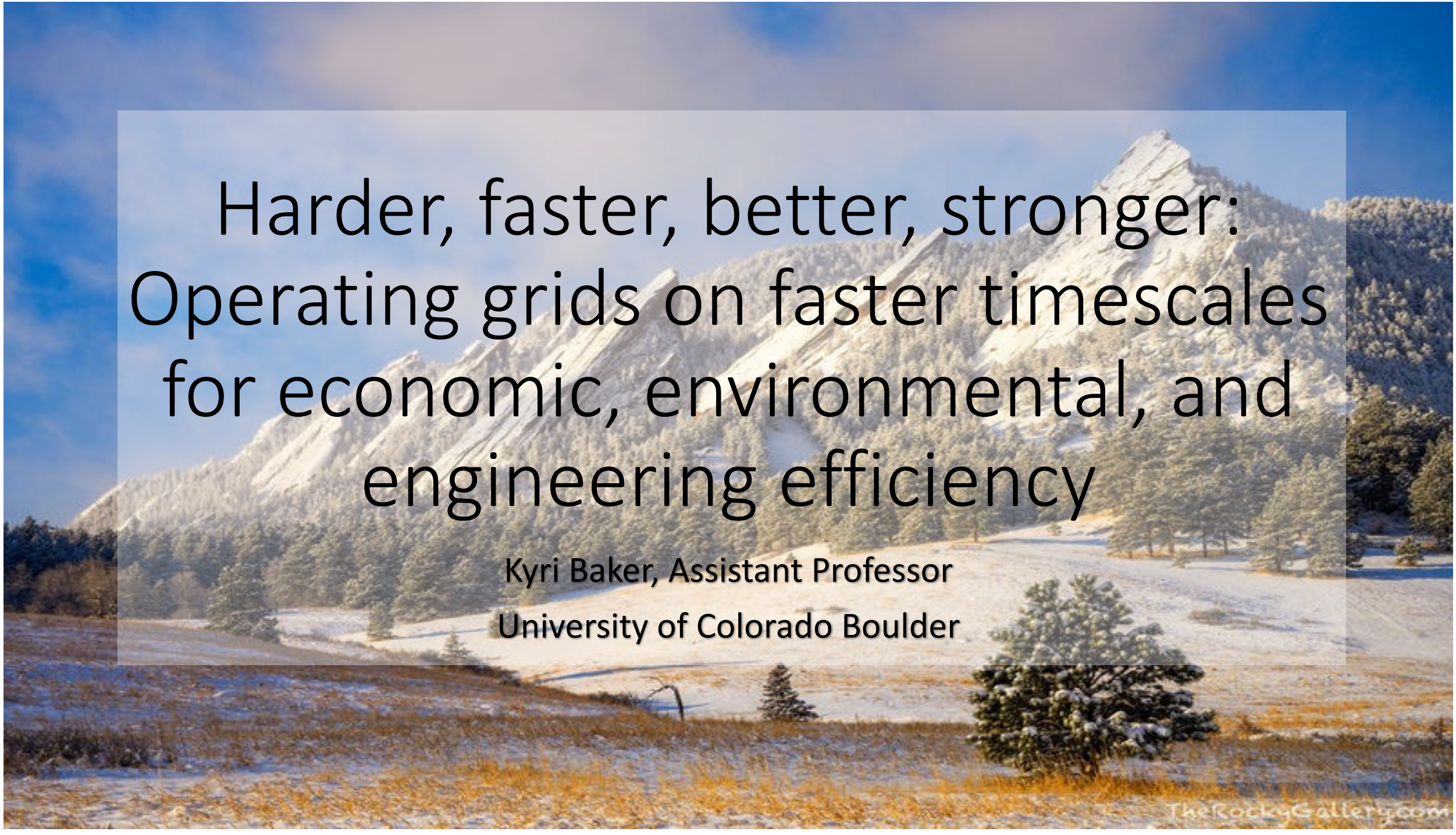


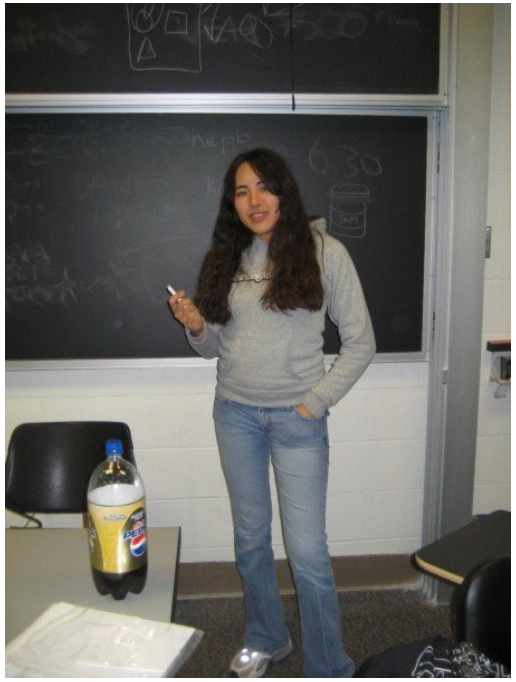
Harder, faster, better, stronger: Operating grids on faster timescales for economic, environmental, and engineering efficiency

Kyri Baker, Assistant Professor

University of Colorado Boulder

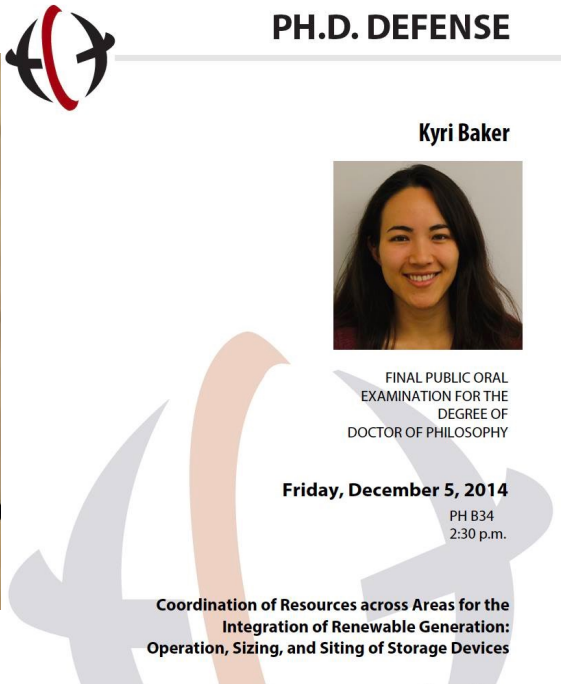
Before I moved to Colorado, I spent 9 years of my life here at CMU!

Moving into Mudge Hall, 2006



Pretending to be a professor,
Baker Hall, 2007

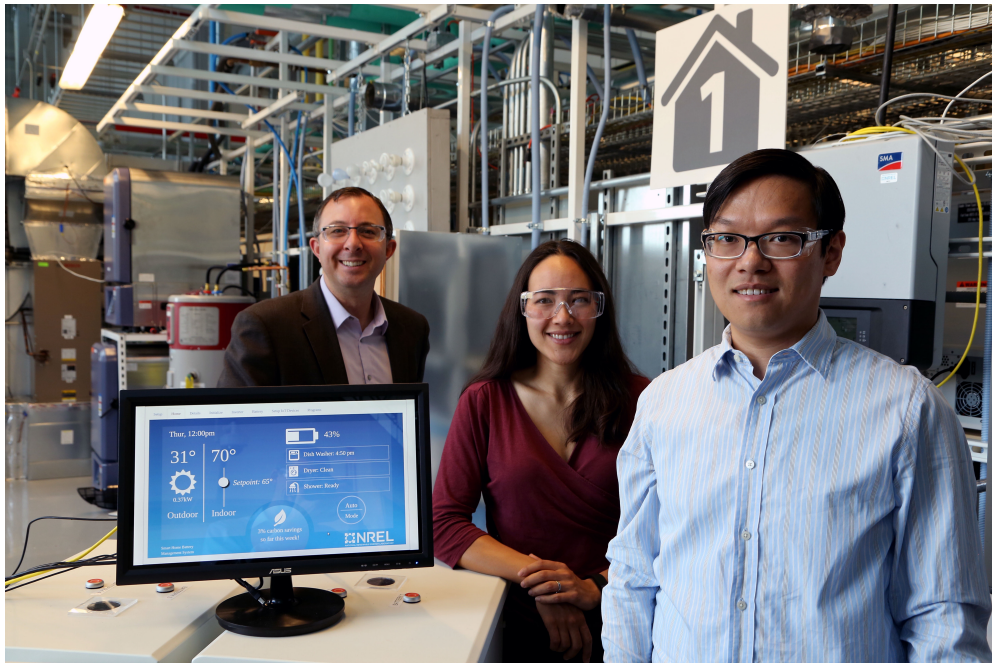
Performing at ECE Day, 2010



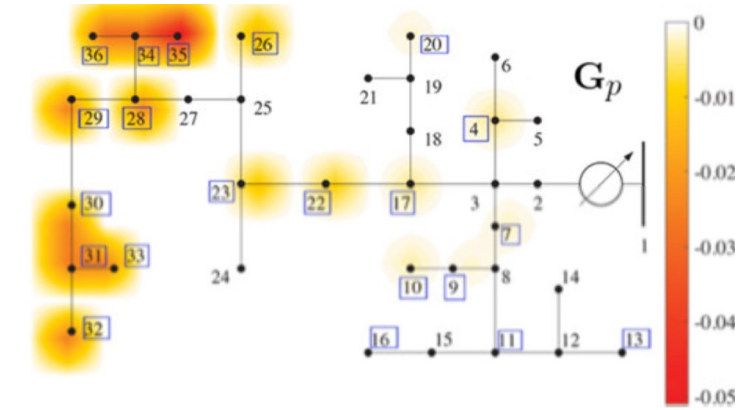
PhD defense, Dec 2014

GRIFFINLab

University of Colorado Boulder



Smart home research with NREL



Patented smart inverter control algorithms

arpa·e

GRID OPTIMIZATION (GO) COMPETITION

Award-winning Grid Optimization team

Talk Outline

GRIFFINLab

1. Harder, faster, better, stronger grids: the time is now

2. Learning **AC OPF** solutions very quickly

3. Using learning to obtain **distributed DC OPF** solutions faster

How complex is the grid, really?

The US National Academy of Engineering ranks electric power networks as **the number 1 greatest engineering achievement of the 20th century.**

(ranked by how much an achievement improved quality of life)

