Learning to Optimize: An Accelerated Deep Learning Framework for AC Optimal Power Flow Problem

Yu Zhang

Department of Electrical and Computer Engineering
University of California, Santa Cruz

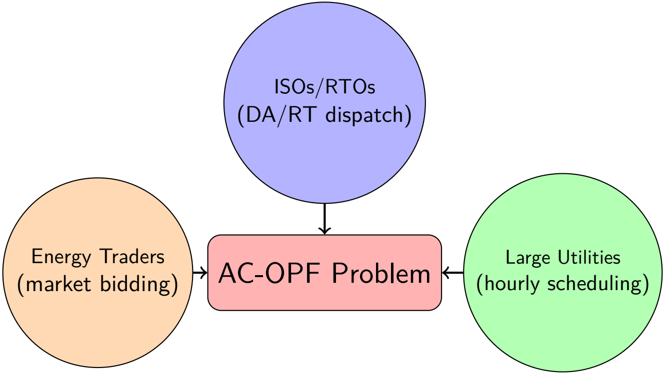
ICCOPT 2025 @USC

Outline

Outline

What is AC-OPF?

AC-OPF seeks the most cost-effective operating point of a power grid, minimizing operational cost (e.g. generation cost or power losses), subject to physical constraints (e.g., voltage limits and power flow limits).



Grid Optimization Competition

- Many variants:
  1. Multi-period, $N - k$ security constrained
  2. Decentralized: Lagrangian relaxation, ADMM, federated learning, etc
  3. Stochastic: expectation, chance constraint, CVaR, etc,
  4. Robust: affine or ellipsoidal uncertainty set
  5. Distributionally robust: KL divergence, Wasserstein distance, etc

Grid Optimization Competition

- Many variants:
  1. Multi-period, $N - k$ security constrained
  2. Decentralized: Lagrangian relaxation, ADMM, federated learning, etc
  3. Stochastic: expectation, chance constraint, CVaR, etc,
  4. Robust: affine or ellipsoidal uncertainty set
  5. Distributionally robust: KL divergence, Wasserstein distance, etc

- A highly **nonlinear** and **nonconvex** problem, even in its basic single-period form.

Grid Optimization Competition

- Many variants:
  1. Multi-period, $N - k$ security constrained
  2. Decentralized: Lagrangian relaxation, ADMM, federated learning, etc
  3. Stochastic: expectation, chance constraint, CVaR, etc,
  4. Robust: affine or ellipsoidal uncertainty set
  5. Distributionally robust: KL divergence, Wasserstein distance, etc

- A highly **nonlinear** and **nonconvex** problem, even in its basic single-period form.

- Grid Optimization Competition 2015-2024 (the longest ARPA-E project in history):
  1. Challenge 1 (C1): security-constrained AC-OPF /\$3.4M prize/
  2. Challenge 2 = C1 + unit commitment /\$2.4M prize/
  3. Challenge 3 = C2 + multiperiod dynamic markets /\$3M prize/

Towards Real-Time and High-Frequency OPF

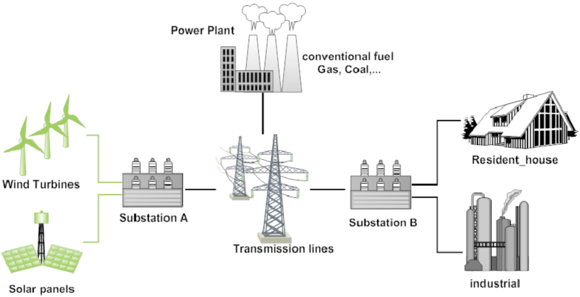


Figure: Renewables enabling two-way power flows [1]

- Repeatedly solving OPF to account for uncertainties in bus loading and generator bids in electricity markets [13].
- Monte Carlo methods for probabilistic OPF with correlated wind [12]; multi-period OPF under uncertainty [6].
- Power network expansion planning [2].

Algorithm Development for AC-OPF

- **Early Methods (1960s–1980s):**
  - Linear and Quadratic Programming (LP, QP) for steady-state analysis.
  - Limited by inability to handle nonlinearity and complex dynamics.
- **Nonlinear Programming Methods (1990s):**
  - NLP for modeling nonlinear constraints more accurately.
  - Interior-Point Methods (IPM) and Sequential Quadratic Programming (SQP) for large-scale and non-convex problems.

Algorithm Development for AC-OPF

- **Early Methods (1960s–1980s):**
  - Linear and Quadratic Programming (LP, QP) for steady-state analysis.
  - Limited by inability to handle nonlinearity and complex dynamics.
- **Nonlinear Programming Methods (1990s):**
  - NLP for modeling nonlinear constraints more accurately.
  - Interior-Point Methods (IPM) and Sequential Quadratic Programming (SQP) for large-scale and non-convex problems.
- **Advanced Techniques (2000s–2010s):**
  - Augmented Lagrangian and Trust-Region Methods for improved convergence.
  - Conic relaxations (SOCP, SDP): tight under certain conditions.
- **Recent Developments (2020s):**
  - Deep learning (CNN, RNN, GNN, etc) and data-driven optimization for faster, scalable solutions.

