2024 IEEE Texas Power and Energy Conference (TPEC) | 979-8-3503-3120-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/TPEC60005.2024.10472209

Enhancing ACPF Analysis: Integrating Newton-Raphson Method with Gradient Descent and Computational Graphs

Masoud Barati *Senior Member, IEEE*

***Abstract*—This manuscript presents a novel approach utilizing computational graph strategies for solving the power flow equations through the synergistic use of Newton-Raphson (NR) and Gradient Descent (GD). As a foundational element for operational and strategic decision-making in electrical networks, the power flow analysis has been rigorously examined for decades. Conventional solution techniques typically depend on second-order processes, which may falter, especially when faced with subpar starting values or during heightened system demands. These issues are becoming more acute with the dynamic shifts in generation and consumption patterns within modern electrical systems. Our research introduces a dual-mode algorithm that amalgamates the principles of first-order operation. This inventive method is adept at circumventing potential local minima traps that hinder current methodologies, thereby reinforcing the dependability of power flow solutions. We substantiate the effectiveness of our advanced algorithm with comprehensive testing on established IEEE benchmark systems. Our findings reveal that our approach not only expedites the convergence process but also ensures consistent performance across diverse system states, signifying a meaningful progression in the realm of power flow computation.**

***Index Terms*—ACPF analysis, Automation differentiation, Chain rule, Computational graph, Newton-Raphson.**

I. Introduction

The integration of renewable energy sources such as wind and solar power is revolutionizing the power systems landscape, presenting new challenges that stem from their variable and intermittent nature. The once-predictable flow of electricity is now subject to fluctuations, leading to a power grid that is more dynamic and less predictable than ever before. This transformation calls for advanced computational techniques capable of conducting power flow analysis with greater resilience and adaptability.

Traditional power flow analysis methods, like the Newton-Raphson (NR) technique, have provided reliable solutions for decades. However, the NR method is primarily designed for stable and predictable systems and may struggle with the irregularities introduced by renewable sources. This is primarily because the NR method's success hinges on good initial approximations and conditions that remain close to normal operating ranges. As renewable integration intensifies, these conditions are increasingly difficult to guarantee, leading to potential convergence issues and inaccuracies. Power flow analysis, crucial in the power system field, involves solving nonlinear algebraic equations. The Newton-Raphson (NR) method, widely used for its rapid convergence, iteratively updates solutions using the Jacobian's inverse [1], [2]. However, this method faces challenges in convergence when initial guesses are far from the final solution or the Jacobian matrix becomes problematic during iterations [3]. Various strategies address these issues, such as augmenting system states [4], exploring polar versus rectangular formulations [5], refining starting points [6], [7], and employing alternate Jacobian approximations [8]–[10]. An innovative approach reformulates power flow as an optimization problem, integrating complementarity constraints for PV buses [11]–[15]. This paper presents a cutting-edge algorithm blending projected gradient descent (GD) and Newton-Raphson (NR) methods, uniquely targeting computational challenges in AC power flow (ACPF) problems. This novel strategy redefines the ACPF problem as an optimization challenge, allowing for gradient descent steps without requiring Jacobian matrix inversion, a limitation of conventional NR techniques. Projected GD, effective in maintaining constraints, does not inherently circumvent local optima and saddle points, typical in deterministic optimization. The algorithm smartly transitions to NR methods for quicker convergence as it approaches the global optimum.

This complex scenario demands a sophisticated solution adaptable to the dynamic power grid environment. An effective answer is the integration of the NR method with GD and computational graphs. This comprehensive approach combines NR's iterative resolution prowess with GD's adaptive learning strengths—celebrated for its effectiveness in complex, high-dimensional domains like machine learning and AI.

Computational graphs represent a further leap in this integrated method. By mapping the intricate relationships of power system variables as a network of nodes and edges, computational graphs offer a clear visualization of the power flow problem. They simplify the application of both NR and GD by providing a framework for systematic calculations and updates to the system's state, facilitating the management of the non-linearities characteristic of modern power grids.