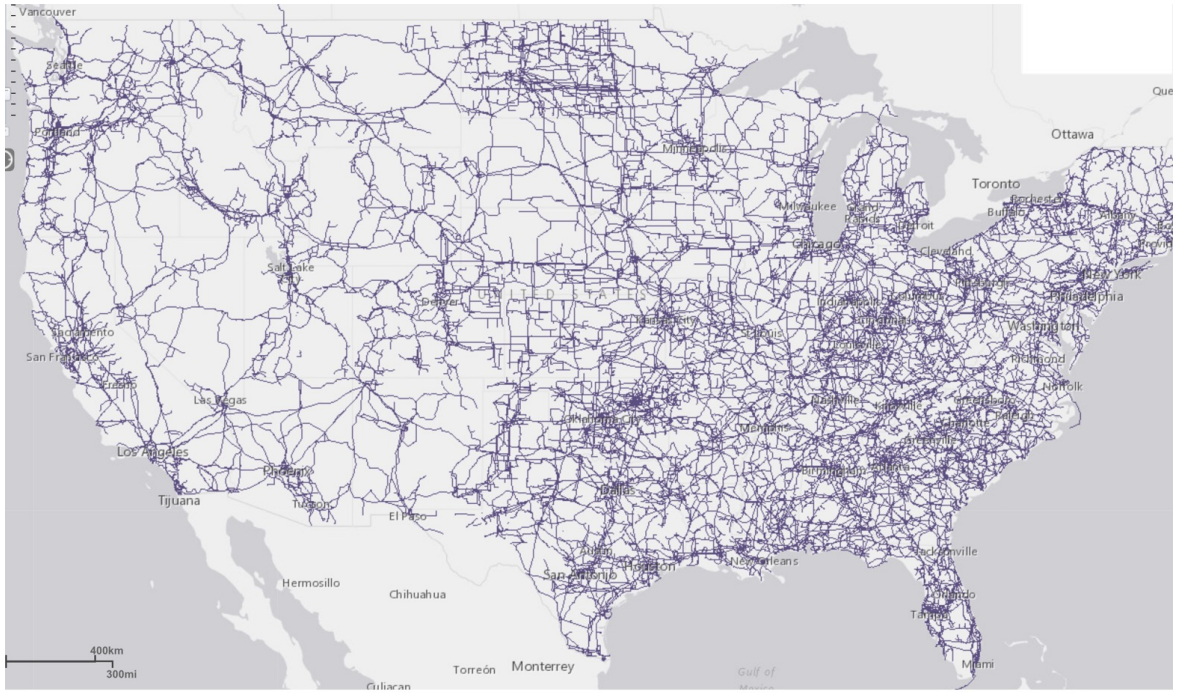


FIGURE 1: TRANSMISSION WIRES 138,000 VOLTS AND ABOVE

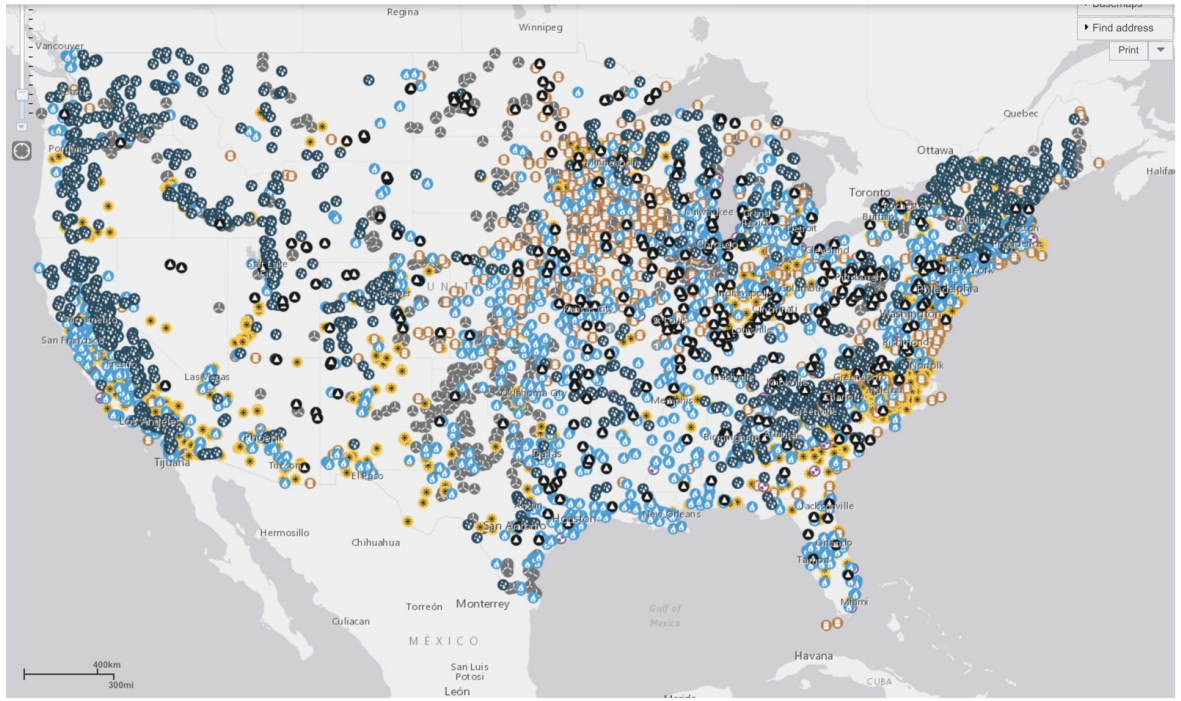
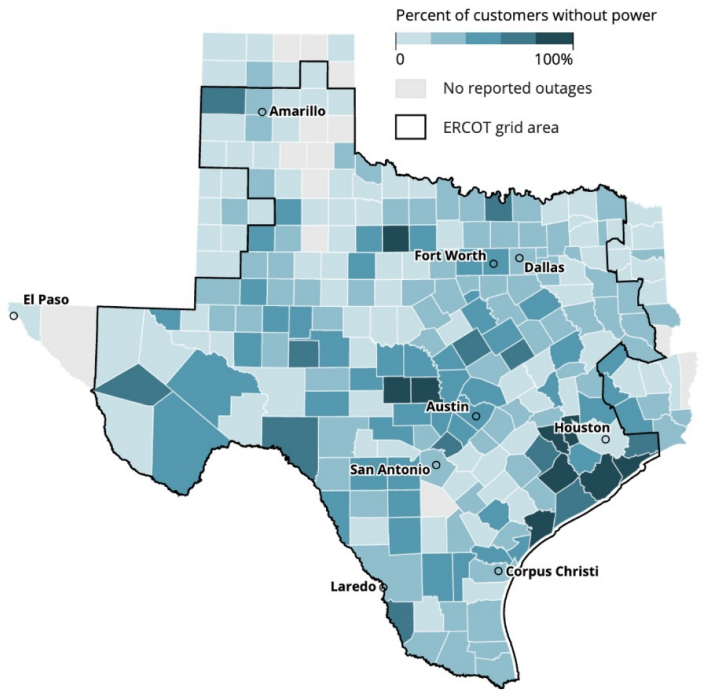


FIGURE 2: POWER GENERATORS (NON-RENEWABLE & RENEWABLE).

Outages are increasing…

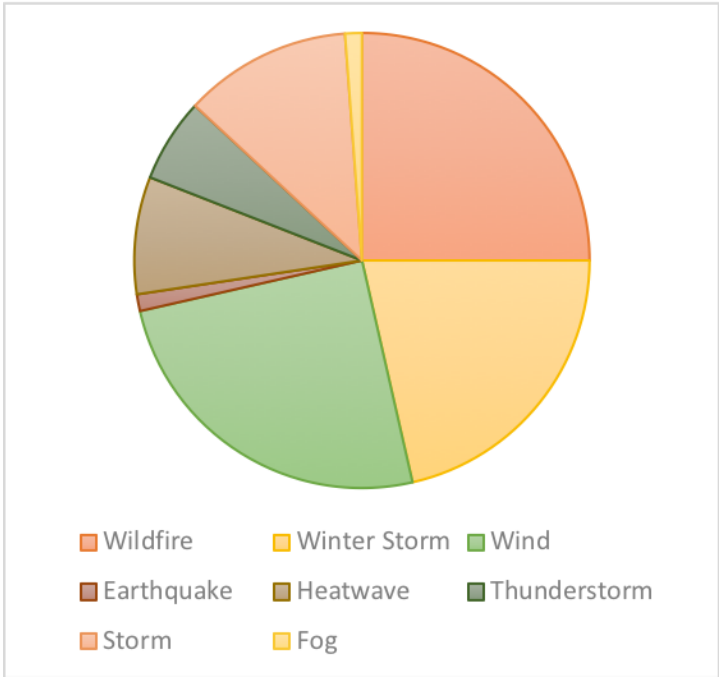
By by 2012, the average annual weather-related power outages had doubled since 2003

Transmission infrastructure is **aging,** with more than a quarter of the grid fifty years old or older.



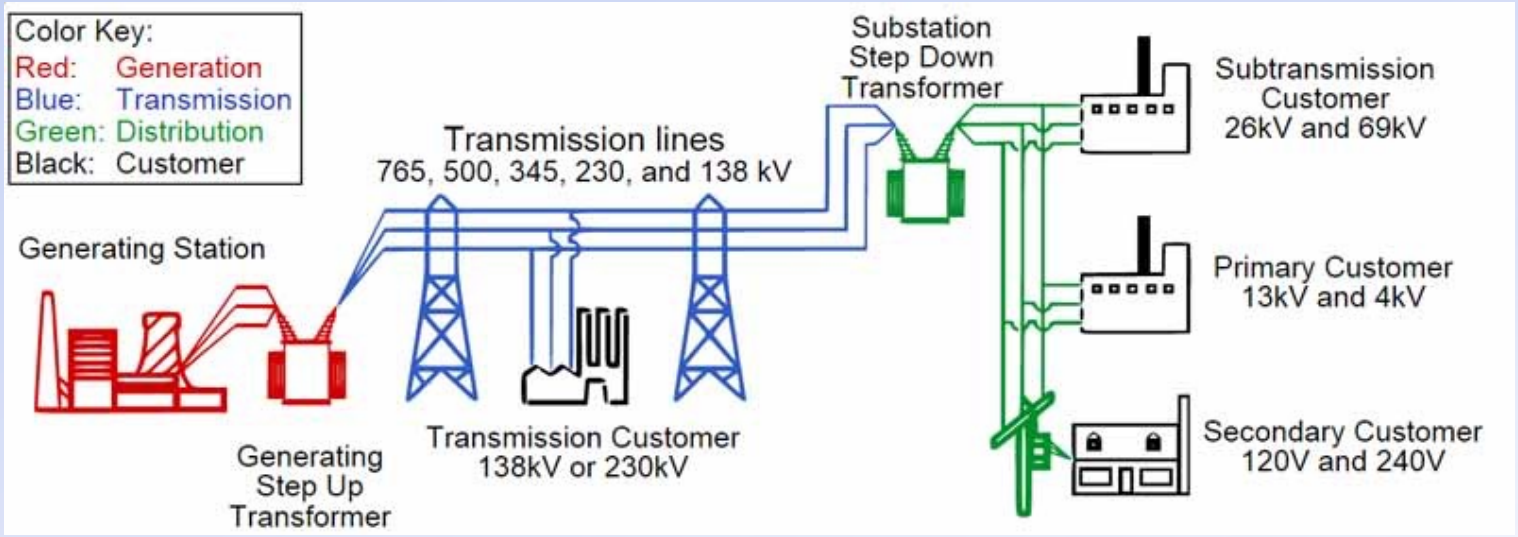
Source: The Texas Tribune and ProPublica

Feb. 16, 2021



Causes of major power outages
from 2004-2015 from a portion of WECC

A complex, synchronized system…

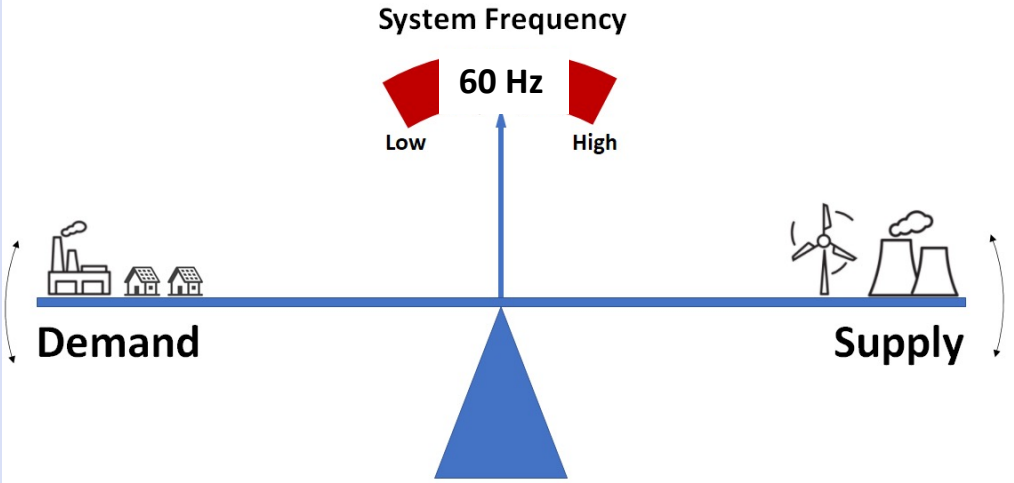


International, interstate power exchanges

Hundreds of miles of infrastructure

Millions of participants

Supply and demand must be balanced at all times in order to keep the grid stable



System frequency reflects the closeness of balance between **demand** and **supply**

How do we currently optimize large-scale grids?

$$\min_{\mathbf{x}} \quad f(\mathbf{x})$$
$$\text{s.t}: \quad g_i(\mathbf{x}) = 0$$
$$h_j(\mathbf{x}) \leq 0$$

Minimize cost to deliver power to consumers

Subject to physics, reliability constraints, etc.

How do we currently optimize large-scale grids?

$$\min_{\mathbf{x}} \quad f(\mathbf{x})$$

Minimize cost to deliver power to consumers

$$\text{s.t}: \quad g_i(\mathbf{x}) = 0$$
$$h_j(\mathbf{x}) \leq 0$$

Subject to physics, reliability constraints, etc.

This problem is very hard to solve quickly.