Outline

1. Introduction of AC Optimal Power Flow (AC-OPF)
2. Unsupervised Learning via Lagrangian Duality
3. Semi-supervised Learning with Physics-Informed Gradient Estimation
4. Summary

Gaps in Prior Work

Method	Limitation
Nonlinear programming	Heavy computational burden
Semi-definite programming (SDP) [9]	AC infeasible
DC-OPF [3]	AC infeasible
Supervised DNN [10, 5, 8, 11]	Ignore constraints, load mismatch, or data generated by conventional solvers
Unsupervised DNN, *DC3*[†] [4]	Fixed loss function
Unsupervised DNN, *NGT*[*] [7]	Load mismatch

[†] Priya Donti, David Rolnick, Zico Kolter. "DC3 (Deep Constraint Completion and Correction): A learning method for optimization with hard constraints." ICML 2021.

[*] Wanjun Huang, Minghua Chen. "DeepOPF-NGT (No Ground Truth): A fast unsupervised learning approach for solving AC-OPF problems without ground truth." ICML 2021 Workshop.

Contributions of Our Work

Our work:† Unsupervised deep learning for OPF via Lagrangian duality

- Utilize an augmented Lagrangian framework that integrates constraint violation into the training loss.
- Adaptively update Lagrange multipliers to enhance training efficiency and convergence.
- Leverage fast decoupled power flow (FDPF) to accelerate computation.

†Kejun Chen, Shourya Bose, Yu Zhang. "Unsupervised Deep Learning for AC Optimal Power Flow via Lagrangian Duality," GLOBECOM 2022.

AC-OPF Formulation

$$\underset{\mathbf{V},\boldsymbol{\theta},\mathbf{P}_g,\mathbf{Q}_g,\mathbf{s}^2}{\text{minimize}} \quad \sum_i c_i(P_{g,i}) \tag{1a}$$

subject to

$$\text{Nodal balance eq.} \begin{cases} P_{g,i} - P_{d,i} = V_i \sum_{j=1}^{N} V_j(G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij})\,\forall i \in \mathcal{N} \\ Q_{g,i} - Q_{d,i} = V_i \sum_{j=1}^{N} V_j(G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij})\,\forall i \in \mathcal{N} \end{cases} \tag{1b}$$

$$\text{Branch flow eq.} \begin{cases} p_{ij} = -G_{ij}V_i^2 + V_iV_j(G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij}), \ \forall (i,j) \in \mathcal{M} \\ q_{ij} = B_{ij}V_i^2 + V_iV_j(G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij}), \ \forall (i,j) \in \mathcal{M} \end{cases} \tag{1c}$$

$$\text{Flow limits} \begin{cases} s_{ij}^2 = p_{ij}^2 + q_{ij}^2, \ \forall (i,j) \in \mathcal{M} \\ s_{ij}^2 \leq (s_{ij}^{\max})^2, \ \forall (i,j) \in \mathcal{M} \end{cases} \tag{1d}$$

$$\text{Generation and voltage limits} \begin{cases} P_{g,i}^{\min} \leq P_{g,i} \leq P_{g,i}^{\max}, \ \forall i \in \overline{\mathcal{N}}_d \\ Q_{g,i}^{\min} \leq Q_{g,i} \leq Q_{g,i}^{\max} \ \forall i \in \overline{\mathcal{N}}_d \\ V_i^{\min} \leq V_i \leq V_i^{\max} \ \forall i \in \mathcal{N}. \end{cases} \tag{1e}$$

$$\underset{\mathbf{V},\theta,\mathbf{P}_g,\mathbf{Q}_g,\mathbf{s}^2}{\text{minimize}} \quad \sum_i c_i(P_{g,i}) \tag{1a}$$

subject to

Nodal balance eq. $\begin{cases} P_{g,i} - P_{d,i} = V_i \sum_{j=1}^{N} V_j(G_{ij}\cos\theta_{ij} + B_{ij}\sin\theta_{ij})\,\forall i \in \mathcal{N} \\ Q_{g,i} - Q_{d,i} = V_i \sum_{j=1}^{N} V_j(G_{ij}\sin\theta_{ij} - B_{ij}\cos\theta_{ij})\,\forall i \in \mathcal{N} \end{cases}$

