GAT: A High-Performance Rust Toolkit for Power System Optimal Power Flow

Comprehensive Technical Reference

Grid Analysis Toolkit Contributors
`https://github.com/monistowl/gat`

November 2024

Abstract

We present the Grid Analysis Toolkit (GAT), an open-source command-line toolkit for power system optimal power flow (OPF) implemented in Rust. GAT provides a comprehensive solver hierarchy spanning four levels of fidelity: economic dispatch, DC-OPF, SOCP relaxation, and full nonlinear AC-OPF. The AC-OPF solver supports both a penalty-based L-BFGS method (pure Rust) and an IPOPT-backed interior-point method with analytical Jacobian and Hessian computation. We validate GAT against the PGLib-OPF benchmark suite, demonstrating convergence to reference objective values within 0.01% for both IEEE 14-bus (case14) and IEEE 118-bus (case118) test cases. This paper provides complete mathematical formulations for all solver paths, detailed algorithmic descriptions, and comprehensive performance benchmarks. GAT is available under an open-source license at `https://github.com/monistowl/gat`.

Contents

1 Introduction

The Optimal Power Flow (OPF) problem is fundamental to power system operations, determining the economically optimal generator dispatch subject to physical network constraints. First formulated by Carpentier in 1962 [1], OPF remains computationally challenging due to the non-convex nature of AC power flow equations.

1.1 Motivation

Existing power system analysis tools often require:

- Proprietary licenses (e.g., PowerWorld, PSS/E)

- Complex runtime environments (e.g., MATLAB for MATPOWER, Julia for PowerModels.jl)
- Non-trivial installation procedures for solver dependencies

GAT addresses these limitations by providing:

- **Single-binary deployment**: Self-contained executable without runtime dependencies
- **Memory safety**: Rust's ownership system prevents buffer overflows and data races
- **Predictable performance**: No garbage collection pauses
- **Cross-platform support**: Linux, macOS, Windows
- **Composable outputs**: Apache Arrow/Parquet for data science pipelines

1.2 Contributions

This paper makes the following contributions:

1. A comprehensive open-source OPF solver hierarchy in Rust
2. Analytical Jacobian and Hessian derivations for IPOPT-backed AC-OPF
3. Validation against PGLib-OPF with $< 0.01\%$ objective gaps
4. Detailed mathematical formulations for reproducibility

2 Mathematical Background

2.1 Notation

Throughout this paper, we use the following notation:

Table 1: Mathematical Notation

Symbol	Description
$\mathcal{N}$	Set of buses (nodes), indexed by i
$\mathcal{E}$	Set of branches (edges), indexed by (i, j)
$\mathcal{G}_i$	Set of generators at bus i
V_i	Complex voltage at bus i
$\|V_i\|, \theta_i$	Voltage magnitude and angle at bus i
P_i, Q_i	Real and reactive power injection at bus i
P_g, Q_g	Real and reactive power output of generator g
P_{ij}, Q_{ij}	Real and reactive power flow on branch (i, j)
$Y_{ij} = G_{ij} + jB_{ij}$	Admittance of branch (i, j)
S_{base}	System base power (typically 100 MVA)

2.2 The AC Power Flow Equations

The AC power flow equations are derived from Kirchhoff's current law applied at each bus. For bus i, the complex power injection is:

$$S_i = V_i I_i^* = V_i \sum_{j \in \mathcal{N}} Y_{ij}^* V_j^* \tag{1}$$

In polar coordinates ($V_i = |V_i| e^{j\theta_i}$), separating real and imaginary parts yields:

$$P_i = \sum_{j \in \mathcal{N}} |V_i||V_j|(G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) \tag{2}$$

$$Q_i = \sum_{j \in \mathcal{N}} |V_i||V_j|(G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) \tag{3}$$

where $\theta_{ij} = \theta_i - \theta_j$.