So instead, most grid operators make many simplifications – linearizing the grid models, neglecting losses, estimating line flows…

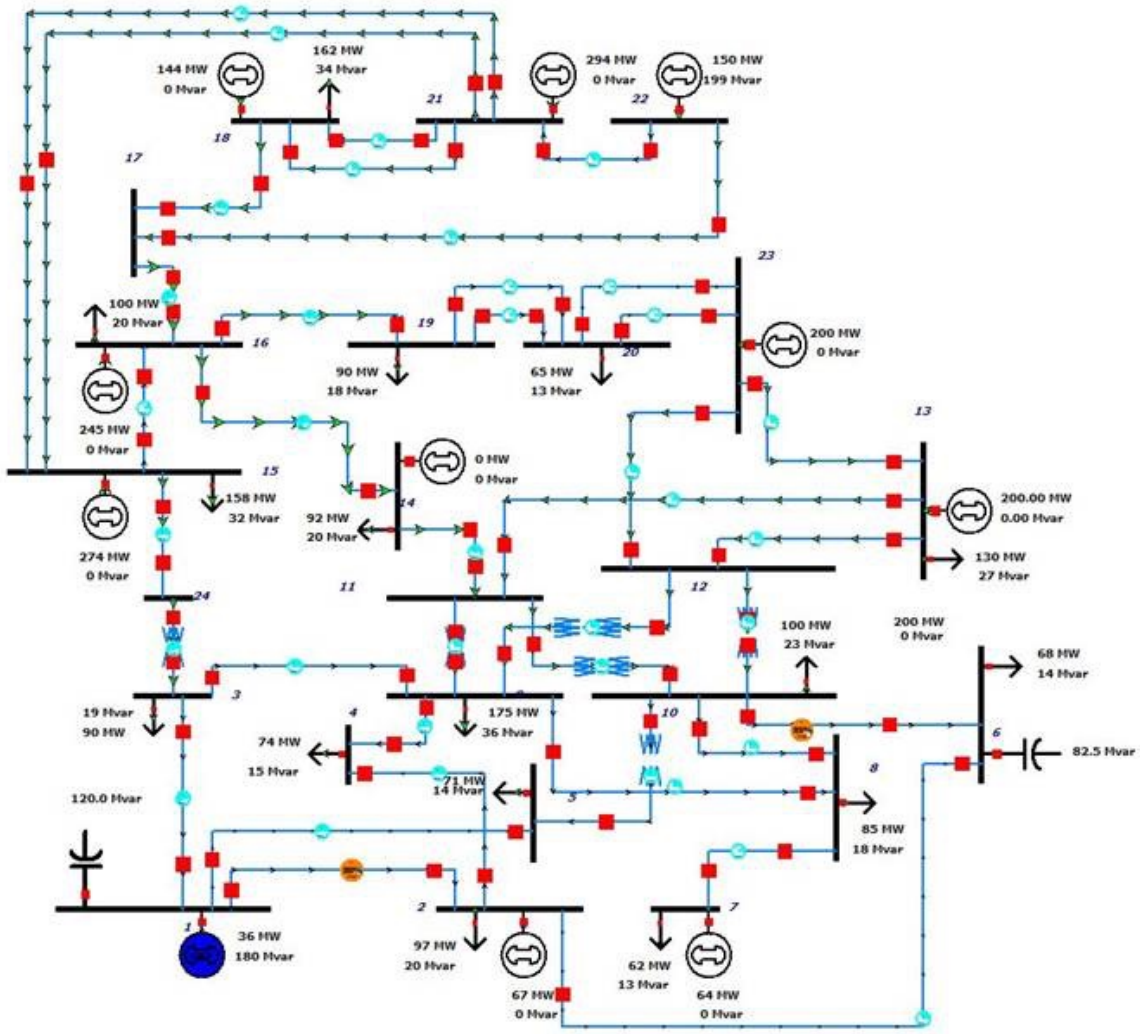
→ **Billions** of dollars annually are lost due to these suboptimalities (M. Cain, R. P. O'Neill, and A. Castillo, "History of optimal power flow and formulations," *FERC Technical Report*, Aug. 2013)

→ **~500 million metric tons of carbon dioxide** can be cut by improving global grid efficiencies (S. Jordaan and K. Surana, https://theconversation.com/we-calculated-emissions-due-to-electricity-loss-on-the-power-grid-globally-its-a-lot-128296)

What should we do?

Ideally, optimizing the grid should use the full fidelity, AC optimal power flow model

But this is hard to use for real-time operation

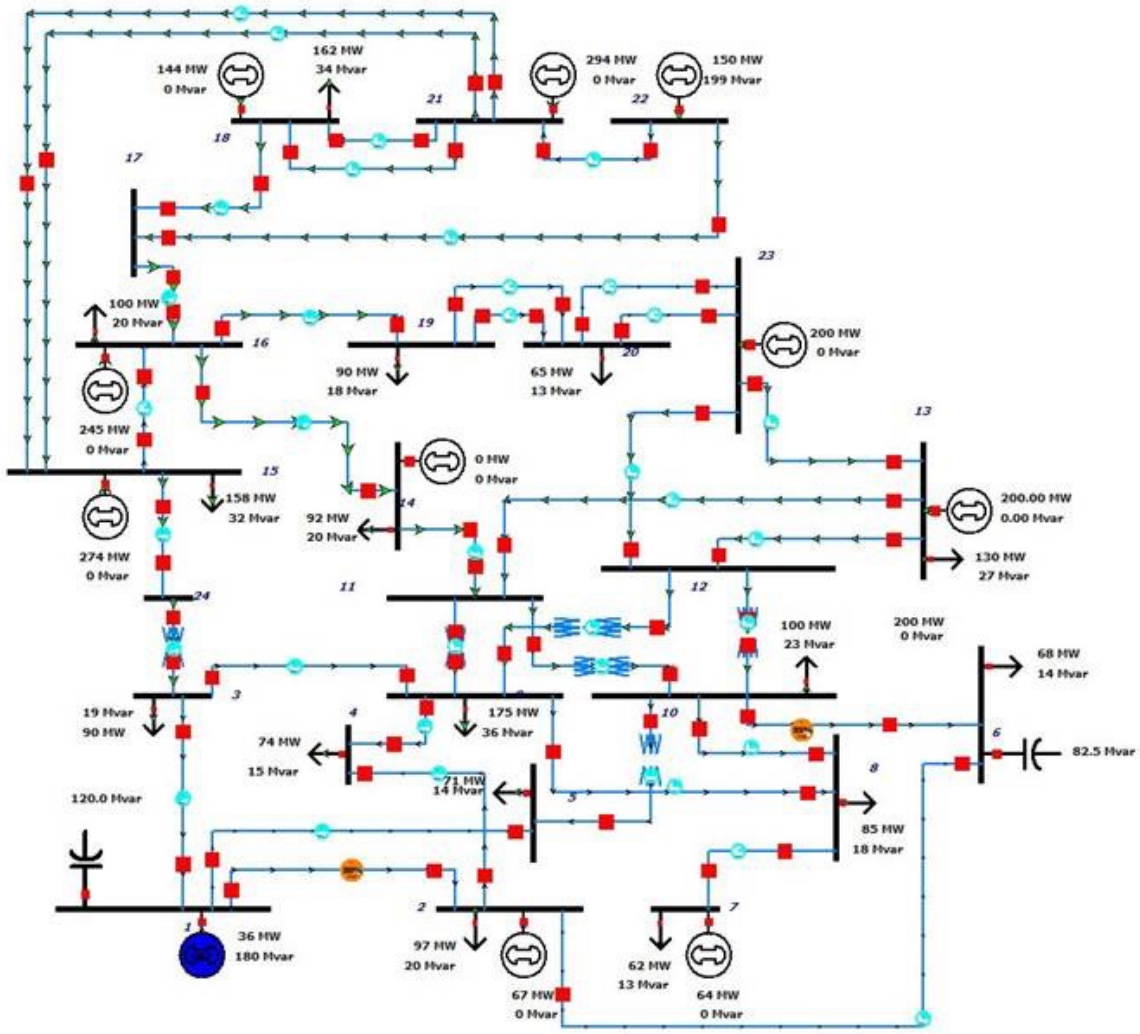


What should we do?

Ideally, optimizing the grid should use the full fidelity, AC optimal power flow model

But this is hard to use for real-time operation

Current operational methods don't capitalize on historical data. We utilize learned operator know-how, but how can we "teach" these algorithms to use previous knowledge?



Who cares about faster, more optimal grids?

The US Department of Energy has sponsored a competition to optimize grids up to 31,000 buses (roughly an entire interconnection).

The problem is **nonconvex,** with **integer variables, AC power flow equations,** considers both **reliability and optimality** in the scoring,

… and they wanted us to solve this in 5 minutes or less.

Who cares about faster, more optimal grids?

GRIFFINLab

Don't want to reveal too much about our solution yet, but the key takeaway is **operating grids faster leads to more reliability and lower costs.**

(Our division 2 and 4 code had a parallelization bug which is fixed now)

Place		Division 1		Division 2		Division 3		Division 4
1	**Electric Stampede**	190,564,664	NU_Columbia_Artelys	208,251,869	**Electric Stampede**	190,565,648	NU_Columbia_Artelys	207,721,687
2	GravityX	171,474,568	GravityX	190,594,154	NU_Columbia_Artelys	171,497,551	GravityX	186,997,477
3	NU_Columbia_Artelys	169,919,107	GMI-GO	170,851,464	Pearl Street Technologies	161,876,057	GMI-GO	170,848,944
4	Pearl Street Technologies	161,876,057	**Electric Stampede**	167,721,694	Gordian Knot	147,298,696	**Electric Stampede**	167,210,475
5	Gordian Knot	146,323,898	Pearl Street Technologies	161,876,057	GravityX	143,962,648	Pearl Street Technologies	161,876,057
6	GOT-BSI-OPF	107,881,004	magos	161,253,968	GOT-BSI-OPF	107,881,004	magos	161,253,968
7	GMI-GO	99,578,006	ARPA-e Benchmark	146,356,600	GMI-GO	99,593,494	ARPA-e Benchmark	146,340,636
8	magos	72,760,192	GOT-BSI-OPF	108,837,590	magos	72,760,192	Monday Mornings	115,001,830
9	ARPA-e Benchmark	49,947,791	Gordian Knot	108,414,227	Monday Mornings	47,645,573	GOT-BSI-OPF	108,837,590
10	Monday Mornings	42,000,595	Monday Mornings	105,517,759	ARPA-e Benchmark	47,329,390	SUGAR-CMU	103,781,578
11	SUGAR-CMU	32,697,483	SUGAR-CMU	103,993,746	SUGAR-CMU	32,664,818	Gordian Knot	102,942,572
12	GERS USA	24,048,950	GERS USA	22,746,121	GERS USA	23,414,842	GERS USA	25,354,657
13	CasePower	0	CasePower	0	CasePower	0	CasePower	0
14	GO-SNIP	0	GO-SNIP	0	GO-SNIP	0	GO-SNIP	0
15	M&H Grid Solutions	0	M&H Grid Solutions	0	M&H Grid Solutions	0	M&H Grid Solutions	0

https://gocompetition.energy.gov/challenges/challenge-2/Leaderboards/Trial3

Talk Outline

1. Harder, faster, better, stronger grids: the time is now

2. Learning **AC OPF** solutions very quickly

3. Using learning to obtain **distributed DC OPF** solutions faster

(*Note: our GO competition solution does not necessarily have anything to do with this section, I just used it as a motivating factor for this talk).*

Main idea:

Let’s lower emissions and cost ***just through updating grid software.***

Main Idea: Optimizing is hard.

→ Can we obtain the solution to an optimization problem without actually solving one?

$$\min_{\mathbf{v},\mathbf{p}_g} \sum_{j\in\mathcal{G}} a_j p_{g,j}^2 + b_j p_{g,j} + c_j$$

$$\text{s.t}:$$

$$|v_i| \sum_{m\in\mathcal{N}} |v_m|(G_{im}\cos(\theta_{im}) + B_{im}\sin(\theta_{im})) = p_{l,i} - \sum_{k\in\mathcal{G}_i} p_{g,k}, \quad \forall i \in \mathcal{N}$$

$$|v_i| \sum_{m\in\mathcal{N}} |v_m|(G_{im}\sin(\theta_{im}) - B_{im}\cos(\theta_{im})) = q_{l,i} - \sum_{k\in\mathcal{G}_i} q_{g,k}, \quad \forall i \in \mathcal{N}$$

$$\underline{p}_{g,j} \le p_{g,j} \le \overline{p}_{g,j}, \quad \forall j \in \mathcal{G}$$

$$\underline{q}_{g,j} \le q_{g,j} \le \overline{q}_{g,j}, \quad \forall j \in \mathcal{G}$$

$$\underline{|v|} \le |v_i| \le \overline{|v|}, \quad \forall i \in \mathcal{N}.$$

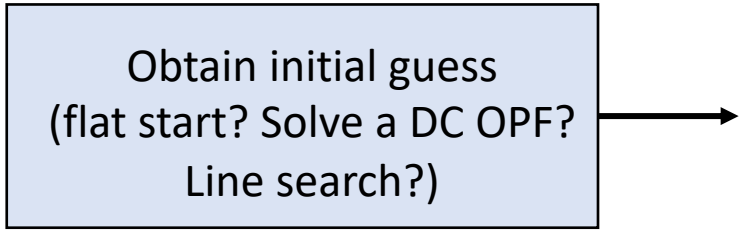
AC Optimal Power Flow (OPF) can comprise a nonconvex problem with thousands of constraints and variables.