Aim to train a neural network to learn the mapping from input $\mathbf{x} := [\mathbf{P}_d; \mathbf{Q}_d]$ to output $\mathbf{y}_o := [\mathbf{V}; \theta; \mathbf{P}_g; \mathbf{Q}_g; \mathbf{s}^2]$

Br

(1c)

$$\text{Flow limits} \begin{cases} s_{ij}^2 = p_{ij}^2 + q_{ij}^2, \ \forall (i,j) \in \mathcal{M} \\ s_{ij}^2 \leq (s_{ij}^{\max})^2, \ \forall (i,j) \in \mathcal{M} \end{cases} \tag{1d}$$

$$\text{Generation and voltage limits} \begin{cases} P_{g,i}^{\min} \leq P_{g,i} \leq P_{g,i}^{\max}, \ \forall i \in \overline{\mathcal{N}}_d \\ Q_{g,i}^{\min} \leq Q_{g,i} \leq Q_{g,i}^{\max} \ \forall i \in \overline{\mathcal{N}}_d \\ V_i^{\min} \leq V_i \leq V_i^{\max} \ \forall i \in \mathcal{N}. \end{cases} \tag{1e}$$

Variable Splitting

Split the output into three groups of variables:

$$\mathbf{y}_o := [\mathbf{V}; \theta; \mathbf{P}_g; \mathbf{Q}_g; \mathbf{s}^2] = \Big[\underbrace{\mathbf{P}_g; \mathbf{V}_{\overline{\mathcal{N}}_d}}_{\mathbf{y}} \;\vdots\; \underbrace{\theta; \mathbf{V}_d}_{\mathbf{z}_1} \;\vdots\; \underbrace{P_{g,\mathrm{ref}}; Q_{g,\mathrm{ref}}; \mathbf{Q}_g; \mathbf{s}^2}_{\mathbf{z}_2} \Big]$$

Motivation: Ensure feasibility with respect to power balance and physical flow constraints across all operating conditions.

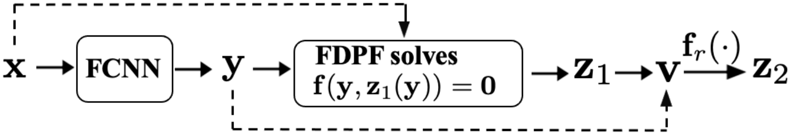


Figure: The proposed learning framework: $\mathbf{y} = \mathcal{F}_{\mathbf{W}}(\mathbf{x})$ is the *NN*'s output. $\{\mathbf{z}_1, \mathbf{z}_2\} = \mathrm{PF}(\mathbf{y})$ are obtained via power flow (PF) analysis.

Deep Learning Meets Lagrangian Duality

AC-OPF's compact formulation (without equality constraints):

$$\begin{aligned} \min \quad & f(\mathbf{y}, \mathbf{z}_2) && \text{(2a)} \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2) \leq \mathbf{0} \quad (\boldsymbol{\mu}) && \text{(2b)} \end{aligned}$$

- We propose using the augmented Lagrangian as the **training loss**:

$$\boxed{L(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2; \boldsymbol{\mu}) = f(\mathbf{y}, \mathbf{z}_2) + \frac{1}{2\alpha} \mathbf{1}^\top \Big([\boldsymbol{\mu} + \alpha \mathbf{h}]_+^2 - \boldsymbol{\mu} \odot \boldsymbol{\mu} \Big)}$$

Proposed Approach

Algorithm Unsupervised deep Learning via Lagrangian duality

Input: Load demand dataset $\mathcal{X}$.

for epoch $i = 1$ to n **do**
    Sample data points $\mathbf{x} \in \mathcal{X}$.
    Compute $\mathbf{y}$ through feedforward propagation.
    Obtain $\mathbf{z}_1$ using a PF solver (e.g., FDPF or Newton-Raphson).
    Compute $\mathbf{z}_2$ based on power balance and branch flow equations.
    Compute the training loss and update $\mathbf{W}$ via backprop.
    Update $\boldsymbol{\mu}_{i+1} \leftarrow \boldsymbol{\mu}_i$:
    if $i \mod m \equiv 0$ **then**
        update $\boldsymbol{\mu}_{i+1} \leftarrow \mathrm{ReLU}\big(\boldsymbol{\mu}_i + \alpha \mathbf{h}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2)\big)$ {stabilize training}
    end if
end for

Simulation Setup

- **Test cases:** IEEE-30/-118, which provide nominal values of load demand $\tilde{\mathbf{P}}_d$ and $\tilde{\mathbf{Q}}_d$.
- **Dataset:** 5,000 samples of $\mathbf{P}_d \sim \mathrm{Unif}\left(0.9\tilde{\mathbf{P}}_d, 1.1\tilde{\mathbf{P}}_d\right)$ and $\mathbf{Q}_d \sim \mathrm{Unif}\left(0.9\tilde{\mathbf{Q}}_d, 1.1\tilde{\mathbf{Q}}_d\right)$.
- **Hardware:** NVIDIA Titan RTX GPU (25GB RAM).
- **Training:**
  - FCNN: one hidden layer containing 50 neurons for IEEE-30 and 100 neurons for IEEE-118
  - Optimizer: Adam (PyTorch 1.7.1)
  - Maximum training epochs: $n = 1000$
  - Mini-batch size: $b = 32$