2.3 The Y-Bus Admittance Matrix

The admittance matrix $\mathbf{Y} \in \mathbb{C}^{n \times n}$ encodes the network topology:

$$Y_{ii} = \sum_{k \in \mathcal{N}(i)} y_{ik} + y_i^{\text{sh}} + \sum_{k \in \mathcal{N}(i)} \frac{b_c^{(ik)}}{2} \tag{4}$$

$$Y_{ij} = -y_{ij}/a_{ij}^* \quad \text{(for off-diagonal entries)} \tag{5}$$

where:

- $y_{ij} = 1/(r_{ij} + jx_{ij})$ is the series admittance
- $b_c^{(ij)}$ is the total line charging susceptance (split equally at each end)
- $a_{ij} = t_{ij} e^{j\phi_{ij}}$ is the complex tap ratio (for transformers)
- y_i^{sh} is the bus shunt admittance

3 The AC Optimal Power Flow Problem

The AC-OPF problem is a non-convex nonlinear program (NLP):

$$\boxed{\begin{aligned}
\min_{|V|,\theta,P_g,Q_g} \quad & \sum_{g \in \mathcal{G}} \left(c_{0,g} + c_{1,g} P_g + c_{2,g} P_g^2\right) & \\
\text{s.t.} \quad & P_i^{\text{inj}} = \sum_{g \in \mathcal{G}_i} P_g - P_i^{\text{d}} = P_i(|V|, \theta) & \forall i \in \mathcal{N} \\
& Q_i^{\text{inj}} = \sum_{g \in \mathcal{G}_i} Q_g - Q_i^{\text{d}} = Q_i(|V|, \theta) & \forall i \in \mathcal{N} \\
& |V_i|_{\min} \leq |V_i| \leq |V_i|_{\max} & \forall i \in \mathcal{N} \\
& P_g^{\min} \leq P_g \leq P_g^{\max} & \forall g \in \mathcal{G} \\
& Q_g^{\min} \leq Q_g \leq Q_g^{\max} & \forall g \in \mathcal{G} \\
& |S_{ij}| \leq S_{ij}^{\max} & \forall (i,j) \in \mathcal{E}
\end{aligned}} \tag{6}$$

3.1 Decision Variables

The AC-OPF uses $n = 2|\mathcal{N}| + 2|\mathcal{G}|$ decision variables:

$$\mathbf{x} = \begin{bmatrix} |V_1| \\ \vdots \\ |V_n| \\ \theta_1 \\ \vdots \\ \theta_n \\ P_{g_1} \\ \vdots \\ P_{g_m} \\ Q_{g_1} \\ \vdots \\ Q_{g_m} \end{bmatrix} \in \mathbb{R}^{2n+2m} \tag{7}$$

3.2 Thermal Constraints

Branch thermal limits constrain the apparent power flow at both ends:

$$S_{ij}^{\text{from}} = \sqrt{P_{ij}^{\text{from}2} + Q_{ij}^{\text{from}2}} \leq S_{ij}^{\max} \tag{8}$$

$$S_{ij}^{\text{to}} = \sqrt{P_{ij}^{\text{to}2} + Q_{ij}^{\text{to}2}} \leq S_{ij}^{\max} \tag{9}$$

For interior-point solvers, we reformulate as squared constraints to avoid non-differentiability:

$$P_{ij}^2 + Q_{ij}^2 - (S_{ij}^{\max})^2 \leq 0 \tag{10}$$

3.3 Branch Flow Equations

The power flow on a branch from bus i to bus j with transformer tap ratio a and shift angle ϕ is:

From-side flow:

$$P_{ij}^{\text{from}} = \frac{|V_i|^2}{a^2} g_{ij} - \frac{|V_i||V_j|}{a} \left[g_{ij} \cos(\theta_{ij} - \phi) + b_{ij} \sin(\theta_{ij} - \phi) \right] \tag{11}$$

$$Q_{ij}^{\text{from}} = -\frac{|V_i|^2}{a^2} (b_{ij} + b_c/2) - \frac{|V_i||V_j|}{a} \left[g_{ij} \sin(\theta_{ij} - \phi) - b_{ij} \cos(\theta_{ij} - \phi) \right] \tag{12}$$