In this context, computational graphs not only serve as a visual aid but as a foundational tool that transforms the power flow analysis into a more flexible and adaptive process. This allows for the systematic application of GD, which

Masoud Barati is with the Department of Electrical and Computer Engineering and Industrial Engineering, Swanson School of Engineering, University of Pittsburgh, PA, USA (email: masoud.barati@pitt.edu). This work was supported by the NSF ECCS Award 1711921.

979-8-3503-3120-2/24/$31.00 ©2024 IEEE

can iteratively adjust the system state by moving against the gradient of the error surface, thus providing a mechanism to overcome the shortcomings of traditional methods.

The convergence of these methods—NR's precision, GD's adaptability, and computational graphs' clarity—creates a powerful toolkit for today's power system analysts. It equips them to tackle the stochastic nature of renewable energy sources and ensures that power flow analysis remains a reliable and insightful process, crucial for the planning and operation of modern, sustainable power systems.

The paper is structured to first outline the ACPF problem (Section II), describe the computational graph algorithm and NR method (Section III), provide numerical simulations for six test case studies and comparison with existing methods (Section IV), and conclude with insights and findings (Section V).

II. Power Flow Equations

In an electrical network with n nodes, each node, indexed as k, possesses a set of electrical properties: a complex voltage including the magnitude voltage V_k, and phase angle θ_k, alongside its associated active P_k and reactive Q_k power components. These properties can be collectively represented in vectorial form as $\mathbf{V} = (V_1, \ldots, V_n), \theta = (\theta_1, \ldots, \theta_n), \mathbf{P} = (P_1, \ldots, P_n)$, and $\mathbf{Q} = (Q_1, \ldots, Q_n)$. The network's admittance matrix is denoted by $\mathbf{Y}$. This allows the encapsulation of the network's power flow equations into a concise notation: $g(\mathbf{V}) = \mathbf{P} + j\mathbf{Q} = \operatorname{diag}\left(\mathbf{V}\mathbf{V}^{\dagger}\mathbf{Y}^{\dagger}\right)$, where $(\cdot)^{\dagger}$ signifies the conjugate transpose operation.

When presented with a complex load vector $\mathbf{s}$, the ACPF calculation seeks the magnitude voltage vector $\mathbf{V}$ and phase angle θ that satisfies $g(\mathbf{V}, \theta) = \mathbf{s}$. Rather than confronting this nonlinear equation head on, an optimization framework is posited for resolution. To address the system of equations $g(\mathbf{V}, \theta) = \mathbf{s}$, we employ an optimization approach aimed at minimizing the error $\epsilon = g(\mathbf{V}, \theta) - \mathbf{s}$. A least-square loss function, which quantifies the error ϵ as $\min_{\mathbf{V},\theta} \frac{1}{2}\|\epsilon\|_2^2$ which is defined in (1).

$$\min_{\mathbf{V},\theta} \frac{1}{2}\|g(\mathbf{V}, \theta) - \mathbf{s}\|_2^2 = \min_{\mathbf{V},\theta} \frac{1}{2}\sum_{i=1}^{n}\left(g_i(\mathbf{V}, \theta) - s_i\right)^2 \quad (1)$$

It is important to clarify that Equation (1) does not represent an optimal power flow (OPF) problem. Rather, to address the issues of infeasibility and solvability in the ACPF analysis, we approach the ACPF by minimizing the least-square error. Consequently, we continue to refer to the problem defined in Equation (1) as a ACPF problem. In scenarios where the ACPF problem is solvable, the ideal outcome for the objective measure equates to zero. Under these circumstances, there exists an optimal voltage vector, denoted as $\mathbf{V}^*, \theta^*$, fulfilling the condition $g\left(\mathbf{V}^*, \theta^*\right) = \mathbf{s}$. Considering the problem's structure as an unconstrained one, characterized by a continuously differentiable objective function, the application of gradient descent emerges as an intuitive method for finding a solution.