Performance Metrics

- **Optimality**: Measured by the optimal generation cost.
- **Feasibility rate**: The ratio of satisfied inequality constraints to the total number of inequality constraints.
- **Violation degree**: $\nu = [\mathbf{h}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2)]_+$, where $[a]_+ = \max\{a, 0\}$.
- **Load mismatch**: The relative error between the reconstructed and input load demands.
- **Computational time**: The runtime for solving the problem.

Competing Methods

- *NGT*: Different variable splitting scheme with training loss

$$\ell_{\text{NGT}} := \underbrace{f(\mathbf{y}, \mathbf{z}_2)}_{\text{obj. function}} + \eta \times \left[(1-\tau) \underbrace{\|\boldsymbol{\nu}\|_2^2}_{\text{constr. violation}} + \tau \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}_{\text{load mismatch}} \right] \quad (3)$$

- *DC3*: Same variable splitting scheme with training loss

$$\ell_{\text{DC3}} := f(\mathbf{y}, \mathbf{z}_2) + \lambda \|\boldsymbol{\nu}\|_2^2 \quad (4)$$

- *Proposed*: The training loss

$$\ell_{\text{prop}}(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2; \boldsymbol{\mu}) = f(\mathbf{y}, \mathbf{z}_2) + \frac{1}{2\alpha} \mathbf{1}^\top \left([\boldsymbol{\mu} + \alpha \mathbf{h}]_+^2 - \boldsymbol{\mu} \odot \boldsymbol{\mu} \right) \quad (5)$$

Performance of the NGT Method

Table: Performance of the NGT method on IEEE-30

η	τ	Generation cost	$\boldsymbol{\nu}$ Mean (10^{-6})	$\boldsymbol{\nu}$ Max (10^{-4})	Feasibility rate (%)	**Load mismatch**[†] (%)
5	0.2	0.0642	0.23	0.03	99.99	5.29
	0.5	0.0651	1.72	1.92	99.68	5.13
	0.8	0.0648	11.41	13.2	99.21	5.56
10	0.2	0.0646	0.20	0.25	99.95	5.55
	0.5	0.0662	1.14	1.41	99.77	5.52
	0.8	0.0664	6.33	6.78	99.38	5.22
15	0.2	0.0654	0.46	0.57	99.88	5.50
	0.5	0.0665	0.88	0.99	99.80	5.22
	0.8	0.0670	4.29	4.81	99.48	5.08

Table: Performance of the NGT method on IEEE-118 ($\tau = 0.5$)

η	Generation cost	$\boldsymbol{\nu}$ Mean (10^{-4})	$\boldsymbol{\nu}$ Max (10^{-2})	Feasibility rate (%)	**Load mismatch** (%)
5	8.80	2.88	2.56	79.10	140.40
10	12.71	0.43	0.25	99.20	21.93
15	12.91	0.09	0.51	98.69	18.77
20	13.07	0.00	0.20	99.50	16.94

[†] "A small ratio (around 1%) of load-generation imbalance is acceptable."

Comparison with DC3 Method

Table: Comparisons with DC3 and MIPS on IEEE-118

Method	λ	Generation cost	ν Mean (10^{-6})	ν Max (10^{-4})	Feasibility rate (%)	Runtime (s)
Different λ in DC3	1	13.145	23	72	97.66	0.52
	3	13.156	8.20	31	98.48	0.52
	5	13.161	7.92	32	98.62	0.52
	10	13.174	4.27	18	98.94	0.52
	15	13.181	2.88	13	99.16	0.52
	20	13.184	2.96	13	99.11	0.52
MATPOWER Interior Point Solver (MIPS)	-	13.137	0	0	99.95	35.33
Our method (NR)	-	13.162	**1.66**	**9**	99.21	0.52
Our method (FDPF)	-	13.158	**1.45**	**8**	99.17	**0.16**

- For DC3, there is a trade-off: larger λ tightens constraint satisfaction at the expense of higher generation cost.
- We outperform DC3 with lower violation degrees and achieve **220×** speedup over MIPS with 0.16% suboptimality.

Outline

Semi-supervised Learning for OPF

- Supervised learning: $(\mathcal{X}, \mathcal{Y})$, heavier computational burden.
- Unsupervised learning: $\mathcal{X}$, no guidance for training.

