Bridging Deep Learning and Electric Power Systems

Priya L. Donti

CMU-CS-22-142

August 2022

Computer Science Department, School of Computer Science
Department of Engineering & Public Policy, College of Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

J. Zico Kolter, Co-chair	*Carnegie Mellon University*
Inês Azevedo, Co-chair	*Stanford University*
Jeff Schneider	*Carnegie Mellon University*
M. Granger Morgan	*Carnegie Mellon University*
Yoshua Bengio	*Université de Montréal*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the U.S. Department of Energy Computational Science Graduate Fellowship (DE-FG02-97ER25308), the National Science Foundation Graduate Research Fellowship Program (DGE1252522), the Siebel Scholars Program, the Center for Climate and Energy Decision Making through a cooperative agreement between the National Science Foundation and Carnegie Mellon University (SES-00949710), and the National Science Foundation Expedition on Expanding the Horizons of Computational Sustainability (CCF-1522054).

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government, or any other entity.

Keywords: machine learning, deep learning, implicit layers, forecasting, estimation, optimization, control, electric power systems, optimal power flow, climate change mitigation, climate change adaptation

To those working to tackle climate change

Abstract

Climate change is one of the most pressing issues of our time, requiring the rapid mobilization of many tools and approaches from across society. Machine learning has been proposed as one such tool, with the potential to supplement and strengthen existing climate change efforts. In this thesis, we provide several directions for the principled design and use of machine-learning-based methods (with a particular focus on deep learning) to address climate-relevant problems in the electric power sector.

In the first part of this thesis, we present statistical and optimization-based approaches to estimate critical quantities on power grids. Specifically, we employ regression-based tools to assess the climate- and health-related emissions factors that are used to evaluate power system interventions. We also propose a matrix completion-based method for estimating voltages on power distribution systems, to enable the integration of distributed solar power.

Motivated by insights from this work, in the second part of this thesis, we focus on the design of deep learning methods that explicitly capture the physics, hard constraints, and domain knowledge relevant to the settings in which they are employed. In particular, we leverage the toolkit of implicit layers in deep learning to design forecasting methods that are cognizant of the downstream (stochastic) decision-making processes for which a model's outputs will be used. We additionally design fast, feasibility-preserving neural approximators for optimization problems with hard constraints, as well as deep learning-based controllers that provably enforce the stability criteria or operational constraints associated with the systems in which they are deployed. These methods are directly applicable to problems in electric power systems, as well as being more broadly relevant for other physical and safety-critical domains.

While part two demonstrates how power systems can yield fruitful directions for deep learning research, in the last part of this thesis, we demonstrate vice versa how insights from deep learning can yield fruitful directions for research in power systems. Specifically, we show how methods inspired by the implicit layers literature can be used to assess policy-relevant inverse problems on the power grid. We further show how combining insights from implicit layers and adversarially robust deep learning can allow us to provide scalable heuristic solutions to two central problems in power systems – N-k security-constrained optimal power flow and stochastic optimal power flow – that have seldom been addressed at realistic scale due to their computational intractability.

Overall, this thesis demonstrates how bridging insights from deep learning and electric power systems can help significantly advance methods in both fields, in addition to addressing high-impact problems of relevance to climate action.

Acknowledgments

Research is fundamentally a reflection of people – the coming together of their ideas, influence, stories, and support. During my Ph.D., I have been lucky to benefit from the mentorship, collaboration, and friendship of many wonderful people, who have shaped both my work and the person I've become in doing it.

At the top of the list are my advisors, Zico Kolter and Inês Azevedo. Zico and Inês are both among the most brilliant people I know, while simultaneously being some of the most compassionate and kind-hearted. Zico's approach to melding insights from different bodies of work has significantly influenced my own interdisciplinary research approach, and I have always valued his ability to consistently energize research discussions (no matter how much progress I previously had or had not made) in service of fostering new ideas and actionable next steps. In addition, I have appreciated Zico's thoughtfulness in convening important cultural conversations within the research group, as well as his leadership in fostering inclusion at CMU. Inês' breadth of expertise in the climate and energy space is truly immense – spanning a multitude of sectors, geographies, and approaches – and I have benefited greatly from her cross-cutting perspectives and mentorship on these topics. I have also appreciated Inês' attentiveness to checking in with her advisees not only about research but also about life, which has helped me feel more comfortable in my own skin as a researcher and laid the groundwork for open and transparent communication. I am extremely grateful for all the time, insight, and wisdom that Zico and Inês have shared with me over the years, as well as their encouragement in enabling me to take risks and explore non-traditional pathways of work.