No wonder basically no grid operators use it for optimizing generator dispatch!

How does it work?

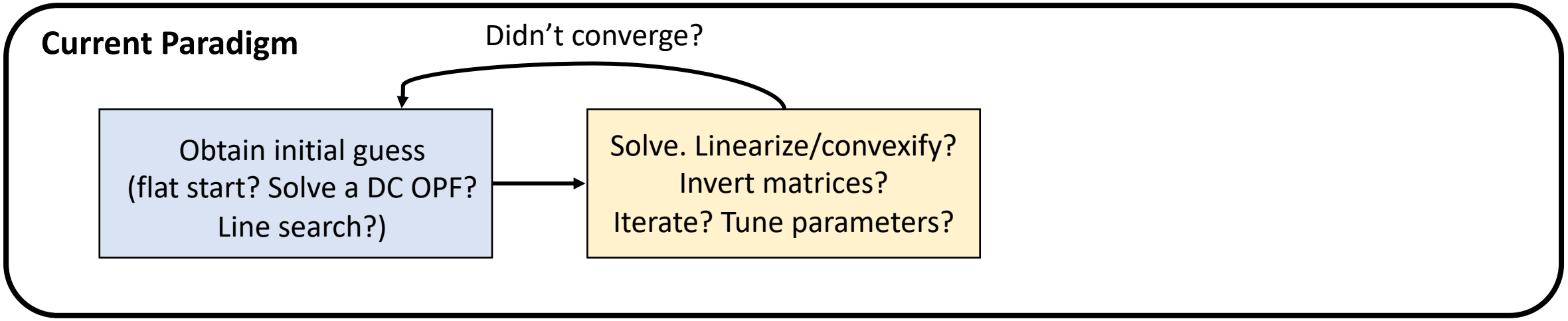
→ Can we obtain the solution to an optimization problem without actually solving one?

Current Paradigm



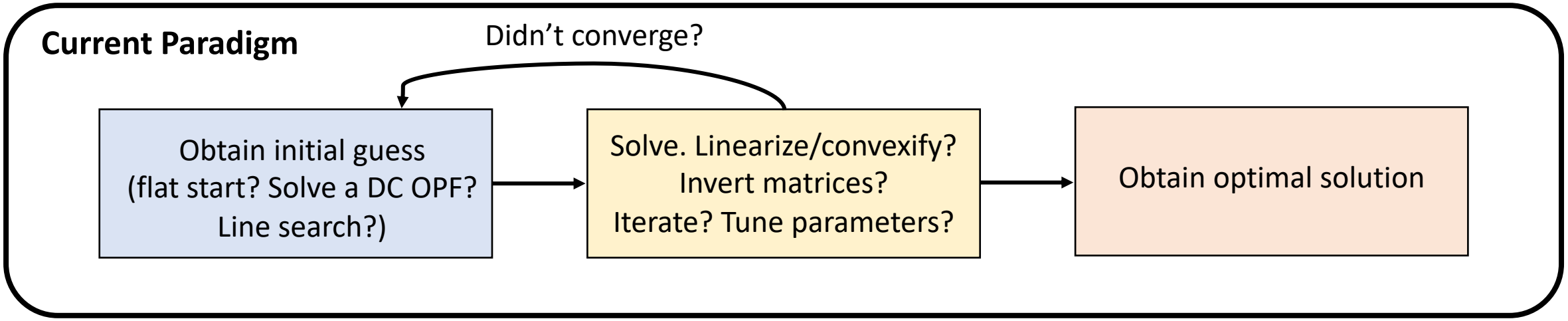
How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?



How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?



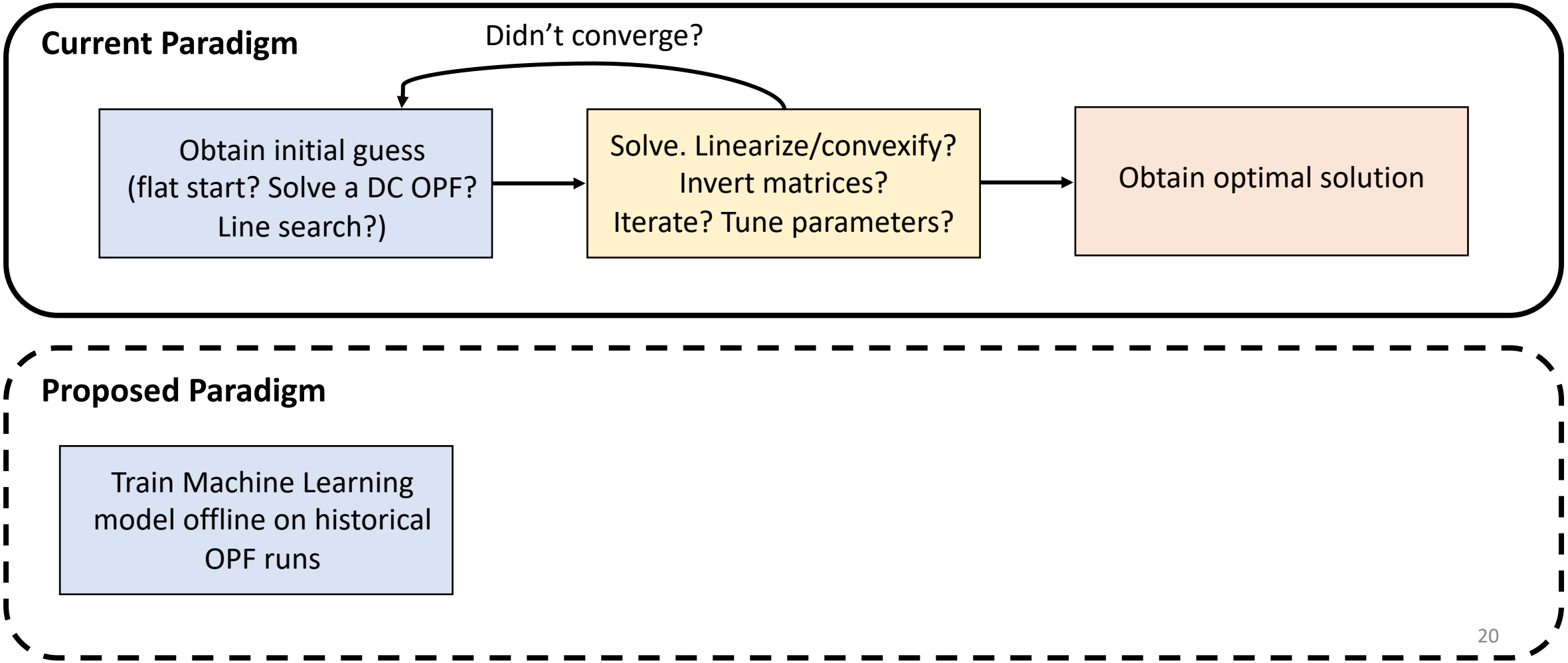
Can we use historical OPF runs to inform future optimizations?

"the volume of data being generated in the power sector has grown tremendously….
the majority of data is either not logged, or they are overwritten very quickly[1]"

[1] H. Akhavan-Hejazi and H. Mohsenian-Rad, "Power systems big data analytics: An assessment of paradigm shift barriers and prospects," *Energy Reports*, 2018.

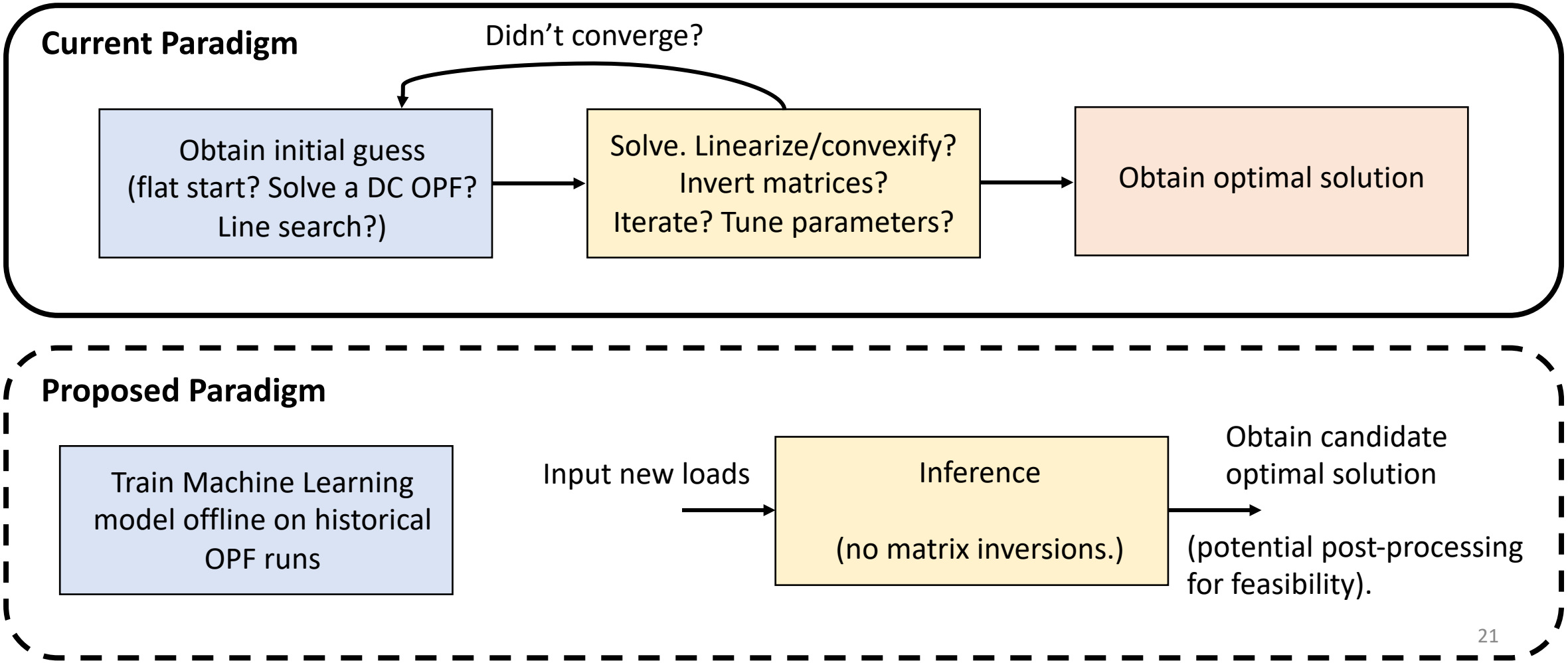
How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?



How does it work?

→ Can we obtain the solution to an optimization problem without actually solving one?



Does this actually work?

July 2020: A. Velloso and P. Van Hentenryck obtain negligible feasibility and optimality gaps (under 0.1%) for Security Constrained DC OPF on a 1,888 bus network in less than two seconds[2].

Sept. 2019: A. Zamzam and K. Baker obtain feasible AC OPF solutions with negligible optimality gaps 6-20x faster than a state of the art solver[3].

Dec. 2020: H. Lange, B. Chen, M. Berges, and S. Kar solve difficult AC OPF problems (with binary commitment variables) by using deep learning and the Holomorphic Embedded Load Flow method (HELM)[4]

[2] A. Velloso and P. Van Hentenryck, "Combining Deep Learning and Optimization for Security-Constrained Optimal Power Flow," https://arxiv.org/abs/2007.07002, July 2020.

[3] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," IEEE SmartGridComm, https://arxiv.org/abs/2861719, 2019.