[†] Kejun Chen, Shourya Bose, Yu Zhang. "Physics-Informed Gradient Estimation for Accelerating Deep Learning based AC-OPF," IEEE Trans. on Industrial Informatics, June 2025.

Semi-supervised Learning for OPF

- Supervised learning: $(\mathcal{X}, \mathcal{Y})$, heavier computational burden.
- Unsupervised learning: $\mathcal{X}$, no guidance for training.

Highlights[†]: We develop a semi-supervised learning framework for data augmentation; i.e., load demand dataset $\mathcal{X} \mapsto$ augmented dataset $(\hat{\mathcal{X}}, \hat{\mathcal{Y}})$.

Algorithm Data Generation Using Pseudo-labeling

1: Use a solver to obtain optimal solutions for a small subset of $\mathcal{X}$.
2: Train a ridge regression model to map inputs $\mathbf{x}$ to outputs $\mathbf{y}$.
3: Generate pseudo-labels $\mathbf{y}$ for the remaining samples in $\mathcal{X}$. Project the infeasible pseudo values into the corresponding box constraint, i.e., $\mathbf{y} = \min\{\max\{\mathbf{y}, \mathbf{y}^{\min}\}, \mathbf{y}^{\max}\}$
4: Compute $\mathbf{z}_1$ and $\mathbf{z}_2$ using power balance and branch flow equations.

[†] Kejun Chen, Shourya Bose, Yu Zhang. "Physics-Informed Gradient Estimation for Accelerating Deep Learning based AC-OPF," IEEE Trans. on Industrial Informatics, June 2025.

Training Loss Design

Our training loss:

$$\boxed{\ell(\mathbf{y}, \mathbf{z}_1, \mathbf{z}_2) = L_c + w_o L_o + w_s L_s,}$$

where

- Objective function L_o.
- Supervised learning loss: $L_s(\mathbf{P}_g, \mathbf{v}) = \|\mathbf{P}_g - \tilde{\mathbf{P}}_g\|_2 + \|\mathbf{v} - \tilde{\mathbf{v}}\|_2$.
- Constraint violation loss: $L_c(\mathbf{z}_2, \mathbf{V}_d) = l_c(\mathbf{z}_2) + w_v l_c(\mathbf{V}_d)$, where $l_c(\mathbf{c}) = \left\|[\mathbf{c} - \mathbf{c}^{\max}]_+\right\|_2 + \left\|[\mathbf{c}^{\min} - \mathbf{c}]_+\right\|_2$ quantifies the box constraint violation.

Gradient Computation for Backpropagation

- We need to compute the derivatives for **backprop**

$$\frac{\mathrm{d}\ell}{\mathrm{dW}} = \frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}} \times \frac{\mathrm{d}\mathbf{y}}{\mathrm{dW}},$$

  where computing $\frac{\mathrm{d}\mathbf{y}}{\mathrm{dW}}$ is easy.

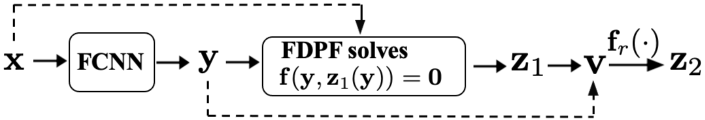
- Using the chain rule

$$\frac{\mathrm{d}\ell(\mathbf{y}, \mathbf{z}_1(\mathbf{y}), \mathbf{z}_2(\mathbf{y}, \mathbf{z}_1(\mathbf{y})))}{\mathrm{d}\mathbf{y}} = \frac{\partial \ell}{\partial \mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_1}\frac{\mathrm{d}\mathbf{z}_1}{\mathrm{d}\mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\mathrm{d}\mathbf{z}_2}{\mathrm{d}\mathbf{y}} \tag{6}$$

$$\frac{\mathrm{d}\mathbf{z}_2(\mathbf{y}, \mathbf{z}_1(\mathbf{y}))}{\mathrm{d}\mathbf{y}} = \frac{\partial \mathbf{z}_2}{\partial \mathbf{y}} + \frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1}\frac{\mathrm{d}\mathbf{z}_1}{\mathrm{d}\mathbf{y}} \tag{7}$$

  which are computed using the power flow and branch flow equations.

