, Y. H. Tsai, "Solving Nonlinear Problems with Singular Initial Conditions Using a Perturbed Scalar Homotopy Methods", *International Journal of Nonlinear Sciences and Numerical Simulation*, Vol. 14, Issue. 6, Aug. 2018.
- [73] A. Pandey, M. Jereminov, M. R. Wagner, D. Bromberg, G. Hug, L. Pileggi, "Robust Powerflow and Three-Phase Power Flow Analyses," in *IEEE Transactions on Power Systems*, 2018.
- [74] A. B. Birchfield, E. Schweitzer, H. Athari, T. Xu, T. J. Overbye, A. Scaglione, and Z. Wang, "Validation metrics to assess the realism of synthetic power grids," *Energies*, vol. 10, no. 8, p. 1233, Aug. 2017.
- [75] NERC Standard TOP-001-3 — Transmission Operations.
- [76] K. Schneider, P. Phanivong, J. Lacroix, "IEEE 342-Node Low Voltage Network Test System", *IEEE PES General Meeting Conference*, July 2014.
- [77] "Technical Analysis of Operational Events and Market Impacts During the September 2013 Heat Wave," PJM Interconnection, Dec. 23, 2013.
- [78] Kron G. (1963) Diakoptics: The Piecewise Solution of Large-Scale Systems, MacDonald Publishing.
- [79] N. Rabbat, A. Sangiovanni-Vincentelli AND H. Hsieh, "A multilevel Newton algorithm with macro modeling and latency for the analysis of large-scale nonlinear circuits in the time domain," *IEEE Trans. Circuits and Systems*, CAS-26, pp. 733-741.
- [80] M. Vlach, "LU decomposition and forward-backward substitution of recursive bordered block diagonal matrices," *IEEE Proceedings G, Electronic Circuits and Systems*, vol. 132, no. 1, pp. 24–31, February 1985.
- [81] Chai, J., Zhu, N., Bose, A., & Tylavsky, D. "Parallel newton type methods for power system stability analysis using local and shared memory multiprocessors," *IEEE Trans. Power Syst.*, 6(4), 1539–1545.
- [82] A. R. Newton and A. L. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *IEEE Transactions on Electron Devices*, vol. 30, no. 9, pp. 1184– 1207, September 1983.
- [83] J. M. Ortega and W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, 2000.
- [84] Modern Grid Initiative: Distribution Taxonomy Final Report, *Pacific Northwest National Laboratory*, November 15th, 2008.
- [85] M. Honkula, K. Karanko, and J. Roos, "Improving the convergence of combined newton Raphson and gauss-newton multi-level iteration method," *2002 IEEE*

- International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353), May 2002.*
- [86] P. Li, Parallel Circuit Simulation: A Historical Perspective and Recent Developments, Now Publishers.
 - [87] Quadrennial Technology Review: An Assessment of Energy Technologies and Research Opportunities, Department of Energy, U.S., September 2015.
 - [88] M. LaScala, M. Brucoli, F. Torelli and M. Trovato, "A Gauss-Seidel-block-Newton Method for Parallel Transient Stability Analysis," *IEEE PES Winter Meeting*, Atlanta, February 1990.
 - [89] M. P. Desai, I. N. Hajj, "On the Convergence of Block Time-Point Relaxation Methods for Circuit Simulation," Coordinated Science Laboratory, University of Illinois, Urbana Champaign, Jan. 1988.
 - [90] A. Minot, Y. M. Lu and N. Li, "A Distributed Gauss-Newton Method for Power System State Estimation," in *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3804-3815, Sept. 2016.
 - [91] J. R. Marti, H. Ahmadi, and L. Bashualdo, "Linear Power Flow Formulation Based on a Voltage-Dependent Load Model," *IEEE Trans. Power Delivery*, vol. 28, issue: 3, April 2013.
 - [92] Private communication between Bryan Hooi, Machine Learning Department, Carnegie Mellon University and Amritanshu Pandey, Electrical and Computer Engineering Department, Carnegie Mellon University.
 - [93] Yousef Saad and Maria Sosonkina, "Distributed Schur Complement Techniques for General Sparse Linear Systems," *SIAM Journal on Scientific Computing* 1999 21:4, 1337-1356.