This thesis would also not have been possible without the support of my committee members, Granger Morgan, Yoshua Bengio, and Jeff Schneider. Granger's consistent nudges to justify "why machine learning" and succinctly communicate takeaways for decision-makers, as well as his pedagogy on policy-relevant topics, have decidedly helped mature my thinking in these areas. I've appreciated Yoshua's incisive questions regarding why and under what conditions different methods might work, as well as his efforts in mainstreaming topics relevant to climate change and social good within the machine learning community. While my interactions with Jeff began more recently, I have valued our exchanges on the commonalities and differences that emerge when using reinforcement learning and control across various safety-critical domains.

I would further like to thank several additional mentors for the guidance and resources they have provided me before and during my Ph.D. Carla Gomes has been an amazing role model for what it means to do computer science research that is both technically interesting and societally impactful, and I am grateful for her indefatigable mentorship of myself and many others across the computational sustainability community. Jay Apt and Granger's initiatives through the CMU Electricity Industry Center have played a crucial role in helping me

build deeper connections to players in the electric power sector, enabling me to get timely industry feedback and thereby shaping the trajectory of my work. I thank Jim Boerkoel for mentoring me on undergraduate research at Harvey Mudd, and Gopal Ramchurn for writing the "Putting the 'Smarts' into the Smart Grid" paper that first sparked my interest in my current line of research.

During my Ph.D., I was fortunate to find a group of collaborators eager to foster work at the intersection of climate change and machine learning, beginning with the writing of a comprehensive overview paper and later through the founding of Climate Change AI. I cannot even begin to articulate how lucky I feel to have worked with such incredible teammates. Thanks especially to David Rolnick and Lynn Kaack, my CCAI co-founders and now some of my dearest friends, for their tireless dedication, edifying research and policy discussions, careful thought leadership, and metaphorical "continuous Zoom calls" over the last three years. Thanks also to CCAI's other founding and early team members, Alex Lacoste, Andrew Ross, Anna Waldman-Brown, David Dao, Evan Sherwin, Karthik Mukkavilli, Kelly Kochanski, Konstantin Klemmer, Kris Sankaran, Natasha Jaques, Nikola Milojevic-Dupont, Sasha Luccioni, Sharon Zhou, and Tegan Maharaj. As CCAI has grown, it has additionally been a pleasure to work with Ankur Mahesh, Dea Bankova, Ebude Antem Yolande Ebong, Felipe Oviedo, Gen Patterson, Hari Prasanna Das, Issa Tingzon, Ján Drgoňa, Jeremy Irvin, Jesse Dunietz, Jess Fan, John Kieffer, Kai Jeggle, Kameliya Petrova, Kasia Tokarska, Katherine Stapleton, Kelton Minor, Kureha Yamaguchi, Lauren Kuntz, Marcus Voss, Maria João Sousa, Mark Roth, Meareg Hailemariam, Oli Mendivil Ramos, Peetak Mitra, Raphaela Kotsch, Simone Nsutezo Fobi, Wei-Wei Lin, and Yumna Yusuf, as well as several newer members. Thanks also to the members of the CCAI Advisory Board – Andrew Ng, Carla Gomes, Catherine Nakalembe, Claire Monteleoni, Craig Smith, Demis Hassabis, Felix Creutzig, Inês Azevedo, Jennifer Chayes, John Platt, Konrad Körding, Tobias Schmidt, Yoshua Bengio, and Zico Kolter – for invaluable input and guidance.

My Ph.D. research has benefited greatly from collaborations with both a number of the individuals above and many others from across disciplinary boundaries. I thank Brandon Amos, Bryan Wilder, Mahyar Fazlyab, Mel Roderick, and Po-Wei Wang for their collaboration on topics at the nexus of machine learning, optimization, and control. I've further been fortunate to learn from the power and energy systems expertise of several academic and industry collaborators, including through my internships at NREL and National Grid; many thanks to Aayushya Agarwal, Andrey Bernstein, Bingqing Chen, Dan Drew, Fraser McMillan, Jack Kelly, James Kelloway, Kyri Baker, Larry Pileggi, Lyndon Ruff, Mario Bergés, Rui Yang, Yajing Liu, and Yingchen Zhang. I have enjoyed working with Cyrus Hodes, Emma Strubell, George Kamiya, Marta Kwiatkowska, Nico Miailhe, Pete Clutton-Brock, Raja Chatila, and Virginia Dignum on topics of climate policy and impact assessment in the context of machine learning. Thanks also to Suman Ravuri for working with me on cli-

mate and weather topics during my internship at DeepMind, alongside Karel Lenc, Matthew Willson, Piotr Mirowski, Remi Lam, Shakir Mohamed, and Sims Witherspoon. Last but certainly not least, I thank Amy Wang, Marissa Liu, Neeraj Bedmutha, and Tom Wright for taking a chance on me as a "research supervisor" and for bringing such motivation to their work.