4 H. Lange, B. Chen, M. Berges, and S. Kar, "Learning to Solve AC Optimal Power Flow by Differentiating through Holomorphic Embeddings," https://arxiv.org/pdf/2012.09622.pdf Dec. 2020

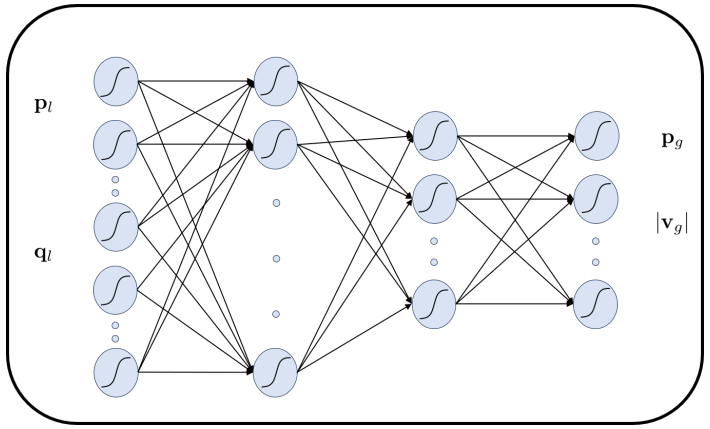
Why is it powerful?

→ We can convexify or linearize the hard equations (e.g. the AC power flow equations) to solve these problems quickly, but convexifying generally makes us lose information

→ Neural networks can represent nonconvex, complicated relationships between variables

→ Inference (the process of making a prediction) mostly involves applying functions, multiplying, and adding

→ It's an approximation, but it can be a damn good one



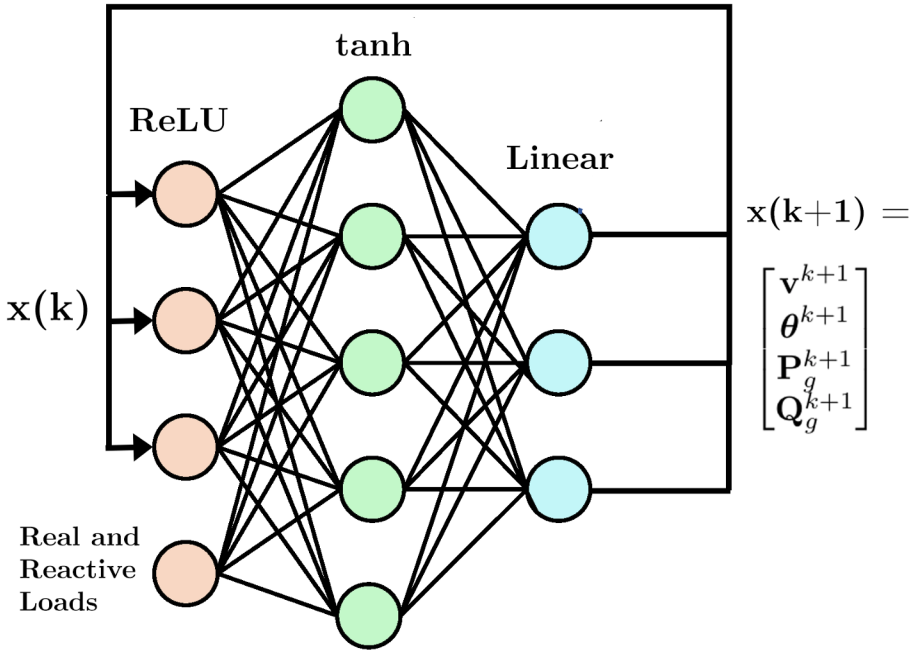
Emulate the solver itself?

→ Train a neural network to emulate iterations from the Matpower Interior Point Solver (MIPS)

Generate training data by storing MIPS iterations for various loading scenarios

Input: Current candidate solution x(k) and system loads (real and reactive)

Output: Next candidate solution x(k+1)

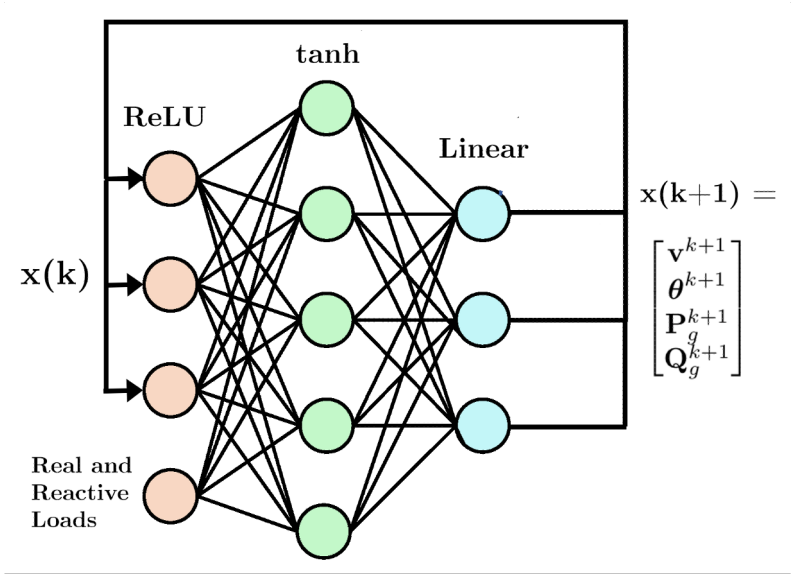


Idea: Use a NN to *iterate*

Both Newton-Raphson and Learning-Boosted Newton Raphson are fixed point iterations:

$$\mathbf{x}^{k+1} = F(\mathbf{x}^k)$$

But the learning-boosted method has an easier to evaluate *F()*



Key Idea:

It may be easier to learn what direction to move in than to directly learn an optimal solution from an initial point

[8] K. Baker, "A Learning-boosted Quasi-Newton Method for AC Optimal Power Flow," https://arxiv.org/abs/2007.06074, Workshop on machine learning for engineering modeling, simulation and design @ NeurIPS 2020.

Optimality and Feasibility gaps

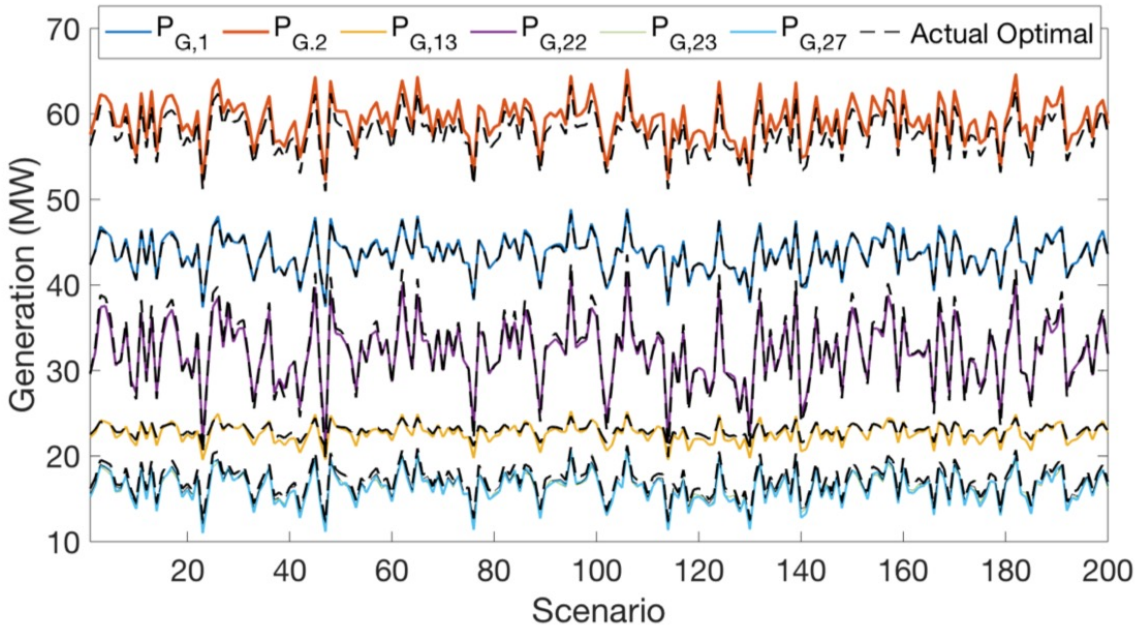


Fig. 2. Predicted generation values for 200 test scenarios (colors) and actual optimal values (black dashed line) for the IEEE 30-bus system.

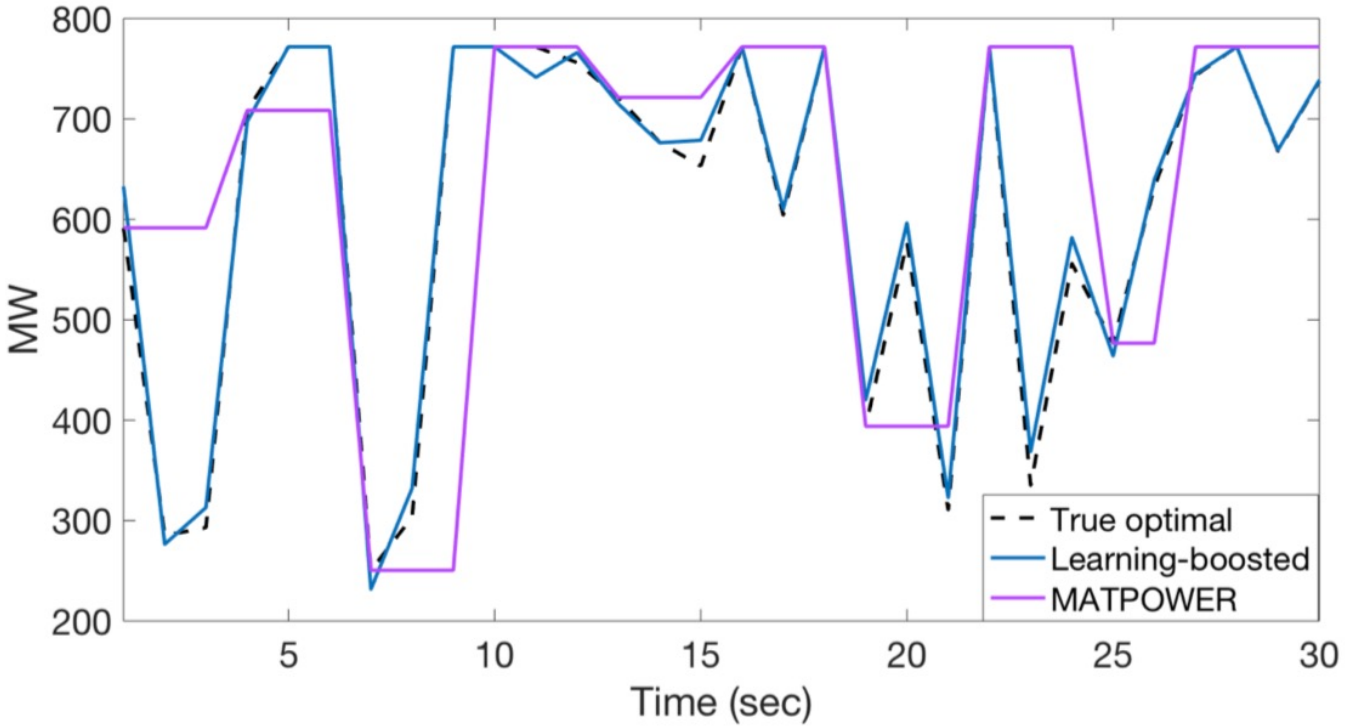
MEAN ABSOLUTE ERROR ACROSS TESTING DATASET

Case	MAE: Voltage Magnitude (pu)	MAE: Active Power (MW)	MAPE: Cost (%)
30-bus	0.004 pu	0.64 MW	0.29%
300-bus	0.009 pu	10.47 MW	0.65%
500-bus	0.099 pu	0.62 MW	0.66%
1,354-bus	0.019 pu	7.55 MW	1.16%

TABLE IV
TIME TO CONVERGENCE FOR EACH NETWORK.

Case	Mean Time (s)	Max Time (s)	Variance in Time (s)	Mean Speedup
30-bus MIPS	0.04	0.41	4.52e-04	
30-bus NN	0.06	0.42	3.53e-04	**-0.66x**
300-bus MIPS	1.09	10.87	2.09	
300-bus NN	0.03	0.42	0.001	**36.3x**
500-bus MIPS	1.46	2.96	0.49	
500-bus NN	0.08	0.45	3.25e-4	**18.3x**
1,354-bus MIPS	7.64	19.89	15.55	
1,354-bus NN	0.34	0.69	0.0016	**22.5x**

Real-time optimal solution tracking



Slack-bus generator tracking optimal solutions (black dashed line) for the IEEE-500 bus system.

MATPOWER MIPS solver takes a couple seconds to solve, meaning its optimal generation setpoints are outdated in between solutions.

The learning-boosted method provides approximate optimal solutions in less than a second, making it more appropriate for real-time optimization.