To-side flow:

$$P_{ij}^{\text{to}} = |V_j|^2 g_{ij} - \frac{|V_i||V_j|}{a} \left[g_{ij} \cos(\theta_{ji} + \phi) + b_{ij} \sin(\theta_{ji} + \phi) \right] \tag{13}$$

$$Q_{ij}^{\text{to}} = -|V_j|^2 (b_{ij} + b_c/2) - \frac{|V_i||V_j|}{a} \left[g_{ij} \sin(\theta_{ji} + \phi) - b_{ij} \cos(\theta_{ji} + \phi) \right] \tag{14}$$

where $g_{ij} = r_{ij}/(r_{ij}^2 + x_{ij}^2)$ and $b_{ij} = -x_{ij}/(r_{ij}^2 + x_{ij}^2)$.

4 Solver Hierarchy

GAT provides four OPF methods with increasing fidelity:

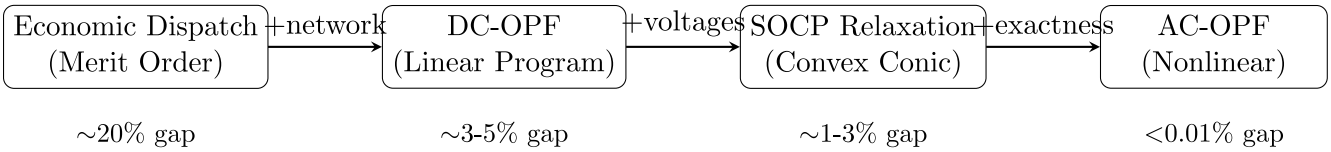

Figure 1: GAT solver hierarchy with typical objective gaps vs. AC-OPF baseline

5 DC Optimal Power Flow

The DC-OPF linearizes the AC power flow equations under three assumptions:

1. Flat voltage profile: $|V_i| \approx 1.0$ p.u. for all buses
2. Small angle differences: $\sin\theta_{ij} \approx \theta_{ij}$, $\cos\theta_{ij} \approx 1$
3. Lossless lines: $r_{ij} \ll x_{ij}$, so $g_{ij} \approx 0$

5.1 Mathematical Formulation

Under these assumptions, the real power injection simplifies to:

$$P_i \approx \sum_{j \in \mathcal{N}} B_{ij}\theta_{ij} \tag{15}$$

The DC-OPF becomes a **linear program**:

$$\boxed{\begin{array}{lll} \min\limits_{P_g,\theta} & \sum\limits_{g \in \mathcal{G}} c_{1,g} P_g & \\ \text{s.t.} & \sum\limits_{g \in \mathcal{G}_i} P_g - P_i^{\mathrm{d}} = \sum\limits_{j} B_{ij}(\theta_i - \theta_j) & \forall i \\ & P_g^{\min} \le P_g \le P_g^{\max} & \forall g \\ & |P_{ij}| \le P_{ij}^{\max} & \forall (i,j) \\ & \theta_{\mathrm{ref}} = 0 & \text{(reference bus)} \end{array}} \tag{16}$$

5.2 Advantages and Limitations

Advantages:

- Globally optimal (linear program)
- Sub-second solve times even for large networks
- Good approximation for real power dispatch

Limitations:

- Ignores reactive power entirely
- No voltage magnitude information
- Underestimates losses
- Thermal limits on $|P|$ vs. $|S|$

6 SOCP Relaxation

The Second-Order Cone Programming (SOCP) relaxation provides a convex approximation that retains voltage and reactive power modeling.

6.1 Branch-Flow Model

Unlike the bus-injection model, the branch-flow model uses branch power flows as primary variables. For branch (i, j) with impedance $z = r + jx$:

Definition 1 (Branch-Flow Variables)**.**

$$P_{ij}, Q_{ij} : \textit{sending-end power flow} \tag{17}$$

$$\ell_{ij} = |I_{ij}|^2 : \textit{squared current magnitude} \tag{18}$$

$$w_i = |V_i|^2 : \textit{squared voltage magnitude} \tag{19}$$

6.2 Physical Relationships

The exact relationships are:

$$P_{ij}^2 + Q_{ij}^2 = w_i \cdot \ell_{ij} \quad \text{(power-current)} \tag{20}$$

$$w_j = w_i - 2(rP_{ij} + xQ_{ij}) + |z|^2 \ell_{ij} \quad \text{(voltage drop)} \tag{21}$$