For ease of reference, the objective function in equation (1) is represented as $\mathcal{L}$. The gradient of $\mathcal{L}$ concerning $\mathbf{V}, \theta$ is derived using the chain rule and can be expressed as:

$$\nabla_{\mathbf{V},\theta}\mathcal{L} = \mathbf{J}^{\top}(g(\mathbf{V}, \theta) - \mathbf{s}) \quad (2)$$

where $\mathbf{J}$ symbolizes the Jacobian matrix associated with the power flow. The conventional Gradient Descent (GD) formula is then:

$$\mathbf{V}_{t+1} = \mathbf{V}_t - \eta\nabla_{\mathbf{V}}\mathcal{L}\left(\mathbf{V}_t\right) \quad (3)$$

$$\theta_{t+1} = \theta_t - \eta\nabla_{\theta}\mathcal{L}\left(\theta_t\right) \quad (4)$$

with t indicating the iteration stage, and η representing the step size or learning rate, which can be either constant or variable.

The sets of bus indices for the reference bus, PV buses, and PQ buses are denoted as $\mathcal{I}_{\text{ref}}$, $\mathcal{I}_{\text{PV}}$, and $\mathcal{I}_{\text{PQ}}$, respectively. In the context of equation (3) and (4), adjustments are made only to the voltage angles $\delta_{\{i\}}$ for $i \notin \mathcal{I}_{\text{ref}}$ and the voltage magnitudes $v_{\{m\}}$ for $i \in \mathcal{I}_{\text{PQ}}$. This selective updating also allows for the setting of specific voltage magnitudes on PV buses.

From equations (3) and (4), the GD algorithm ceases under two circumstances: 1) the global optimum is achieved with $g(\mathbf{V}, \theta) - \mathbf{s} = 0$, or 2) the Jacobian $\mathbf{J}$ becomes singular, and $g(\mathbf{V}, \theta) - \mathbf{s}$ falls within the null space of $\mathbf{J}^{\top}$.

The latter scenario implies that the iterative values $\mathbf{V}_t$ and θ_t are caught in a local minimum or at a saddle point. To break free from this impasse, the algorithm must diverge from the gradient path (which becomes null) and adopt a different trajectory. The chosen direction for this shift should not be arbitrary but strategically selected. In this paper, we propose a specific approach to guide this directional change.

III. Computational Graph

A. *What is computational graph?*

A computational graph is a network where each node signifies an arithmetic operation. It is a structural representation used to depict and compute mathematical expressions efficiently. Consider the elementary mathematical formula:

$$p = x + y \quad (5)$$

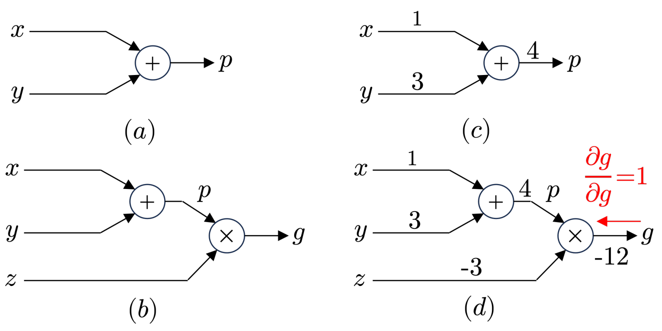

Fig. 1. Computational graph for a simple calculation

The computational graph for the equation is depicted in Fig. 2(a), where a node marked with a "+" sign adds inputs x and y to produce output g ". For a complex example, consider the following equation:

$$g = (x + y) \cdot z. \tag{6}$$

In the computational graph shown in Fig. 2(b), the initial node combines x and y by addition, which then merges with z in a multiplication node to produce the output g.