My time at CMU has been greatly enriched by the community across my two departments, advisor groups, and beyond. Many thanks to the members of LocusLab for providing feedback on my work, for the interesting research discussions spanning many different areas of machine learning, and for bringing so much mirth to our interactions. In particular, I thank Brandon Amos for catalyzing the research in differentiable optimization, for writing high-quality open-source code whose style I've learned a lot from, and for sharing stories of his eclectic adventures in GAN-land. I thank Eric Wong for many valuable conversations about research and navigating research careers, for not being afraid to "manage up," and for always finding the best restaurants during conferences. I thank Mel Roderick for his insightful, multi-faceted perspectives on what it means to "do good" given the skills and privileges we have. I thank Vaishnavh Nagarajan for his consistently high-quality feedback on research and presentations, for his caring mentorship of many within and outside the group, and for being CMU's number-one candid photographer. Thanks also to Gaurav Manek for managing the research cluster, and to many present and former members of LocusLab, including Alnur Ali, Anna Bair, Asher Trockman, Ashwini Pokle, Christina Baek, Chun Kai Ling, Dylan Sam, Ezra Winston, Filipe de Avila Belbute-Peres, George Haff, Huan Zhang, Jeremy Cohen, Jonathan Dinu, Josh Williams, Leslie Rice, Matt Wytock, Mihir Mongia, Mingjie Sun, Po-Wei Wang, Pratyush Maini, Rizal Fathony, Runtian Zhai, Sachin Goyal, Sam Sokota, Saurabh Garg, Shaojie Bai, Suvansh Sanjeev, Swami Gurumurthy, Victor Akinwande, Xiao Zhang, Yash Savani, Yiding Jiang, and Zhili Feng.

I thank the members of the INES research group and broader CEDM community for providing perspectives on a diverse range of climate and energy topics, and for bringing such deep policy and industry expertise to bear on these issues. I thank Evan Sherwin for his consistently thoughtful questions and feedback, and for providing significant input on how to properly compute marginal emissions factors. I thank Greg Schivley for the long pair programming sessions as we aimed to wrangle gigabytes of EPA data. I thank Jeremy Keen, Luke Lavin, and Sean Smillie for leading the Energy Club reading group. I think Vanya Britto for many productive virtual co-working sessions after the start of the COVID-19 pandemic. Thanks also to many other members of the INES research group and broader CEDM community, including Amanda Quay, Angelena Bohman, Brian Sergi, Brock Glasgo, Cristóbal de la Maza, Daniel Gingerich, Daniel Posen, Daniel Sun, Emily Grayek, Erin Mayfield, Evan Sherwin, Fan Tong, Gerad Freeman, Jake Ward, Jessica Lovering, Jorge Izar, Julian Lamy, Liza Reed, Long Lam, Lynn Kaack, Matt Babcock, Matt

Bruchon, Michael Craig, Michael Whiston, Nat Horner, Nichole Hanus, Nyla Khan, Parth Vaishnav, Peter Tschofen, Priyank Lathwal, Sara Schwetschenau, Sarah Robb, Sarah Troise, Shayak Sengupta, Sinnott Murphy, Thomas Deetjen, Tobi Adekanye, Yamit Lavi, and many others I have certainly missed.

My extended CSD and SCS communities have provided me with an incredible amount of support through the vicissitudes of Ph.D. research and coursework. I thank the members of these communities for the mentorship, for the board game nights, for performing together in musicals and revues, and for working together to improve diversity, equity, inclusion, and culture. A particular thanks to my officemate Ellen Vitercik for all the insightful conversations over the years; to Bailey Flanigan for her tireless work on the DEI course; to my fast first-year friends Ben Berg, Kai Ye, and Ziv Scully; to Sol Boucher for convening our dissertation working sessions; to my first-year mentor Deby Katz; and to my project and homework groups for intro ML and intermediate stats, including Amanda Coston, Charles Wu, Dimitris Konomis, Evren Gokcen, Laurie Jin, Min Lee, Rudina Morina, and Steven Dang.

I thank my EPP community for the late-night homework sessions, karaoke shenanigans, soccer games, and the D&D sessions that were almost-but-not-quite an escape from the policy-relevant questions we were studying. I would particularly like to thank my "Rockin EPP" cohort – Aman Tyagi, Bingyin Hu, Christophe Combemale, Gerad Freeman, Guannan He, Jake Ward, Jihoon Shin, Jorge Izar, Kristen Allen, Liza Reed, Nicole Racine, Patrick Funk, Ria Laurejis, Sarah Robb, Shayak Sengupta, Tobi Adekanye, and Vanya Britto – for their friendship and continued support both before and after quals.