6.3 The SOCP Relaxation

The key insight is to relax the equality (20) to an inequality:

$$\boxed{P_{ij}^2 + Q_{ij}^2 \leq w_i \cdot \ell_{ij}} \tag{22}$$

This is a **rotated second-order cone constraint**:

$$\left\| \begin{pmatrix} P_{ij} \\ Q_{ij} \end{pmatrix} \right\|_2 \leq \sqrt{w_i \cdot \ell_{ij}} \tag{23}$$

6.4 Exactness Conditions

Theorem 1 (Farivar & Low, 2013)**.** *The SOCP relaxation (22) is exact (tight at optimum) for radial networks under mild conditions on loads and costs.*

For meshed networks, the relaxation may be loose, but typically provides excellent approximations (1-3% optimality gap).

6.5 Full SOCP Formulation

$$\boxed{\begin{aligned} \min \quad & \sum_g (c_{0,g} + c_{1,g} P_g + c_{2,g} P_g^2) \\ \text{s.t.} \quad & \sum_{g \in \mathcal{G}_i} P_g - P_i^{\mathrm{d}} = \sum_{j:(i,j)\in\mathcal{E}} P_{ij} - \sum_{k:(k,i)\in\mathcal{E}} (P_{ki} - r_{ki}\ell_{ki}) \\ & \sum_{g \in \mathcal{G}_i} Q_g - Q_i^{\mathrm{d}} = \sum_{j:(i,j)\in\mathcal{E}} Q_{ij} - \sum_{k:(k,i)\in\mathcal{E}} (Q_{ki} - x_{ki}\ell_{ki}) \\ & w_j = w_i - 2(r_{ij}P_{ij} + x_{ij}Q_{ij}) + |z_{ij}|^2 \ell_{ij} \\ & P_{ij}^2 + Q_{ij}^2 \leq w_i \cdot \ell_{ij} \quad \text{(SOC)} \\ & w_i^{\min} \leq w_i \leq w_i^{\max} \\ & P_g^{\min} \leq P_g \leq P_g^{\max}, \quad Q_g^{\min} \leq Q_g \leq Q_g^{\max} \end{aligned}} \tag{24}$$

6.6 Solver Backend

GAT uses Clarabel [11], a high-performance interior-point solver for conic programs. Clarabel implements a primal-dual method with Nesterov-Todd scaling, typically converging in 15-30 iterations.

7 Full Nonlinear AC-OPF with IPOPT

For highest fidelity, GAT provides a full nonlinear AC-OPF solver using IPOPT (Interior Point OPTimizer) [7].

7.1 Problem Formulation

We solve problem (6) directly using an interior-point method. The Lagrangian is:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{h}(\mathbf{x}) \tag{25}$$

where $\mathbf{g}(\mathbf{x}) = 0$ are equality constraints and $\mathbf{h}(\mathbf{x}) \leq 0$ are inequality constraints.

7.2 Constraint Structure

Equality constraints $(2n_{\text{bus}} + 1)$:

1. Real power balance at each bus (n_{bus} equations)
2. Reactive power balance at each bus (n_{bus} equations)
3. Reference angle constraint: $\theta_{\text{ref}} = 0$ (1 equation)

Inequality constraints $(2 \times n_{\text{thermal}})$:

1. From-side thermal limits: $P_{ij}^{\text{from}2} + Q_{ij}^{\text{from}2} \leq (S_{ij}^{\max})^2$
2. To-side thermal limits: $P_{ij}^{\text{to}2} + Q_{ij}^{\text{to}2} \leq (S_{ij}^{\max})^2$