B. Gradient calculation on computational graphs

In stochastic gradient descent with Newton-Raphson, computational graphs guide each iteration's solution refinement. The forward pass processes inputs sequentially through the graph, moving from initial to terminal nodes, akin to a journey from origin to destination. This involves specific input values progressing through layers and functions at each iteration. For illustrative purposes, let's assign specific values to the inputs as follows: $x = 1$, $y = 3$, $z = -3$, as illustrated in Fig. 2(c) and Fig. 2(d). With these values assigned, executing a forward pass allows the computation of intermediary and final output values at each node. In Fig. 2(c), commencing with the values $x = 1$ and $y = 3$, we calculate the intermediate output $p = 4$. Subsequently in Fig. 2(d), employing $p = 4$ and $z = -3$, we determine the final output $g = -12$. The computation progresses linearly from inputs to outputs, with gradient calculation determining each input's impact on the final output, crucial for refining solutions via gradient descent optimization. Consider the necessity to determine the following gradient values:

$$\frac{\partial x}{\partial f}, \frac{\partial y}{\partial f}, \frac{\partial z}{\partial f} \tag{7}$$

Initiating the backward pass, we calculate the rate of change of the final output in relation to itself, which, by definition, yields a value of one.

$$\frac{\partial g}{\partial g} = 1. \tag{8}$$

Visualizing our computational graph post this step is shown in Fig. 2(d).

Proceeding, we reverse through the multiplication operation. Here, we need to compute the gradient at the nodes p and zm where g is the product of p and z ($p = x + y$ and $g = p \cdot z$).

Using a computational graph in ACPF calculation offers several benefits:

1) Efficient Backpropagation: It allows for automatic differentiation, making the calculation of gradients for optimization algorithms (like gradient descent) more efficient and accurate.
2) Improved Performance: It enables optimization of computational resources and parallel processing, speeding up calculations.
3) Easier Debugging and Visualization: It helps in visualizing and understanding complex operations, which aids in debugging and improving ACPF models in different use cases and test cases.

Not using a computational graph can lead to:

1) Manual Gradient Calculation: This can be error-prone and computationally intensive, especially for complex models.
2) Reduced Efficiency: Without the optimized execution paths that computational graphs provide, computations may be slower and less efficient.
3) Difficulty in Scaling: Manual implementations without computational graphs can be challenging to scale for large power grids.

C. Automatic Differentiation

Our approach will employ automatic differentiation to determine $\partial z/\partial x$ and $\partial z/\partial y$. Initially, we consider a single node defined by the equation $z = z(x, y)$, which is a component of a broader graph culminating in a scalar value, denoted as ℓ. Presuming we have successfully computed $\partial \ell/\partial z$, the task then is to find $\partial \ell/\partial x$ and $\partial \ell/\partial y$. To determine the derivative with respect to x, the following formula is utilized:

$$\frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial z} \cdot \frac{\partial z}{\partial x}$$

Similarly, for the derivative with respect to y, the equation is:

$$\frac{\partial \ell}{\partial y} = \frac{\partial \ell}{\partial z} \cdot \frac{\partial z}{\partial y}$$

It's important to clarify that the expression $\ell(z(x, y))$ may seem slightly confusing. It simply signifies that ℓ is a function of z, which in turn is a function of x and y. In a more complex graph, ℓ could depend on numerous other variables.

In computational graphs, understanding the derivative of the final output ℓ with respect to a node's output allows reverse calculation of ℓ's derivative relative to the node's inputs. This principle enables backpropagation from the final output to initial inputs, a core concept in ACPF calculation.

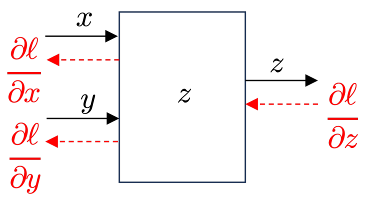
x
y
z
z
$\frac{\partial \ell}{\partial x}$
$\frac{\partial \ell}{\partial y}$
$\frac{\partial \ell}{\partial z}$

Fig. 2. Computational graph illustrating backward differentiation paths for the function $z = z(x, y)$.