Implicit Gradient Computation



- By the implicit function theorem, we have

$$\frac{\mathrm{d}\mathbf{z}_1(\mathbf{y})}{\mathrm{d}\mathbf{y}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{z}_1}\right)^{-1}\left(\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right) = -\mathbf{J}_{z1}^{-1}\mathbf{J}_y \tag{8}$$

- $\frac{\partial \mathbf{f}}{\partial \mathbf{z}_1} := \mathbf{J}_{z1}$ can be extracted from $\mathbf{J}^{\mathrm{nodal}}$, and

$$\frac{\partial \mathbf{f}}{\partial \mathbf{y}} := \mathbf{J}_y = \left[\begin{array}{cc} \frac{\partial \mathbf{f}}{\partial \mathbf{P}_g} & \frac{\partial \mathbf{f}}{\partial \mathbf{V}_{\overline{\mathcal{N}}_d}} \end{array}\right] \in \mathbb{R}^{(2N_d+N_g)\times(2N_g+1)}$$

Compute Jacobian Matrices

- Finally, we get

$$\frac{\mathrm{d}\ell}{\mathrm{d}\mathbf{y}} = \frac{\partial \ell}{\partial \mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{y}} - \left(\frac{\partial \ell}{\partial \mathbf{z}_1} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1}\right) \times \mathbf{J}_{z1}^{-1}\mathbf{J}_y \tag{9}$$

$$= \frac{\partial \ell}{\partial \mathbf{y}} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{y}} - \mathbf{k}^\top \mathbf{J}_y\,, \tag{10}$$

where $\mathbf{k} \in \mathbb{R}^{2N_d+N_g}$ can be obtained by solving the linear system (more stable than matrix inversion):

$$\frac{\partial \ell}{\partial \mathbf{z}_1} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1} = \mathbf{k}^\top \mathbf{J}_{z1}\,. \tag{11}$$

Proposed Batch Gradient Estimation

- Let $\mathcal{J}_{z1} \in \mathbb{R}^{(2N_d+N_g)\times(2N_d+N_g)\times b}$ be the mini-batch tensor version of $\mathbf{J}_{z1}$. Then, the complexity of solving (11) in each training iteration is $\mathcal{O}(b \times (2N_d + N_g)^3)$.
- Mini-batch training increases gradient computation cost linearly with batch size b.
- We develop three ways to reduce computation: **Linearized Jacobian**, **Decoupled Jacobian**, and **Batch-mean Jacobian**.

Linearized Jacobian Estimation

Linearized Jacobian: linear approximations of the PF equations.

- Small angle difference $\implies$ $\sin\theta_{ij} \approx \theta_{ij}$, $\cos\theta_{ij} \approx 1$.
- Voltage magnitude near 1 per unit, with the mean value of all pseudo-measurements denoted as $\bar{v}_i$. Thus,

$$\begin{aligned} V_i \sin\theta_{ij} &\approx \bar{v}_i\theta_{ij}, \quad V_j \sin\theta_{ij} \approx \bar{v}_j\theta_{ij}\,, \\ V_iV_j \sin\theta_{ij} &\approx \bar{v}_i\bar{v}_j\theta_{ij}, \quad V_iV_j\cos\theta_{ij} \approx \bar{v}_iV_j\,. \end{aligned}$$

- The off-diagonal and main diagonal elements can be expressed as two linear systems:

$$\left[J_{ij}^{P\theta}\ J_{ij}^{PV}\ J_{ij}^{Q\theta}\ J_{ij}^{QV}\right]^\top = \mathbf{A}_{(ij)}\mathbf{v}_{(ij)} \tag{12}$$

$$\left[J_{ii}^{P\theta}\ J_{ii}^{PV}\ J_{ii}^{Q\theta}\ J_{ii}^{QV}\right]^\top = \mathbf{A}_{(i)}\mathbf{v}, \tag{13}$$

where $\mathbf{v}_{(ij)} = [\theta_i, \theta_j, V_i, V_j]^\top$.

Decoupled Jacobian Estimation

Consider the weak coupling in P-V and Q-θ:

- $\mathbf{J}_{z_1} := \begin{bmatrix} \mathbf{J}_{\mathbf{z}_1}^{P\theta} & \mathbf{J}_{\mathbf{z}_1}^{PV} \\ \mathbf{J}_{\mathbf{z}_1}^{Q\theta} & \mathbf{J}_{\mathbf{z}_1}^{QV} \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{J}_{\mathbf{z}_1}^{P\theta} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathbf{z}_1}^{QV} \end{bmatrix}$.
- The linear system (11) becomes $\frac{\partial \ell}{\partial \mathbf{z}_1} + \frac{\partial \ell}{\partial \mathbf{z}_2}\frac{\partial \mathbf{z}_2}{\partial \mathbf{z}_1} = \left[\ \mathbf{k}_p^\top \mathbf{J}_{z1}^{P\theta}, \quad \mathbf{k}_q^\top \mathbf{J}_{z1}^{QV}\ \right]$
- Computational complexity reduces to $\mathcal{O}\left((N_d + N_g)^3\right)$.