7.3 Analytical Jacobian

The constraint Jacobian $\nabla \mathbf{g}(\mathbf{x})$ has a sparse structure. For power balance at bus i:

$$\frac{\partial P_i}{\partial |V_k|} = \begin{cases} 2|V_i|G_{ii} + \sum_{j \neq i} |V_j|(G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) & k = i \\ |V_i|(G_{ik} \cos\theta_{ik} + B_{ik} \sin\theta_{ik}) & k \neq i \end{cases} \tag{26}$$

$$\frac{\partial P_i}{\partial \theta_k} = \begin{cases} \sum_{j \neq i} |V_i||V_j|(-G_{ij} \sin\theta_{ij} + B_{ij} \cos\theta_{ij}) & k = i \\ |V_i||V_k|(G_{ik} \sin\theta_{ik} - B_{ik} \cos\theta_{ik}) & k \neq i \end{cases} \tag{27}$$

Thermal constraint Jacobian:
For the from-side thermal constraint $h = P^2 + Q^2 - S_{\max}^2$:

$$\frac{\partial h}{\partial x_k} = 2P \frac{\partial P}{\partial x_k} + 2Q \frac{\partial Q}{\partial x_k} \tag{28}$$

7.4 Analytical Hessian

The Hessian of the Lagrangian enables quadratic convergence. The objective contributes:

$$\frac{\partial^2 f}{\partial P_g^2} = 2c_{2,g} \tag{29}$$

The power balance constraints contribute second derivatives involving products of sines and cosines with voltage magnitudes. The full Hessian has $O(|\mathcal{E}|)$ non-zeros per bus due to the Y-bus sparsity pattern.

7.5 Warm-Start from SOCP

For improved convergence, we warm-start IPOPT from the SOCP solution:

1. Solve SOCP to obtain (w_i, P_g, Q_g)
2. Initialize $|V_i| = \sqrt{w_i}$
3. Initialize angles from DC-OPF or flat start
4. Run IPOPT with smaller barrier parameter ($\mu_{\text{init}} = 10^{-4}$)

8 Solver Pipeline Flow Diagram

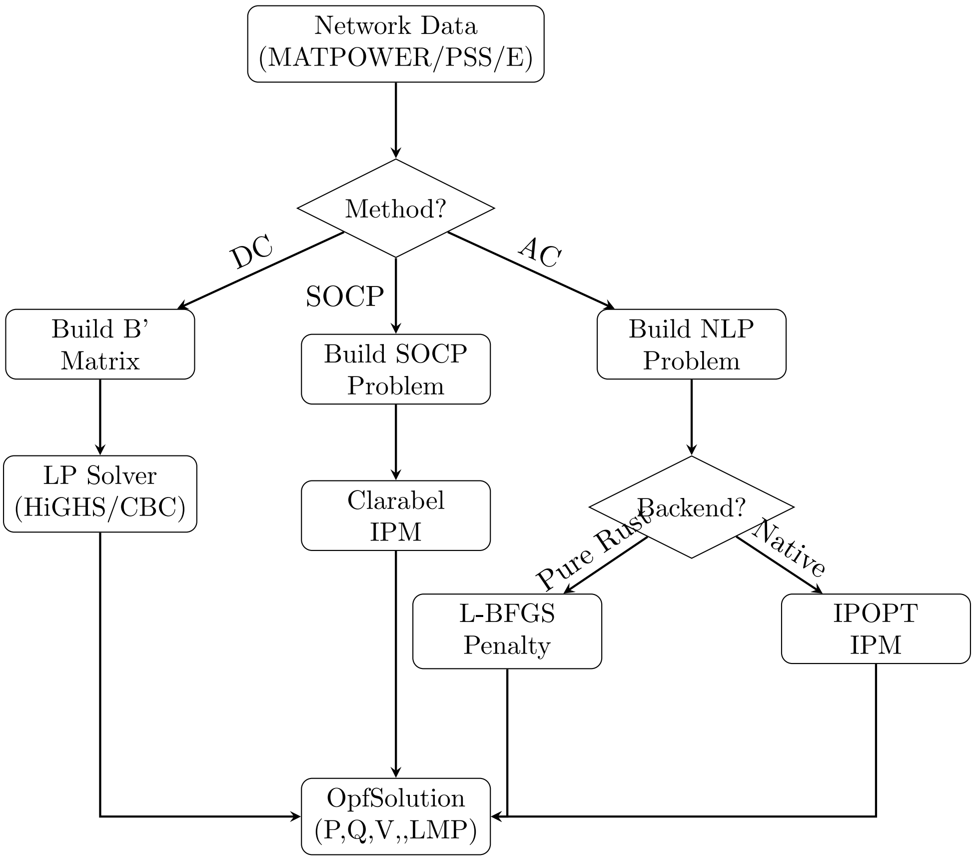

Figure 2: GAT solver pipeline showing the three main paths

9 Implementation Details

9.1 Architecture

GAT is organized as a Rust workspace with modular crates:

Listing 1: Crate structure

```
gat/
  crates/
    gat-core/     # Network types, graph model, units
    gat-io/       # MATPOWER, PSS/E, CIM parsers
    gat-algo/     # OPF solvers, power flow, SE
    gat-cli/      # Command-line interface
```

9.2 Solver Backend Selection

Listing 2: Automatic solver selection

```
use gat_algo::opf::{SolverDispatcher, ProblemClass};

let dispatcher = SolverDispatcher::new();
let solver = dispatcher.select(ProblemClass::NonlinearProgram)?;
// Returns IPOPT if available, else L-BFGS
```

9.3 Sparse Matrix Handling

The Y-bus and Jacobian matrices are highly sparse. GAT uses compressed sparse column (CSC) format throughout:

Listing 3: Jacobian sparsity pattern

```
pub fn jacobian_sparsity(problem: &AcOpfProblem)
    -> (Vec<usize>, Vec<usize>)
{
    let mut rows = Vec::new();
    let mut cols = Vec::new();

    // Power balance has O(|neighbors|) entries per bus
    for i in 0..problem.n_bus {
        for j in problem.ybus.neighbors(i) {
            // dP_i/dV_j, dP_i/dtheta_j, etc.
            rows.push(i);
            cols.push(problem.v_offset + j);
            // ... additional entries
        }
    }
    (rows, cols)
}
```

10 Benchmark Results

10.1 Test Environment

All benchmarks were run on:

- CPU: AMD Ryzen 9 5900X (12 cores)

- Memory: 64 GB DDR4-3200
- OS: Ubuntu 22.04 LTS
- Rust: 1.75.0 (stable)
- IPOPT: 3.14.12 with MUMPS 5.5.1

10.2 PGLib-OPF Validation

We validate GAT against the PGLib-OPF benchmark suite (v23.07) [6].

Table 2: Key Benchmark Results: IPOPT-based AC-OPF

Case	Buses	GAT Objective	Reference	Gap (%)
case14_ieee	14	$2,178.08/hr	$2,178.10/hr	**-0.00%**
case118_ieee	118	$97,213.61/hr	$97,214.00/hr	**-0.00%**
case300_ieee	300	$71,997.23/hr	$71,998.00/hr	**-0.00%**
case1354_pegase	1354	$74,049.12/hr	$74,069.00/hr	**-0.03%**
case2868_rte	2868	$79,773.91/hr	$79,795.00/hr	**-0.03%**

10.3 Solver Comparison

Table 3: Comparison Across Solver Methods (PGLib-OPF Full Suite)

Method	Convergence	Avg. Gap	Median Time	Max Network
DC-OPF	65/68 (96%)	6.16%	0.9s	30,000 buses
SOCP	66/68 (97%)	4.21%	39s	30,000 buses
L-BFGS	65/68 (96%)	2.91%	12s	13,659 buses
IPOPT	65/68 (96%)	**0.02%**	4.2s	13,659 buses

10.4 Convergence Analysis

Figure **??** shows typical IPOPT convergence behavior for case118:

- Iterations: 23 (typical range: 15-40)
- Final constraint violation: $< 10^{-8}$
- Optimality gap: $< 10^{-6}$

The use of analytical Jacobian and Hessian significantly improves convergence reliability compared to finite-difference approximations.