D. Gradient Calculation in ACPF

The power flow equations are symbolized as:

$$\underline{g}(\underline{u}, \underline{x}) = \underline{s} \tag{9}$$

It can be rewritten as the following equation:

$$\underline{g}(\underline{u}, \underline{x}) - \underline{s} = \underline{0} \tag{10}$$

Where, the u is the input of the power flow problem; the active power and magnitude voltage of the PV buses; active power and reactive power of the PQ buses. This particular notation reflects the equilibrium of both active and reactive power at each node within an electrical grid. In a network with n nodes, this translates to a total of $2n$ distinct real equations. These equations are formulated as follows:

$$\begin{aligned} P_k(\underline{V},\underline{\theta}) - P_{\text{netk}} &= 0, \quad k=\{1,\cdots,n\} \\ Q_k(\underline{V},\underline{\theta}) - Q_{\text{netk}} &= 0, \quad k=\{1,\cdots,n\} \end{aligned} \tag{11}$$

where, In the framework of bus voltage dynamics, the variables $\underline{V}$ and $\underline{\theta}$ symbolize the magnitude and phase angles of voltages at n distinct nodes respectively. The active and reactive power injections at the k^{th} node are represented by $P_k(\underline{V},\underline{\theta})$ and $Q_k(\underline{V},\underline{\theta})$. These are calculated based on the voltage magnitudes and angles.

The net active and reactive power entering the k^{th} node are denoted as P_{netk} and Q_{netk}. These are derived as the differences between generated power (P_{Gk}, Q_{Gk}) and the power demand (P_{Dk}, Q_{Dk}) at the respective node.

The formulations for the active and reactive power injections at each node are given by:

$$\begin{aligned} P_k(\underline{V},\underline{\theta}) &= V_k \sum_{m\in\{K\}} V_m \left(G_{km}\cos\theta_{km} + B_{km}\sin\theta_{km}\right) \\ Q_k(\underline{V},\underline{\theta}) &= V_k \sum_{m\in\{K\}} V_m \left(G_{km}\sin\theta_{km} - B_{km}\cos\theta_{km}\right) \end{aligned} \tag{12}$$

Here, K refers to the set of nodes adjacent to the k^{th} node. The parameters G_{km} and B_{km} represent the conductance and susceptance of the transmission line between nodes k and m. The term θ_{km} is the angular difference between these nodes.

By reformulating the power flow equation as indicated in equation (12), a more streamlined nested model emerges. This model presents an ideal mathematical structure for the implementation of automatic partial derivative calculations.

$$\begin{aligned} P_k(\underline{p}) &= \sum_{m\in\{K\}} p_{km} \\ Q_k(\underline{q}) &= \sum_{m\in\{K\}} q_{km} \end{aligned} \tag{13}$$

$$\begin{aligned} p_{km}(\underline{c},\underline{s}) &= G_{km}c_{km} + B_{km}s_{km} \\ q_{km}(\underline{c},\underline{s}) &= G_{km}s_{km} - B_{km}c_{km} \end{aligned} \tag{14}$$

$$\begin{aligned} c_{km}(\underline{V},\underline{\theta}) &= V_kV_m\cos\theta_{km} \\ s_{km}(\underline{V},\underline{\theta}) &= V_kV_m\sin\theta_{km} \end{aligned} \tag{15}$$

This revised approach aligns with the computational graph framework and facilitates the automatic computation of gradients. It necessitates the use of a chain rule for the calculation of the Jacobian matrix, integral to gradient determination. This structure is consistent with modern methods of automatic gradient computation, streamlining the process. The power flow equation can be reformulated into a series of nested functions, each dependent on nested variables.

$$\begin{aligned} P &= p(c(V,\theta), s(V,\theta)) \\ Q &= q(c(V,\theta), s(V,\theta)) \end{aligned} \tag{16}$$