I also thank the many CSD/SCS and EPP administrative staff without whom the whole operation would not run; in particular, many thanks to Adam Loucks, Angy Malloy, Ann Stetser, Catherine Copetas, Deb Cavlovich, Debbie Kuntz, Diana Rotondo, Diane Stidle, Elisabeth Udyawar, Lucas Valone, Sara Golembiewski, and Vicki Finney.

I have been fortunate to participate in several communities and initiatives focused on technology and/or social change that have decidedly shaped my thinking on these topics. I thank the team at Tech4Society for employing their skills to support grassroots initiatives in the Pittsburgh area and for being formative in my perspectives on what it means to do truly community-centric work. Thanks especially to my co-founders, Jesse Dunietz, João Martins, Lizzie Silver, Reuben Aronson, and Yuzi Nakamura; to my collaborators on policing equity projects, particularly Ashley Oldshue, Bonnie Fan, Emily Black, Fox, Jack (RIP), Josh Williams, Kristen Buse, Maria De-Arteaga, Michael Madaio, and Nicasio Ng; and the new crop of T4S leadership, notably Arpit Agarwal, Jessie Grosen, and KA Garrett. I also thank the team at Engineers for a Sustainable World for fostering sustainability education and for helping me begin my non-profit journey; I would particularly like to thank my Executive Director, brittany bennett, and my Chapter Relations Co-Director, Sophie Hopps-Weber.

I enjoyed running CompSustNet conferences and webinars with Aaron Ferber, Amrita Gupta, Bryan Wilder, Genevieve Flaspohler, Hari Prasanna Das, Kevin Winner, Lily Xu, Neal Jean, Neil Gaikwad, and Sebastian Ament.

I have further benefited from the support of many additional communities during and leading up to my Ph.D. I particularly want to thank the DOE CSGF fellowship and community for giving me the freedom to explore non-traditional directions, for exposing me to new avenues of work in high performance computing, and for the steadfast cameraderie and peer support. I also thank the Watson Fellowship and community for funding me to travel for a year to study smart grids in different countries and for always reminding me to stay curious.

I have enjoyed living with a lively community of peers during my time in Pittsburgh. I thank my housemates past and present for all the encouragement, deep conversations, meals, movie nights, and companionship (particularly during the pandemic). Thanks to my fellow "Party House" residents Alex LaGrassa, Alfredo Trejo, Annika Froese, EJ Jardas, Fern Paulino, Gabbi Guedes, Isaac Grosof, Jack Burgess, Jeemin Cha, Mahi Hardalupas, Octavio Mesner, Ray Schuur, and Ziv Scully, and my previous roommate, Honey Rosenbloom.

I am deeply grateful to my family – Amma, Naanna, Arun, and Meera Avva, as well as my (vast) extended family – for all their love and support, starting well before the beginning of my Ph.D. journey. Many thanks for investing so much time and money in my education; for instilling in me the importance of hard work, dedication, and empathy; for supporting my decision to "stay in school" for so long; and for looking out for my personal well-being.

To my partner, Shantanu, I give the utmost thanks. Thank you for encouraging me to achieve my goals (through both your words and your actions), for centering and grounding me, for growing and changing with me, and for staying by me throughout the past six years despite us being on opposite sides of the globe.

There are many, many people missing from the list above without whom this thesis wouldn't have been possible. To all my teachers, mentors, colleagues, friends, family, and others who have touched my life and supported me along this journey – from the bottom of my heart, thank you.

Contents

Chapter 1

Introduction

Addressing climate change will require deep cuts in greenhouse gas emissions over the next several decades, as well as concerted efforts to adapt to those impacts of climate change that society will face [IPC18; IPC22a]. Electric power systems will play a key role on both of these fronts. In particular, the electric power sector currently contributes about a quarter of global greenhouse gas emissions, necessitating a transition to renewable and low-carbon power; low-carbon power systems are further critical for decarbonization strategies in other sectors that aim to "electrify" fossil-fueled loads (such as passenger vehicles) [IPC22b]. As climate change induces greater weather extremes, it will also be necessary to increase the robustness, resilience, and reliability of power grids in the face of these extremes [Nat17; RF+21]. In practice, all of these developments require power grids to be operated at increasing speed and scale – for instance, to manage the time-varying nature of renewable energy sources such as solar and wind, or to explicitly account for different power system failure scenarios. As classical techniques have begun struggling to cope with these requirements, many in the electric power sector have started to look towards areas such as machine learning (ML) to provide more modern tools for forecasting, estimation, optimization, and control [Rol+22; PAW14; Ram+12].