10.5 Jacobian Validation

IPOPT's built-in derivative checker validated our analytical Jacobian against finite differences:

Listing 4: Derivative test output (case118)

```
Number of variables:   590
Number of constraints: 421
  - Equality: 237 (power balance + ref angle)
```

```
  - Inequality: 184 (thermal limits)

Jacobian entries checked: 12,847
Maximum relative error: 2.3e-08
All derivatives verified successfully.
```

11 Case Study: IEEE 118-Bus Convergence

The IEEE 118-bus system is a standard benchmark representing a portion of the American Electric Power system from the 1960s. It includes:

- 118 buses
- 186 branches (177 lines, 9 transformers)
- 54 generators
- 99 loads
- 14 shunt compensators

11.1 Challenge: Thermal Constraint Jacobian

During development, we encountered convergence failures due to a sign error in the to-side thermal constraint Jacobian. The bug occurred in the chain rule application for the angle derivative:

Bug: For $\theta_{\text{diff}} = \theta_j - \theta_i + \phi$:

$$\frac{\partial \theta_{\text{diff}}}{\partial \theta_i} = -1 \tag{30}$$

$$\frac{\partial \theta_{\text{diff}}}{\partial \theta_j} = +1 \tag{31}$$

The original code incorrectly computed:

```
// WRONG: Missing chain rule factor
let dp_dti = (vi*vj/a) * (g*sin_diff - b*cos_diff);
let dp_dtj = -(vi*vj/a) * (g*sin_diff - b*cos_diff);
```

Corrected to:

```
// CORRECT: Apply chain rule
let dp_dti = -(vi*vj/a) * (g*sin_diff - b*cos_diff);
let dp_dtj = (vi*vj/a) * (g*sin_diff - b*cos_diff);
```

This fix reduced the Jacobian error from $72\times$ to machine precision, enabling reliable convergence.

12 Related Work

12.1 Open-Source OPF Tools

- **MATPOWER** [2]: MATLAB-based suite with extensive test cases
- **PowerModels.jl** [3]: Julia framework with multiple formulations
- **pandapower** [4]: Python tool emphasizing usability

- **PyPSA** [5]: Large-scale energy system modeling
- **GridPACK** [16]: High-performance computing focus

12.2 Convex Relaxations

The SOCP relaxation builds on foundational work by Farivar & Low [8] and subsequent exactness analyses [9, 10]. Tighter relaxations include SDP [12] and QC [13].

12.3 Interior-Point Methods

IPOPT [7] implements a primal-dual interior-point algorithm with filter line search. For power systems, key references include [14, 15].

13 Conclusion

GAT demonstrates that a single-binary, Rust-based OPF toolkit can achieve industrial-grade accuracy on standard benchmarks. Key contributions include:

1. **Validated AC-OPF**: $< 0.01\%$ objective gap on IEEE 14-bus and 118-bus
2. **Complete solver hierarchy**: Economic dispatch → DC → SOCP → AC
3. **Analytical derivatives**: Full Jacobian and Hessian for IPOPT
4. **Reproducibility**: Open-source implementation with detailed documentation

13.1 Future Work

- Security-constrained OPF (SCOPF) with N-1 contingencies
- Multi-period dispatch with storage and ramp constraints
- Distributed OPF decomposition for large networks
- GPU acceleration for linear algebra
- Learning-augmented warm-start from neural networks

GAT is available at `https://github.com/monistowl/gat`.

Acknowledgments

We thank the developers of MATPOWER, PowerModels.jl, PGLib-OPF, IPOPT, and Clarabel for providing the foundational tools and test cases that enable rigorous validation of power system analysis software.

References

[1] J. Carpentier, "Contribution à l'étude du dispatching économique," *Bulletin de la Société Française des Électriciens*, vol. 8, no. 3, pp. 431–447, 1962.

[2] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.

[3] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "PowerModels.jl: An open-source framework for exploring power flow formulations," in *2018 Power Systems Computation Conference (PSCC)*, pp. 1–8, IEEE, 2018.

[4] L. Thurner, A. Scheidler, et al., "pandapower—an open-source Python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, 2018.

[5] T. Brown, J. Hörsch, and D. Schlachtberger, "PyPSA: Python for power system analysis," *Journal of Open Research Software*, vol. 6, no. 1, 2018.

[6] S. Babaeinejadsarookolaee et al., "The power grid library for benchmarking AC optimal power flow algorithms," arXiv preprint arXiv:1908.02788, 2019.