The gradient can be determined by applying the chain rule in the following manner:

$$\begin{aligned} \frac{\partial P}{\partial \theta} &= \frac{\partial P}{\partial p}\cdot\frac{\partial p}{\partial c}\cdot\frac{\partial c}{\partial \theta} + \frac{\partial P}{\partial p}\cdot\frac{\partial p}{\partial s}\cdot\frac{\partial s}{\partial \theta} \\ \frac{\partial Q}{\partial \theta} &= \frac{\partial Q}{\partial q}\cdot\frac{\partial q}{\partial c}\cdot\frac{\partial c}{\partial \theta} + \frac{\partial Q}{\partial q}\cdot\frac{\partial q}{\partial s}\cdot\frac{\partial s}{\partial \theta} \\ \frac{\partial P}{\partial V} &= \frac{\partial P}{\partial p}\cdot\frac{\partial p}{\partial c}\cdot\frac{\partial c}{\partial V} + \frac{\partial P}{\partial p}\cdot\frac{\partial p}{\partial s}\cdot\frac{\partial s}{\partial V} \\ \frac{\partial Q}{\partial V} &= \frac{\partial Q}{\partial q}\cdot\frac{\partial q}{\partial c}\cdot\frac{\partial c}{\partial V} + \frac{\partial Q}{\partial q}\cdot\frac{\partial q}{\partial s}\cdot\frac{\partial s}{\partial V} \end{aligned} \tag{17}$$

Transforming equations (13) and (17) through a first-order Taylor series approximation centered at the point $\left(V^0,\underline{\theta}^0\right)$ results in the following equations.

$$\begin{aligned} & P_k\left(\underline{V}^0+\underline{\Delta V},\underline{\theta}^0+\underline{\Delta\theta}\right) \\ & = P_k\left(\underline{V}^0,\underline{\theta}^0\right) + \left[\frac{\partial P_k(\underline{V},\theta)}{\partial\underline{\theta}} \mid \frac{\partial P_k(\underline{V},\theta)}{\partial\underline{V}}\right]_{(\underline{V}^0,\underline{\theta}^0)} \left[\begin{array}{c}\underline{\Delta\theta}\\ \underline{\Delta V}\end{array}\right] \\ & Q_k\left(\underline{V}^0+\underline{\Delta}V,\underline{\theta}^0+\underline{\Delta\theta}\right) \\ & = Q_k\left(\underline{V}^0,\underline{\theta}^0\right) + \left[\frac{\partial Q_k(\underline{V},\theta)}{\partial\underline{\theta}} \mid \frac{\partial Q_k(\underline{V},\theta)}{\partial\underline{V}}\right]_{(\underline{V}^0,\underline{\theta}^0)} \left[\begin{array}{c}\underline{\Delta\theta}\\ \underline{\Delta V}\end{array}\right] \end{aligned} \tag{18}$$

Replacing equations (18) into (11) yields,

$$\begin{aligned} & P_{netk} - P_k\left(\underline{V}^0,\underline{\theta}^0\right) = \Delta P_k \\ & = \left[\frac{\partial P_k(\underline{V},\theta)}{\partial\underline{\theta}} \mid \frac{\partial P_k(\underline{V},\theta)}{\partial\underline{V}}\right]_{(\underline{V}^0,\underline{\theta}^0)} \left[\begin{array}{c}\underline{\Delta\theta}\\ \underline{\Delta V}\end{array}\right] \\ & Q_{netk} - Q_k\left(\underline{V}^0,\underline{\theta}^0\right) = \Delta Q_k \\ & = \left[\frac{\partial Q_k(\underline{\underline{V}},\theta)}{\partial\underline{\theta}} \mid \frac{\partial Q_k(\underline{V},\theta)}{\partial\underline{V}}\right]_{(\underline{V}^0,\underline{\theta}^0)} \left[\begin{array}{c}\underline{\Delta\theta}\\ \underline{\Delta V}\end{array}\right] \end{aligned} \tag{19}$$