Batch-mean Jacobian Estimation

- Let $\mathcal{T} \in \mathbb{R}^{2N\times b}$ and $\mathcal{T}_{ij} \in \mathbb{R}^{4\times b}$ represent the tensor data of $\mathbf{v}$ and $\mathbf{v}_{(ij)}$, respectively, with $\bar{\mathcal{T}}$ and $\bar{\mathcal{T}}_{ij}$ denoting their batch-mean values.
- Then, we have the estimates of batch-mean Jacobian tensors:

$$\mathcal{J}_{ii} \approx \mathbf{A}_{(i)}\bar{\mathcal{T}}, \quad \mathcal{J}_{ij} \approx \mathbf{A}_{(ij)}\bar{\mathcal{T}}_{ij} \tag{14}$$

- For a given sample, the absolute errors due to the batch-mean replacement in $\mathcal{J}_{ii}^{P\theta}$ and $\mathcal{J}_{ii}^{QV}$ are given as

$$e_{ij}^{P\theta} = \left| \bar{v}_i \bar{v}_j G_{ij}(\theta_{ij} - \bar{\Theta}_{ij}) - \bar{v}_i B_{ij}(V_j - \bar{\mathcal{V}}_j) \right| ,$$
$$e_{ij}^{QV} = \left| \bar{v}_i G_{ij}(\theta_{ij} - \bar{\Theta}_{ij}) - B_{ij}(V_i - \bar{\mathcal{V}}_i) \right| .$$

- They are are small because $\theta_{ij} - \bar{\Theta}_{ij}$ and $V_j - \bar{\mathcal{V}}_j$ are typically negligible. In addition, diagonal errors are small due to the sparsity of the grid topology.

Simulation Setup

- **Test cases:** IEEE-118; PEGASE-1354/-2869/-9241.
- **Dataset:** $\mathbf{P}_d \sim \mathrm{Unif}\left(0.8\tilde{\mathbf{P}}_d, 1.2\tilde{\mathbf{P}}_d\right)$, $\mathbf{Q}_d \sim \mathrm{Unif}\left(0.8\tilde{\mathbf{Q}}_d, 1.2\tilde{\mathbf{Q}}_d\right)$.
  - 4,000 samples for PEGASE-9241
  - 10,000 samples for each of the other three systems
  - Training/validation/test ratio is 7:1:2
- **Hardware:** iMac with a 3.2 GHz CPU and 32 GB RAM.
- **Training:** learning rate schedule improves training convergence and stability.

System	FCNN structure	rate schedule (l_r, milestone, γ)
118	[236, 50, 236]	0.0005, 90, 0.2
1354	[2708, 50, 50, 2708]	0.0005, 70, 0.2
2869	[5738, 50, 50, 50, 5738]	0.0005, 1, 0.1
9241	[18482, 50, 50, 50, 18482]	0.001, 5, 0.01

Performance Metrics

1. **Optimality gap:** $l_{\text{cost}} = \frac{C - C_{\text{MIPS}}}{C_{\text{MIPS}}} \times 100\%$.

2. **Violation degree:** $l_v(\mathbf{z}_a) = [\mathbf{z}_a - \mathbf{z}_a^{\max}]_+ + [\mathbf{z}_a^{\min} - \mathbf{z}_a]_+$. Its maximum and mean values are $l_v^{\max}$ and $\bar{l}_v$.

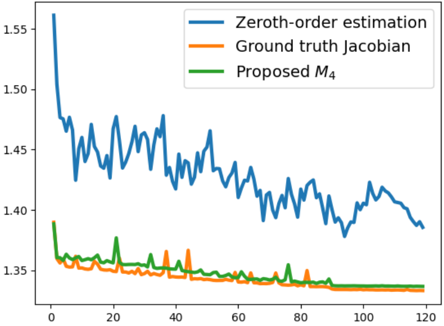
3. **Load mismatch:** Ratio of the absolute error of reconstructed load demand to the ground truth, denoted by e_l.

4. **Computational time:**
   - t_{train} and T_{train} are the per-epoch and total training time
   - T_{prop} and T_{opt} are the testing time for the proposed method and the MIPS solver, respectively.

5. **Storage:** Required memory of the data.

Training Loss Trajectory

- The zeroth-order estimation method performs worst in learning because of inaccurate gradient descent direction.
- Compared to the ground truth Jacobian gradient, M_4 (linearized decoupled Jacobian + batch-mean estimation + reduced branch set) achieves a comparable convergence performance.
- Less training time and storage.



Feasibility and Optimality

System	Evaluation metrics	M_{FCNN}	M_{STRT}	M_{CNN}	M_{DC3}	M_{DUAL}	M_4
118	$\bar{l}_v(10^{-4})$	0.68	**0.02**	7.84	**0.02**	0.13	0.12
	l_{cost}	0.88	0.30	0.00	0.27	0.04	0.54
1354	$\bar{l}_v(10^{-4})$	8.6	2.8	14.7	2.3	3.0	**0.44**
	l_{cost}	0.98	0.03	0.00	0.76	0.80	0.06
2869	$\bar{l}_v(10^{-4})$	43.4	6.7	13.2	1.8	5.0	**1.2**
	l_{cost}	0.91	0.08	0.0	0.80	1.49	0.09
9241	$\bar{l}_v(10^{-4})$	-	70.0	-	-	-	**4.0**
	l_{cost}	-	0.01	-	-	-	0.03

Training and Testing Time

Table: Training time and the storage size of the gradient data.