Convergence

Takes more iterations to converge because each iteration is using an approximate direction, but each iteration is faster

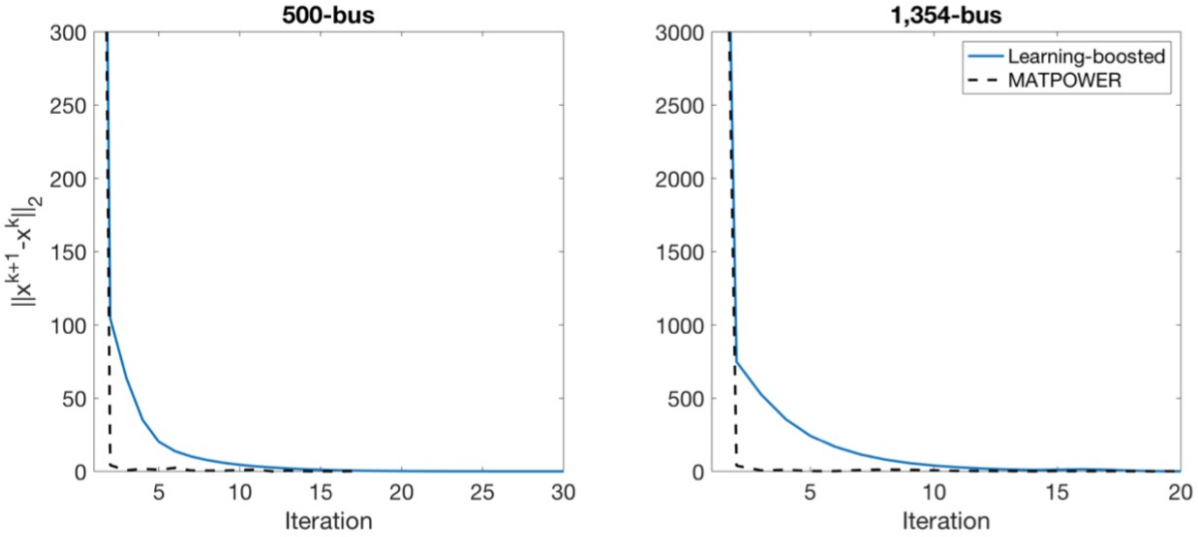
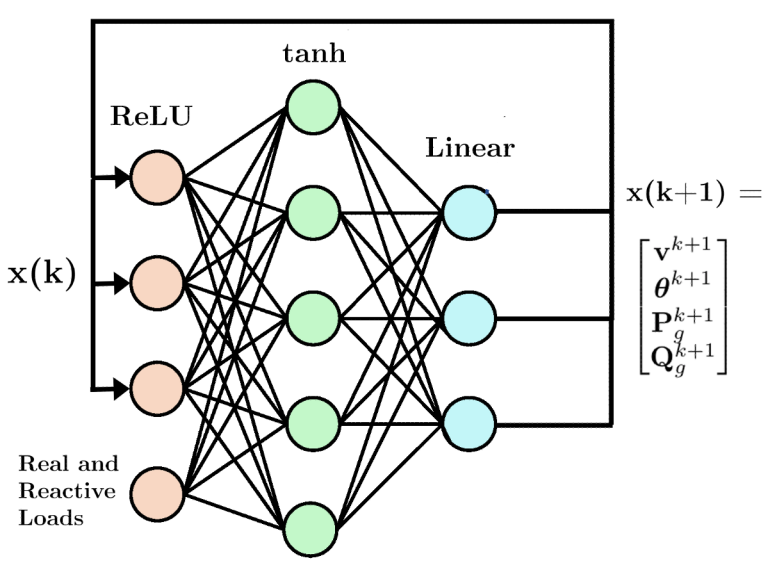


Fig. 3. Norm of $\mathbf{x}^{k+1} - \mathbf{x}^k$ for two scenarios in the 500-bus (left) and 1,354-bus (right) systems.

TABLE IV
TIME TO CONVERGENCE FOR EACH NETWORK.

Case	Mean Time (s)	Max Time (s)	Variance in Time (s)	Mean Speedup
30-bus MIPS	0.04	0.41	4.52e-04	
30-bus NN	0.06	0.42	3.53e-04	**-0.66x**
300-bus MIPS	1.09	10.87	2.09	
300-bus NN	0.03	0.42	0.001	**36.3x**
500-bus MIPS	1.46	2.96	0.49	
500-bus NN	0.08	0.45	3.25e-4	**18.3x**
1,354-bus MIPS	7.64	19.89	15.55	
1,354-bus NN	0.34	0.69	0.0016	**22.5x**

What about feasibility?



Subset of predicted variables

Power flow solver

Feasible and (generally) close to optimal solution

[9] K. Baker, "Emulating AC OPF solvers for Obtaining Sub-second Feasible, Near-Optimal Solutions," https://arxiv.org/pdf/2012.10031, Dec. 2020

How fast is this?

Network	OPF Type	Average Solve Time (s)	Percent Solve Success (%)
30-bus	AC OPF w/Flat Start	0.069 s	100%
	AC OPF w/DC Start	0.079 s	100%
	AC OPF w/PF Start	0.068 s	100%
	NN with PF	0.050 s	100%
	DC OPF	0.010 s	100%
500-bus	AC OPF w/Flat Start	3.26 s	100%
	AC OPF w/DC Start	3.34 s	100%
	AC OPF w/PF Start	3.15 s	100%
	NN with PF	0.12 s	100%
	DC OPF	0.12 s	100%
1354-bus	AC OPF w/Flat Start	20.12 s	59.4%
	AC OPF w/DC Start	11.63 s	59.4%
	AC OPF w/PF Start	10.64 s	59.8%
	NN with PF	0.25 s	98.00%
	DC OPF	0.33 s	100%

Solves in similar time to DC OPF and is **even faster** for large networks

Is **more robust** than other ways of warm-starting **AC OPF**

(many papers on this topic only perturb the base loading scenario +/-10 or 20%. Here, we perturbed the loads +/-40%, which created many “difficult to converge” scenarios)

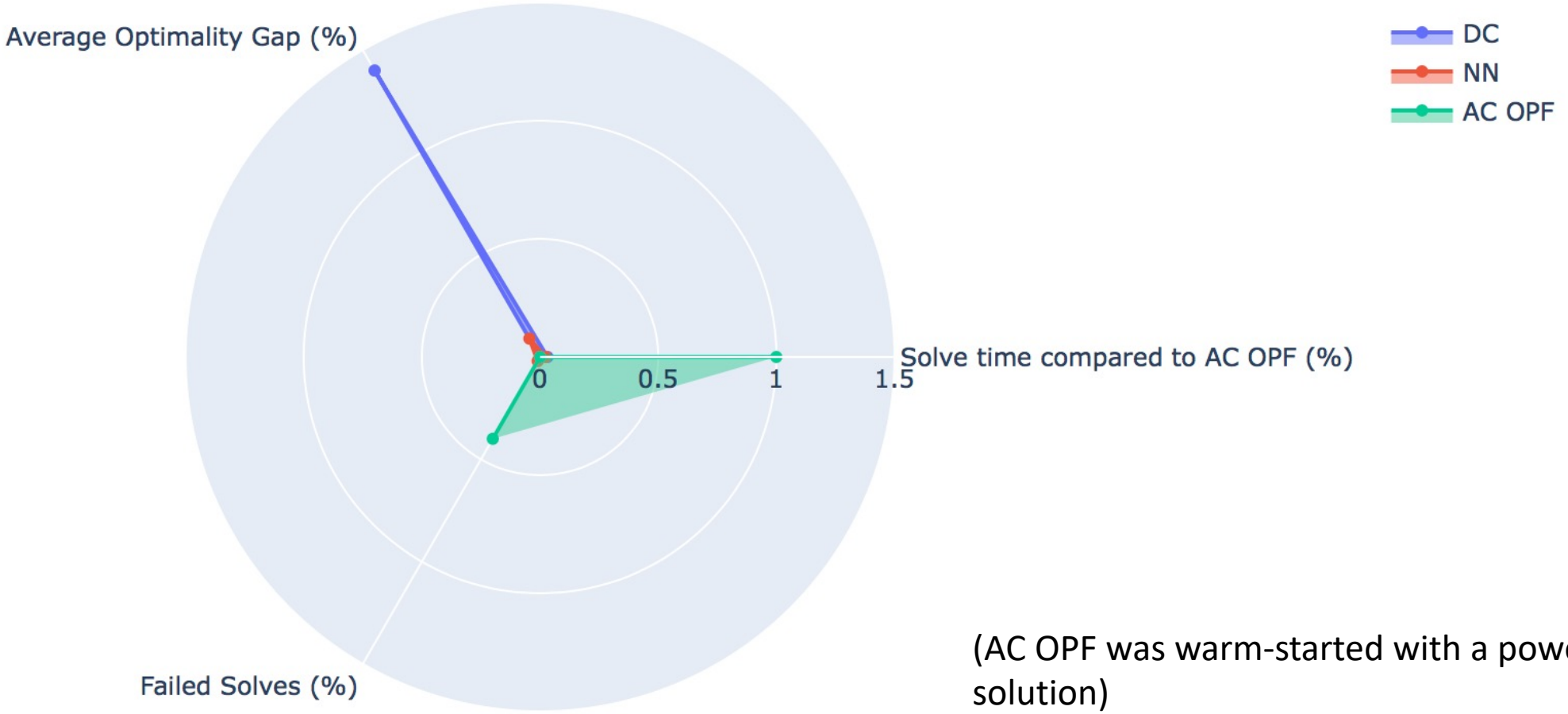
How optimal is this?

Network	Average Gap: NN	Average Gap: DC	Worst Gap: NN	Worst Gap: DC
30-bus	0.20%	1.46%	0.31%	1.82%
500-bus	2.70%	5.30%	10.12%	16.22%
1354-bus	0.09%	1.40%	0.20%	1.73%

(across 500 test loading scenarios)

DC OPF is often used to determine market clearing prices and LMPs. This is currently an “acceptable” optimality gap in industry.

Tradeoffs



(AC OPF was warm-started with a power flow solution)

Talk Outline

1. Harder, faster, better, stronger grids: the time is now

2. Learning **AC OPF** solutions very quickly

3. Using learning to obtain **distributed DC OPF** solutions faster

Joint work led by the National Renewable Energy Laboratory (NREL) in Golden, Colorado

"*Learning-Accelerated ADMM for Distributed DC Optimal Power Flow*,"
D Biagioni, P Graf, X Zhang, AS Zamzam, K Baker, J King, IEEE Control Systems Letters, 2020

Why DC OPF?

DC OPF is used to dispatch generators in many existing power markets

Telling operators to make the jump to AC OPF (especially using a neural network) might be a bit too unrealistic – maybe we can speed up DC OPF first

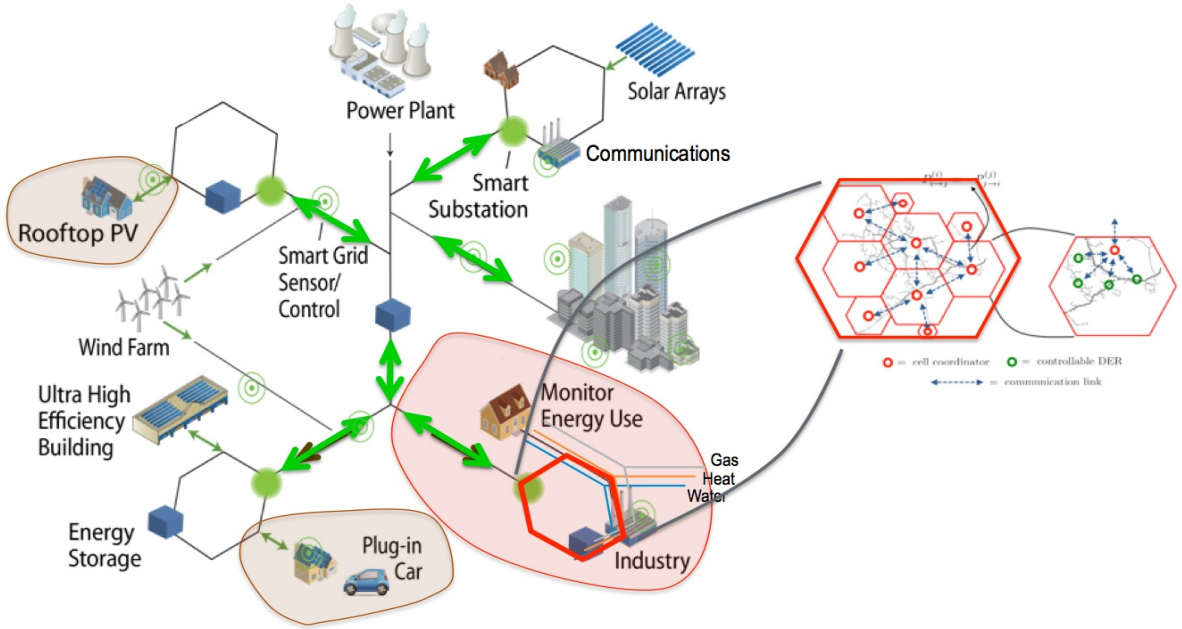
Balancing areas have to coordinate tie-line flows and still have to solve large-scale DC OPF problems – can we start solving these faster (faster than 5 minutes?)