From the full array of linearized power balance equations, a specific subset defined by equation (19) is chosen, considering the unique characteristics of each bus in the network. This subset includes $(n-1)$ active power equations, each representing a different bus excluding the slack bus, and an additional NPQ equations for the reactive power at the load buses. Consequently, the fundamental load flow equation is established as follows:

$$\left[\begin{array}{c}\underline{\Delta P}_{PV}\\ \underline{\Delta P}_{PQ}\\ \underline{\Delta Q}_{PQ}\end{array}\right] = J_{LF}\left[\begin{array}{c}\underline{\Delta\theta}_{PV}\\ \underline{\Delta\theta}_{PQ}\\ \underline{\Delta V}_{PQ}\end{array}\right] \tag{20}$$

where,

$$J_{LF} = \left[\begin{array}{ccc} H_{PV,PV} & H_{PV,PQ} & N_{PV,PQ} \\ H_{PQ,PV} & H_{PQ,PQ} & N_{PQ,PQ} \\ J_{PQ,PV} & J_{PQ,PQ} & L_{PQ,PQ} \end{array}\right] \tag{21}$$

IV. Simulation Results

This section presents the numerical results obtained from the implementation of the numerical techniques described in previous sections. The experiments were conducted on Google Cloud Services using the NVIDIA V100 GPU instance for gradient descent calculation. We implemented the ACPF model with Python programming.

TABLE I
SUMMARY OF INITIAL SOLUTION VIOLATIONS AND THEIR MAGNITUDES IN VARIOUS TEST SYSTEMS

Test System	No. of Violations (Init.Sol.)	Magnitude of the Largest Violation (p.u.)
	V^m	$V_{PQ_i}^m$
IEEE14	7	0.0292
IEEE30	18	0.0826
NEGL39	28	0.0603
IEEE57	36	0.0634
PRCT89	56	0.0475
IEEE118	3	0.0070

The tests involve initially executing an ACPF followed by addressing any potential PQ bus voltage magnitude violations in the load flow solution. To provoke these violations, the lower limits of dependent variables were raised, creating a narrowly feasible region. The results, as shown in Tables I and II, indicate the effective reduction of violations through orthogonal projections onto feasible regions, with violations ranging from 3 in the 118 bus system to 56 in the 89 bus system.

Considering the limitations in control variables (32 for the largest system), it's impractical to address all violations simultaneously. Thus, only the most significant violations, concerning bus magnitude limits, are included in the active constraint set. This approach maintains a manageable system size compared to the original problem and ensures that the reactive power adjustments are minimal yet sufficient to rectify the violations, typically resulting in at least one PQ bus voltage reaching its limit, as observed in Table II's last two columns.

TABLE II
EXTENDED ANALYSIS OF COMPUTATIONAL TIME AND VOLTAGE LIMITS AT BUSES. THIS TABLE SHOWCASES THE TOTAL COMPUTATIONAL TIME AND THE NUMBER OF BUSES MEETING THEIR VOLTAGE LIMITS ($V_i = V_i^{\text{LIM}}$) IN THE FINAL SOLUTION FOR DIFFERENT TEST SYSTEMS. "CG": PROPOSED COMPUTATIONAL GRAPH METHOD

Test System	Total c.p.u. time (sec) CG / traditional NR	No.of Buses with $V_i = V_i^{\text{lim}}$	
		$V_{PQ_i}^m$	$V_{PQ_i}^M$
IEEE14	0.0025 / 0.0011	2	0
IEEE30	0.0032 / 0.0028	1	0
NEGL39	0.4231 / 0.6624	2	2
IEEE57	0.6443 / 0.8216	1	1
PRCT89	1.1046 / 1.6421	1	1
IEEE118	1.3592 / 1.7380	3	0

The cpu time of the computing process for orthogonal projections is closely linked to the number of constraints that are either activated or breached. Correcting deviations in the lower voltage limits of the PQ buses may result in exceeding the upper voltage limits. The procedure is concluded only upon resolving all such discrepancies or upon recognizing the infeasibility of finding a viable solution. Table II's second column demonstrates the computational graph method's superiority over the traditional Newton-Raphson method in large-scale applications. For instance, in the case of a 118-bus test system, there was a notable 21.79% reduction in CPU time.