System	Gradient calculation	Method	Storage	t_{train}	T_{train}
2869	Zeroth-order	M_{FCNN}	0.260MB	7.9min	26.3h
		M_{STRT}		6.1min	10.1h
	Jacobian	M_{DC3}	8.432GB	89.0min	14.8h
	Proposed	M_4	0.263GB	4.9min	47min
9241	Zeroth-order	M_{FCNN}	0.73MB	-	-
		M_{STRT}		39.3 min	10.1h
	Jacobian	M_{DC3}	87.457GB	-	-
	Proposed	M_4	2.745GB	31.1min	5.1h

Table: Testing time results.

System	T_{prop}	T_{MIPS}
1354	10.4s	37.7min
2869	50.2s	109.7min
9241	361.7s	209.1min

Outline

1. Introduction of AC Optimal Power Flow (AC-OPF)
2. Unsupervised Learning via Lagrangian Duality
3. Semi-supervised Learning with Physics-Informed Gradient Estimation
4. Summary

Summary

Take-away points

- Variable splitting affects violation degrees and load mismatch.
- Efficient Jacobian estimation is crucial for fast training.

Future directions

- Apply the proposed learning frameworks to other OPF variants.
- Design scalable and reliable learning models for OPF with unit commitment and uncertain topology variations.
- Develop foundation models for OPF and other grid operations: Multi-task end-to-end learning.

Acknowledgments

Dr. Kejun Chen

Shourya Bose

Hellman Fellows Fund

CITRIS AND THE BANATAO INSTITUTE

UCSC COMMITTEE ON RESEARCH ACADEMIC SENATE

SCAN ME

References I

[1] Abdulhakim Alsaif. Challenges and benefits of integrating the renewable energy technologies into the ac power system grid. *American Journal of Engineering Research (AJER)*, 6(4):95–100, 2017.

[2] Erik F. Alvarez, Juan Camilo López, Luis Olmos, and Andres Ramos. An optimal expansion planning of power systems considering cycle-based AC optimal power flow. *Sustainable Energy, Grids and Networks*, 39:101413, 2024.

[3] Kyri Baker. Solutions of DC-OPF are never AC feasible, 2019. URL `https://arxiv.org/abs/1912.00319`.

[4] Priya Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. In *ICML*, 2021.

[5] Ferdinando Fioretto, Terrence W.K. Mak, and Pascal Van Hentenryck. Predicting AC optimal power flows: Combining deep learning and lagrangian dual methods. In *AAAI*, 2020.

[6] Ren Hu and Qifeng Li. Optimal operation of Power Systems with Energy Storage under uncertainty: A scenario-based method with strategic sampling. *IEEE Transactions on Smart Grid*, 13(2):1249–1260, 2022.

[7] Wanjun Huang and Minghua Chen. DeepOPF-NGT: A fast unsupervised learning approach for solving ac-opf problems without ground truth. In *ICML 2021 Workshop on Tackling Climate Change with Machine Learning*, 2021.

[8] Wanjun Huang, Xiang Pan, Minghua Chen, and Steven H. Low. DeepOPF-V: Solving AC-OPF problems efficiently, 2021. URL `https://arxiv.org/abs/2103.11793`.

References II

[9] Javad Lavaei and Steven H. Low. Zero duality gap in optimal power flow problem. *IEEE Transactions on Power Systems*, 27:92–107, 2012.

[10] Damian Owerko, Fernando Gama, and Alejandro Ribeiro. Optimal power flow using graph neural networks. In *ICASSP*, pages 5930–5934, 2020.

[11] Xiang Pan, Minghua Chen, Tianyu Zhao, and Steven H. Low. DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems. *IEEE Systems Journal*, 17(1):673–683, 2023.

[12] Weigao Sun, Mohsen Zamani, Hai-Tao Zhang, and Yuanzheng Li. Probabilistic optimal power flow with correlated wind power uncertainty via Markov Chain quasi-Monte-Carlo sampling. *IEEE Transactions on Industrial Informatics*, 15(11):6058–6069, 2019.

[13] Gregor Verbic, Antony Schellenberg, William Rosehart, and Claudio A. Canizares. Probabilistic optimal power flow applications to electricity markets. In *2006 International Conference on Probabilistic Methods Applied to Power Systems*, pages 1–6, 2006.