Many benefits to more autonomy in grids

Central Question: How to use AI/ML to enable autonomous control of future power grids?

More specifically, how to enable control paradigms that:

- Are distributed and/or decentralized
- Preserve privacy of subsystems
- Leverage data-driven methods to increase performance, improve scalability, and the like



A spectrum of modalities for distributed control

Model-Based

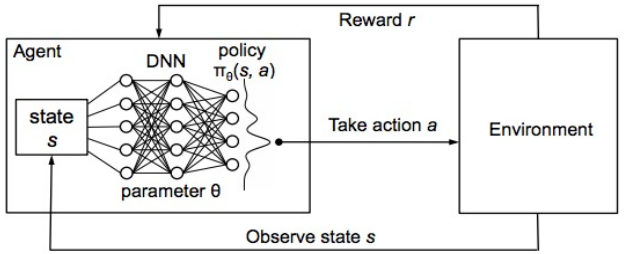
Hybrids

Model-Free

DC Optimal Power Flow

$$
\begin{aligned}
\min_{\mathbf{g},\boldsymbol{\theta}} \quad & \sum_{s=1}^{S} f_s(\mathbf{g}_s) \\
\text{s.t.} \quad & \mathbf{H}_s\boldsymbol{\theta}_s + \sum_u \mathbf{H}_{su}\boldsymbol{\theta}_{su} + \mathbf{g}_s - \mathbf{d}_s = \mathbf{0}_s, && \forall s \\
& \underline{\mathbf{g}}_s \le \mathbf{g}_s \le \overline{\mathbf{g}}_s, && \forall s \\
& \underline{\mathbf{f}}_s \le \mathbf{K}_s\boldsymbol{\theta}_s + \sum_u \mathbf{K}_{su}\boldsymbol{\theta}_{su} \le \overline{\mathbf{f}}_s, && \forall s \\
& \boldsymbol{\theta}_{su} = \mathbf{E}_{su}\boldsymbol{\theta}_u && \forall s, u
\end{aligned}
$$

Reinforcement Learning



	Model-Based	Model-Free
Enforces physical constraints	Yes	No
Convex optimization	Yes	No
Distributed (local) decisions	ADMM	Maybe
Preserve privacy	ADMM	Maybe
Learn directly from data	No	Yes

A spectrum of modalities for distributed control

Model-Based

DC Optimal Power Flow

$$\min_{\mathbf{g},\boldsymbol{\theta}} \quad \sum_{s=1}^{S} f_s(\mathbf{g}_s)$$

$$\text{s.t.} \quad \mathbf{H}_s\boldsymbol{\theta}_s + \sum_u \mathbf{H}_{su}\boldsymbol{\theta}_{su} + \mathbf{g}_s - \mathbf{d}_s = \mathbf{0}_s, \quad \forall s$$

$$\underline{\mathbf{g}}_s \le \mathbf{g}_s \le \bar{\mathbf{g}}_s, \quad \forall s$$

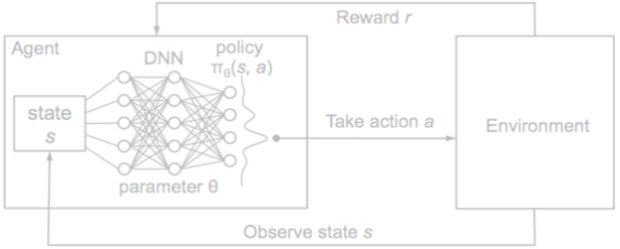
$$\underline{\mathbf{f}}_s \le \mathbf{K}_s\boldsymbol{\theta}_s + \sum_u \mathbf{K}_{su}\boldsymbol{\theta}_{su} \le \bar{\mathbf{f}}_s, \quad \forall s$$

$$\boldsymbol{\theta}_{su} = \mathbf{E}_{su}\boldsymbol{\theta}_u \quad \forall s,u$$

Hybrids

Learning-Accelerated ADMM (This Talk)

Reinforcement Learning



Model-Free

	Model-Based	Model-Free
Enforces physical constraints	Yes	
Convex optimization	Yes	
Distributed (local) decisions	ADMM	
Preserve privacy	ADMM	
Learn directly from data		Yes

Centralized DC OPF : Linearized AC OPF

Cost minimization

$$\min_{\mathbf{g},\boldsymbol{\theta}} \quad \sum_{s=1}^{S} f_s(\mathbf{g}_s)$$

Power balance

$$\text{s.t.} \quad \mathbf{H}_s\boldsymbol{\theta}_s + \sum_u \mathbf{H}_{su}\boldsymbol{\theta}_{su} + \mathbf{g}_s - \mathbf{d}_s = \mathbf{0}_s,$$

Generation limits

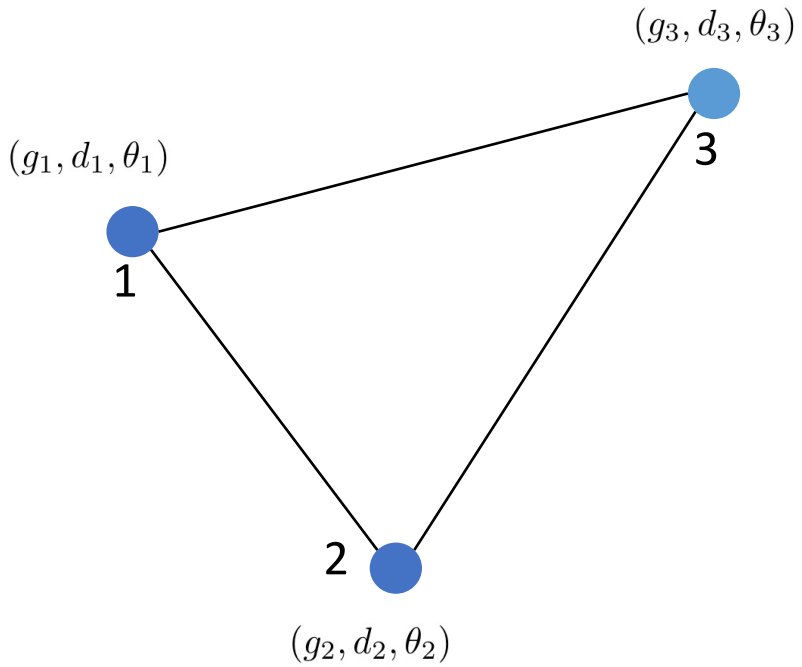
$$\underline{\mathbf{g}}_s \leq \mathbf{g}_s \leq \overline{\mathbf{g}}_s,$$

Voltage angle limits

$$\underline{\mathbf{f}}_s \leq \mathbf{K}_s\boldsymbol{\theta}_s + \sum_u \mathbf{K}_{su}\boldsymbol{\theta}_{su} \leq \overline{\mathbf{f}}_s,$$

Bus partitioning

$$\boldsymbol{\theta}_{su} = \mathbf{E}_{su}\boldsymbol{\theta}_u$$



ADMM for Distributed DC OPF

Augmented Lagrangian

$$\{\mathbf{g}_s^{(k+1)}, \boldsymbol{\theta}_s^{(k+1)}, \boldsymbol{\theta}_{su}^{(k+1)}\} = \underset{\mathbf{g}_s, \boldsymbol{\theta}_s, \boldsymbol{\theta}_{su} \in \mathcal{C}_s}{\operatorname{argmin}} \; f_s(\mathbf{g}_s) \quad + \sum_u \boldsymbol{\lambda}_{su}^{(k)T} \left(\boldsymbol{\theta}_{su} - \overline{\boldsymbol{\theta}}_{su}^{(k)}\right) + \frac{\rho}{2} \left|\left|\boldsymbol{\theta}_{su} - \overline{\boldsymbol{\theta}}_{su}^{(k)}\right|\right|_2^2$$

Consensus update

$$\overline{\boldsymbol{\theta}}_{su}^{(k+1)} = \frac{1}{2}\left(\boldsymbol{\theta}_{su}^{(k+1)} + \mathbf{E}_{su}\boldsymbol{\theta}_u^{(k+1)}\right)$$

Dual update

$$\boldsymbol{\lambda}_{su}^{(k+1)} = \boldsymbol{\lambda}_{su}^{(k)} + \rho\left(\boldsymbol{\theta}_{su}^{(k+1)} - \overline{\boldsymbol{\theta}}_{su}^{(k+1)}\right)$$

Private decisions

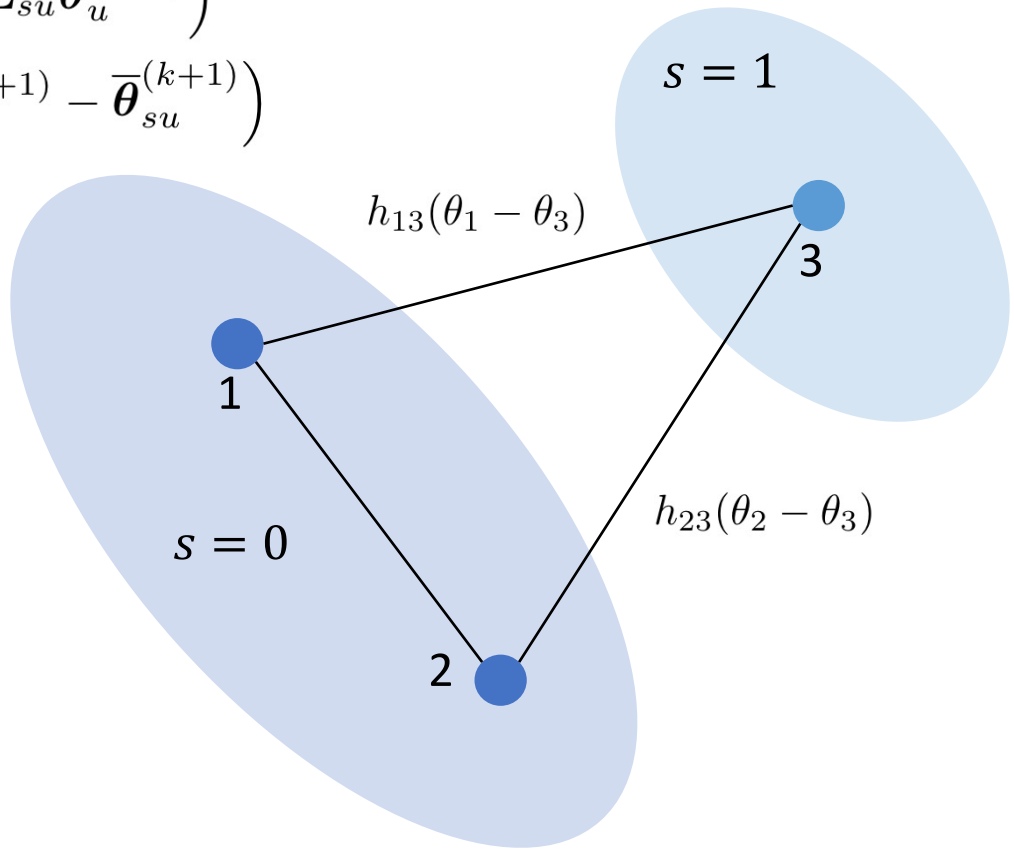
- Generation
- Voltage angles for internal buses

Public decisions

- Voltage angles for partition buses (via shared lines)

Each partition considers external voltage angles on tie lines as a local decision variable, one copy per partition

ADMM forces consensus on these shared variables



ADMM as Dynamical System

ADMM is an iterative algorithm

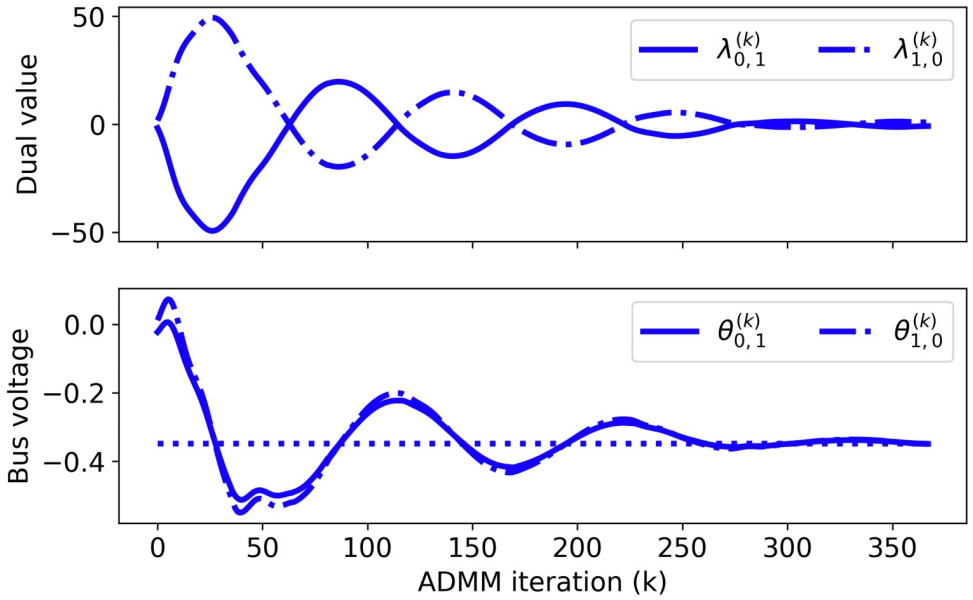
- Known to be slow esp. for high accuracies, and
- Rate is sensitive to step size parameter (ρ)
- Robust to perturbations under mild conditions

When we look at primal and dual variables as a function of iteration, it looks like a steady-state dynamical system

Convergence ⇔ steady state of an ODE (?)