V. CONCLUSION

This paper's empirical findings demonstrate that the computational graph methodologies introduced are not only competitive but also surpass alternative methods in several aspects. The effectiveness of these techniques was evaluated using six electrical power networks. The approach successfully rectified PQ bus voltage limit violations without resorting to penalty functions, instead utilizing orthogonal projection onto feasible solution regions. This method proved efficient both independently and when integrated with the Gradient Projection technique aligned with computational graph strategies. Its efficacy and reliability were particularly evident in resolving significant voltage violations within acceptable computational timeframes.

REFERENCES

[1] B. Stott, "Review of load-flow calculation methods," Proceedings of the IEEE, vol. 62, no. 7, pp. 916-929, 1974.

[2] B. Taheri and D.K. Molzahn, "AC Power Flow Feasibility Restoration via a State Estimation-Based Post-Processing Algorithm," submitted, arXiv:2304.11418, 2023.

[3] F. Milano, "Continuous newton's method for power flow analysis," IEEE Transactions on Power Systems, vol. 24, no. 1, pp. 50-57, 2009.

[4] A. G. Expósito and E. R. Ramos, "Augmented rectangular load flow model," IEEE Transactions on Power Systems, vol. 17, no. 2, pp. 271276, 2002.

[5] A. Hauswirth, S. Bolognani, G. Hug and F. Dörfler, "Projected gradient descent on Riemannian manifolds with applications to online power system optimization," 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 2016, pp. 225-232, doi: 10.1109/ALLERTON.2016.7852234.

[6] J. H. Zheng, W. Xiao, C. Q. Wu, Z. Li, L. X. Wang, and Q. H. Wu, "A gradient descent direction based-cumulants method for probabilistic energy flow analysis of individual-based integrated energy systems," Energy, vol. 265, p. 126290, 2023. [Online]. Available: ISSN 0360-5442.

[7] L. M. Braz, C. A. Castro, and C. Murati, "A critical evaluation of step size optimization based load flow methods," IEEE Transactions on Power Systems, vol. 15, no. 1, pp. 202-207, 2000.

[8] L. Zhang and B. Zhang, "An Iterative Approach to Finding Global Solutions of AC Optimal Power Flow Problems," in ICML 2021 Workshop on Tackling Climate Change with Machine Learning, 2021.

[9] Y. Chen and C. Shen, "A Jacobian-free newton method with adaptive preconditioner and its application for power flow calculations," IEEE Transactions on Power Systems, vol. 21, no. 3, pp. 1096-1103, 2006.

[10] S. Abhyankar, Q. Cui, and A. J. Flueck, "Fast power flow analysis using a hybrid current-power balance formulation in rectangular coordinates," in IEEE PES T&D Conference and Exposition, 2014, pp. 1-5.

[11] M. Pirnia, C. A. Cañizares, and K. Bhattacharya, "Revisiting the power flow problem based on a mixed complementarity formulation approach," IET Generation, Transmission & Distribution, vol. 7, no. 11, pp. 11941201, 2013.

[12] B. Zhang and D. Tse, "Geometry of injection regions of power networks," IEEE Transactions on Power Systems, vol. 28, no. 2, pp. 788797, 2013.

[13] D. P. Bertsekas, "Nonlinear programming," Journal of the Operational Research Society, vol. 48, no. 3, pp. 334-334, 1997.

[14] R. Battiti, "First-and second-order methods for learning: between steepest descent and newton's method," Neural computation, vol. 4, no. 2, pp. 141-166, 1992.

[15] Y. Weng, R. Rajagopal, and B. Zhang, "A geometric analysis of power system loadability regions," IEEE Transactions on Smart Grid, 2019.