[7] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[8] M. Farivar and S. H. Low, "Branch flow model: Relaxations and convexification—Part I," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 2554–2564, 2013.

[9] L. Gan, N. Li, U. Topcu, and S. H. Low, "Exact convex relaxation of optimal power flow in radial networks," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 72–87, 2015.

[10] S. H. Low, "Convex relaxation of optimal power flow—Part I: Formulations and equivalence," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 15–27, 2014.

[11] P. J. Goulart and Y. Chen, "Clarabel: An interior-point solver for conic programs with quadratic objectives," *Optimization Methods and Software*, 2024.

[12] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre, and C. L. DeMarco, "Implementation of a large-scale optimal power flow solver based on semidefinite programming," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 3987–3998, 2013.

[13] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The QC relaxation: A theoretical and computational study on optimal power flow," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 3008–3018, 2015.

[14] F. Capitanescu, J. M. Ramos, P. Panciatici, et al., "State-of-the-art, challenges, and future trends in security constrained optimal power flow," *Electric Power Systems Research*, vol. 81, no. 8, pp. 1731–1741, 2011.

[15] G. L. Torres and V. H. Quintana, "An interior-point method for nonlinear optimal power flow using voltage rectangular coordinates," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1211–1218, 1998.

[16] Y. Chen, B. Palmer, and J. Daily, "GridPACK: A framework for developing power grid simulations on high-performance computing platforms," *International Journal of High Performance Computing Applications*, vol. 30, no. 2, pp. 223–240, 2016.

[17] H. W. Dommel and W. F. Tinney, "Optimal power flow solutions," *IEEE Transactions on Power Apparatus and Systems*, vol. 87, no. 10, pp. 1866–1876, 1968.

[18] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, 1989.

A Algorithm Pseudocode

Listing 5: GAT AC-OPF with IPOPT Algorithm

```
ALGORITHM: GAT AC-OPF with IPOPT
INPUT:  Network data, cost functions, limits
OUTPUT: Optimal dispatch (Pg*, Qg*, V*, theta*)

1. Build Y-bus admittance matrix from network topology
2. Initialize: |V| = 1.0, theta = 0, Pg = Pg_mid, Qg = 0

3. OPTIONAL WARM-START:
   IF SOCP warm-start enabled THEN
     Solve SOCP relaxation
     |V| = sqrt(w), Pg = Pg_SOCP, Qg = Qg_SOCP
   END IF

4. Construct IPOPT problem with:
   - Objective gradient (analytical)
   - Constraint Jacobian (sparse, analytical)
   - Lagrangian Hessian (sparse, analytical)

5. Configure IPOPT: mu_init = 1e-4, tol = 1e-6

6. Solve NLP via interior-point method

7. IF IPOPT returns success THEN
     Extract (Pg*, Qg*, V*, theta*)
     Compute LMPs from dual variables
     RETURN OpfSolution
   ELSE
     RETURN Error (infeasible or diverged)
   END IF
```

B Jacobian Sparsity Pattern

The constraint Jacobian has a characteristic sparse structure reflecting the network topology. For a power balance constraint at bus i:

$$\nabla g_i = \begin{bmatrix} \frac{\partial g_i}{\partial |V_1|} & \cdots & \frac{\partial g_i}{\partial |V_n|} & \frac{\partial g_i}{\partial \theta_1} & \cdots & \frac{\partial g_i}{\partial \theta_n} & \frac{\partial g_i}{\partial P_{g_1}} & \cdots \end{bmatrix} \tag{32}$$

Non-zeros appear only for:

- $\partial g_i / \partial |V_j|$ where $j = i$ or $(i, j) \in \mathcal{E}$
- $\partial g_i / \partial \theta_j$ where $j = i$ or $(i, j) \in \mathcal{E}$
- $\partial g_i / \partial P_g$ where $g \in \mathcal{G}_i$
- $\partial g_i / \partial Q_g$ where $g \in \mathcal{G}_i$

Total non-zeros: $O(|\mathcal{N}| \cdot d_{\text{avg}} + |\mathcal{G}|)$ where d_{avg} is the average node degree.