(There is theory to support this)

So, why not try to predict these converged values and speed up convergence by appropriate warm starting?



Core Idea: Predict converged ADMM values given the first *K*

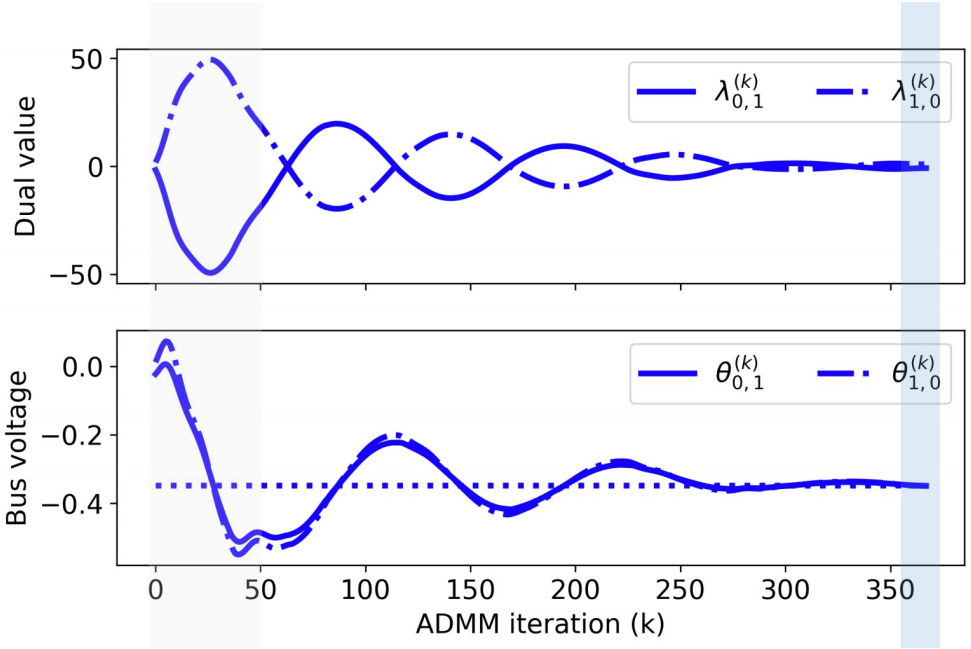
We can treat this as a supervised learning problem

INPUTS: A sequence of ADMM iteration values

OUTPUTS: Converged values

How to generate training data?

- Learn from the past: use fully converged ADMM iterations
- Learn from an oracle: solve the centralized problem to get optimal primal/dual values



Learning-Accelerated ADMM

Algorithm 1: Learning Accelerated-ADMM

Input: DC-OPF ADMM problem specification;
K-step RNN to predict converged $\boldsymbol{\lambda}^*_{su}, \overline{\boldsymbol{\theta}}^*_{su}$

Output: Distributed solution to DC-OPF

Set $k \leftarrow 0$

while *ADMM not converged* **do**

  if $k = K$ **then**

    Predict converged values: $\boldsymbol{\lambda}^*_{su}, \overline{\boldsymbol{\theta}}^*_{su}$

    Overwrite dual variables: $\boldsymbol{\lambda}^{(k)}_{su} \leftarrow \boldsymbol{\lambda}^*_{su}$

    Overwrite consensus variables: $\overline{\boldsymbol{\theta}}^{(k)}_{su} \leftarrow \overline{\boldsymbol{\theta}}^*_{su}$

  Execute ADMM iteration

  $k \leftarrow k + 1$

end

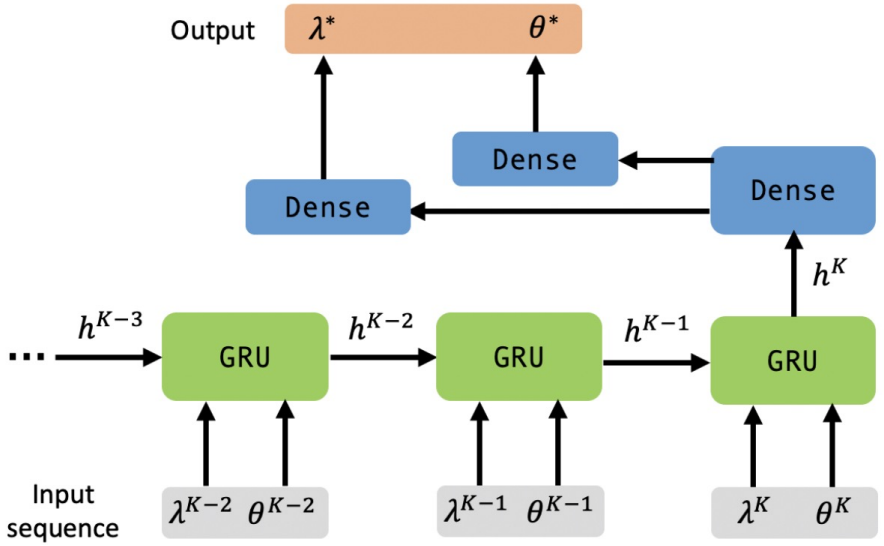


Fig. 9. RNN architecture consists of a single Gated Recurrent Unit layer operating on concatenated inputs, followed by dense prediction layers with two-headed linear outputs.

Results on three networks

(K = 4; predicted values injected in fourth iteration)

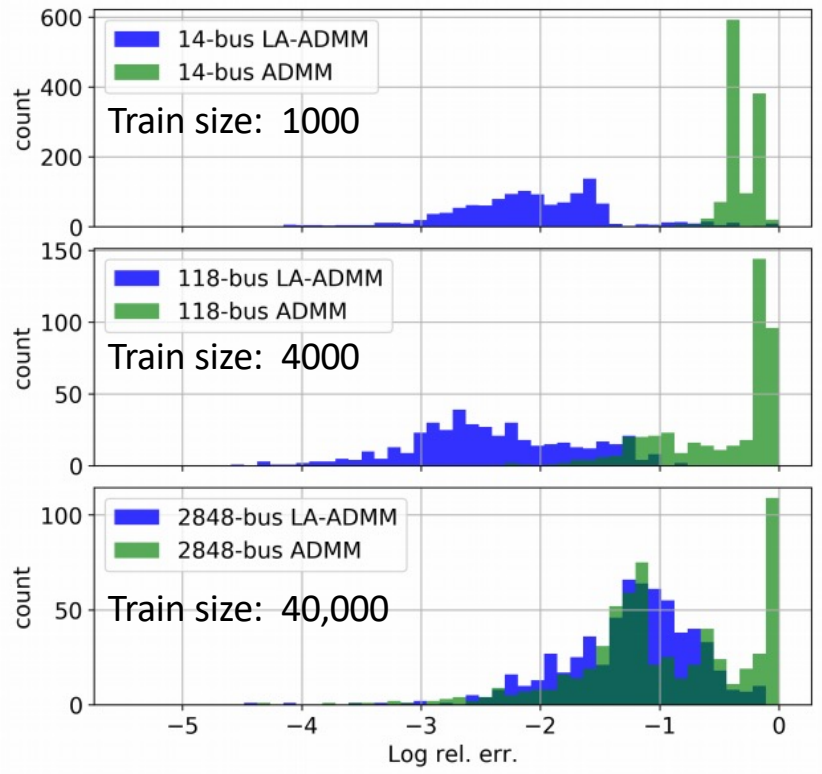


Fig. 4. Histograms showing the $\log_{10}$ relative error in the objective cost, compared with the optimal value, for ADMM and LA-ADMM after 100 iterations. LA-ADMM significantly accelerates convergence for the 14- and 118-bus test cases and mitigates the worst-case examples in the 2848-bus examples.

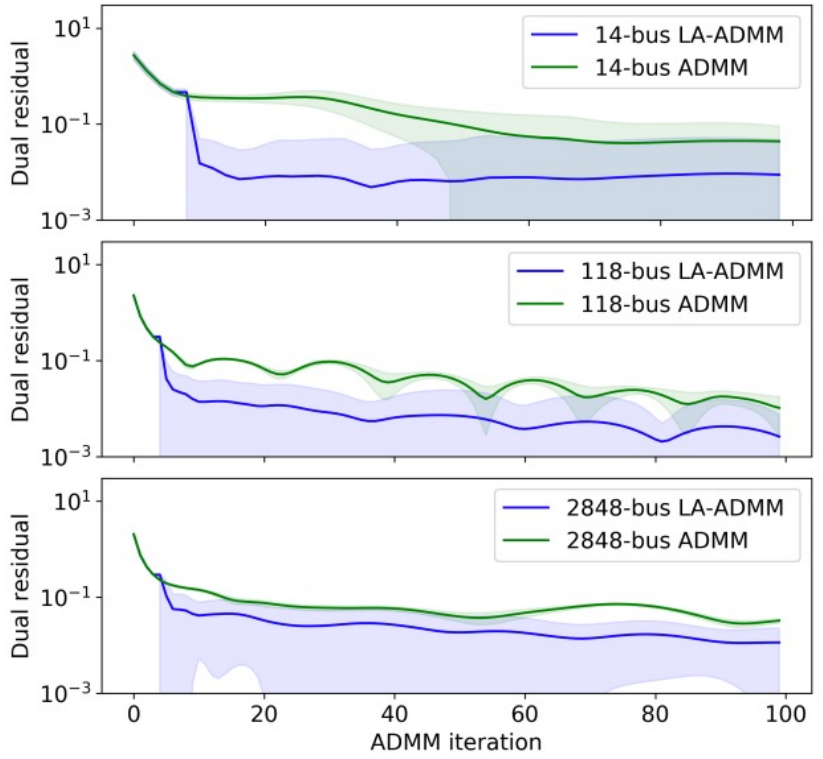
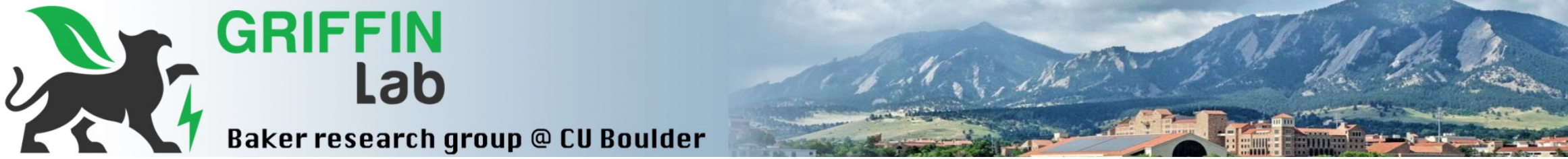


Fig. 5. Residual error as function of ADMM iteration. Solid lines indicate the mean across all runs in the test set while the shaded areas indicate ± 1 standard deviation from the mean.

Conclusions

Inefficient grid operations results in increased emissions, cost, and decreased reliability

In addition to upgrading and installing additional hardware, upgrading the algorithms can help us achieve better grid operations



Thank you! Questions?

Kyri Baker, www.kyrib.com
Kyri.baker@colorado.edu