8.0 LIST OF ACRONYMS

$\mathcal{N}$	Set of buses in the system. Each bus in three-phase power flow problem further consist of three distinct phases.
$\mathcal{G}$	Set of generators in the system.
$\mathcal{L}$	Set of loads in the system.
$\mathcal{T}_X$	Set of transmission lines in the system.
$xfmrs$	Set of transformers in the system.
$\mathcal{Z}$	Set of slack buses in the system.
i, l	$\{1, 2, 3, \dots, \mathcal{N} \}$
G	$\{1, 2, 3, \dots, \mathcal{G} \}$
L	$\{1, 2, 3, \dots, \mathcal{L} \}$
C	$\{R, I\}$
R	Real part of the complex variable.
I	Imaginary part of the complex variable.
k	k^{th} iteration of the Newton-Raphson.
$\{A, a\}$	Correspond to phase A.
$\{B, b\}$	Correspond to phase B.
$\{C, c\}$	Correspond to phase C.
$\{N, n\}$	Correspond to phase N.
Ω_{set}	$\{a, b, c, A, B, C\}$
Ω, t	$\in \Omega_{set}$
$P_G^i + jQ_G^i$	Connected complex power generation at bus i .
$P_L^i + jQ_L^i$	Connected complex power demand at bus i .
$G_{il}^Y + jB_{il}^Y$	Is the bus admittance between buses i and l in the positive sequence Y-matrix.
P_G^{max}, P_G^{min}	Maximum and minimum allowable real power generation for generator G .
Q_G^{max}, Q_G^{min}	Maximum and minimum allowable reactive power generation for generator G .
κ	Vector of participation factors for generators participating in AGC or droop control.
θ_{il}	Is the voltage angle between buses i and l .
$\tilde{V}_i$	Complex voltage at bus i .
V_{Ri}, V_{Ii}	Real and Imaginary part of the complex voltage at bus i .
I_{Ri}, I_{Ii}	Real and Imaginary part of the complex current flowing in a branch connected to bus i .
$V_{Ri}^\Omega, V_{Ii}^\Omega$	Real and Imaginary part of the complex voltage of the phase Ω at bus i .
$\mathcal{O}$	Node in the system with a voltage controlling devices connected to it.
$\mathcal{R}$	Remote node controlled by a voltage controlling device on node $\mathcal{O}$.
$\mathcal{W}$	Controlled node where $\mathcal{W} \in \{\mathcal{O}, \mathcal{R}\}$.

$(P_i^{sp} + j Q_i^{sp})^\Omega$	Specified active and reactive power at bus i for the given phase Ω .
tr	Transformer turns ratio.
θ	Transformer phase shift.
φ	Firing angle for the FACTS device.
$\{Z_P^\Omega, I_P^\Omega, S_P^\Omega, Z_Q^\Omega, I_Q^\Omega, S_Q^\Omega\}$	ZIP load parameters for the phase Ω .
$\{G^{BIG}, B^{BIG}, \alpha_R^{BIG}, \alpha_I^{BIG}\}$	BIG load model parameters.
$\mathbf{x}$	Represents a series element between buses i and l .
sh	Represents a shunt element connected at node i .
$G_{\Omega t}^{\mathbf{x}} + jB_{\Omega t}^{\mathbf{x}}$	Self and mutual admittance of the line between phases t and Ω .
$G_i^{sh} + jB_i^{sh}$	Shunt admittance connected to node i .
ς	Variable limiting factor.
λ	Homotopy factor for Tx Stepping Method. Lies in the closed set $[0, 1]$.
γ	Scaling factor for Tx Stepping Method.
$\{V_C^0, V_C^1, V_C^2\}$	Represents the zero, positive, and negative sequence voltages for the set C .
$\{I_C^0, I_C^1, I_C^2\}$	Represents the zero, positive, and negative sequence currents for the set C .
J	Jacobian of the positive sequence or three-phase power flow solution matrix.
c	Continuous curve for the homotopy methods in the domain of homotopy factor λ .
$\{\mathbf{V}_R^{int}, \mathbf{V}_I^{int}\}$	Internal nodes real and imaginary voltages vector for the sub-circuit.
$\{\mathbf{V}_R^{ext}, \mathbf{V}_I^{ext}\}$	External nodes real and imaginary voltages vector for the sub-circuit.
N-2	Loss of two equipment in the system.
N-3	Loss of three equipment in the system.
T	Sub-circuit in the combined T&D system representing the transmission grid.
D_x	Sub-circuit in the combined T&D system representing the distribution grid x amongst the set of distribution grids.
$\mathcal{F}_T$	System equations for the transmission grid within the combined T&D problem.
$\mathcal{F}_{D_x}$	System equations for the distribution grid x within the combined T&D problem.
$\mathcal{F}_C$	System equations for the coupling network between the transmission and distribution grid in the combined T&D problem.