While there are many power sector applications for which ML can readily be used today, there are also several key challenges that preclude its use in many workflows. For instance, most ML methods struggle to enforce the physics or hard constraints associated with the systems in which they operate; however, in the context of electric power systems, failure to do so can lead to increased costs or even large-scale blackouts. As a result, the development of hybridized techniques that bridge machine learning with relevant physics and domain knowledge [Wil+20] will be critical to furthering work in this area. Even in cases where ML methods themselves may not be best-suited for a particular problem, methodological insights *from* machine learning also may serve to strengthen traditional power systems analysis. For instance, a recent body of work in deep learning has focused on the scalable computation of implicit gradients [KDJ20], with insights that may prove particularly beneficial for power systems workflows relying on iterative optimization techniques.

In this thesis, we demonstrate how challenges in the electric power sector can provide interesting methodological directions for ML, with a specific focus on deep learning. We

also show, vice versa, how methodological insights from deep learning can strengthen power systems research. In both cases, we show how such work can further efforts to decarbonize and strengthen electric power systems, in support of societal climate change goals.

1.1 Contributions

1.1.1 Part I: Estimation tasks in power systems

In Part I of this thesis, we present two estimation-related analyses in power systems. While these analyses do not directly employ ML techniques, this work motivates our design of ML methods in later parts of the thesis – notably, by providing insights on relevant physics, hard constraints, and domain knowledge that are important to incorporate within ML-based workflows. Our contributions in this part of the thesis are as follows:

- Chapter 3 assesses the salience of different key assumptions that are made when estimating a power grid's emissions factors and using them to evaluate potential power system interventions in the PJM Interconnection. We find that assumptions regarding marginal vs. average emissions factors and the time-frame of analysis are particularly significant. Based on this analysis, we recommend that energy modelers and decision-makers carefully consider the assumptions employed within their estimates, and that decision-makers work to improve the availability of relevant data behind these estimates. This analysis also exemplifies the importance of evaluating the quality of estimates through the lens of the *decisions* for which they will be employed, rather than only via standard accuracy-related metrics (as further explored in Chapter 5).
- Chapter 4 presents a novel method to estimate unknown voltages on electric power distribution systems under low-observability conditions, in order to facilitate the integration of distributed energy resources (such as rooftop solar) on these systems. Our method, based on constrained matrix completion, achieves near-perfect voltage estimation performance across many low-observability regimes where standard least-squares-based methods cannot operate. This method demonstrates the efficacy of incorporating physical and structural information within standard statistical methods (as also explored in Chapters 6–8), and has potential implications for decisions regarding where and how many sensors should be installed on distribution systems.

1.1.2 Part II: Optimization-in-the-loop deep learning

In Part II of this thesis, we present novel deep learning-based methods that are able to satisfy the physics, hard constraints, and domain requirements associated with the settings in which they operate. In particular, we formulate these physics, hard constraints, and domain requirements as optimization problems, and leverage the toolkit of optimization layers to embed them within the design of deep learning methods. We call this paradigm "optimization-in-the-loop deep learning," and employ it in the contexts of forecasting, optimization, and control. We note that this paradigm is fundamentally interdisciplinary,

requiring collaboration between (e.g.) deep learning researchers and domain experts. Our contributions in this part of the thesis are as follows:

- Chapter 5 provides an approach for designing probabilistic forecasting models that are well-tuned for the decision-making processes that employ them. Specifically, we propose methods to encode knowledge of the decision-making process within the loss function of a neural network, using differentiable optimization. This can improve end-to-end performance in the real-world systems in which the resultant forecasts are used. We present three experimental evaluations of the proposed approach – a classical inventory stock problem, a real-world electrical grid scheduling task, and a real-world energy storage arbitrage task – and show that the proposed approach outperforms both traditional modeling and purely black-box policy optimization approaches.
- Chapter 6 presents an approach to construct fast, feasibility-preserving approximators for continuous optimization problems, combining insights from deep learning and optimization. This includes optimal power flow problems in power grids, which are generally slow to solve, but must be run at increasingly short timescales to accommodate time-varying renewable energy. We show the efficacy of this approach for both synthetic optimization tasks and the setting of AC optimal power flow, showing that our method can achieve near-optimal objective values while preserving feasibility.
- Chapters 7 and 8 provide frameworks for constructing nonlinear control policies, parameterized by neural networks, that nonetheless enforce enforce hard constraints associated with the systems in which they operate. In Chapter 7, we show how to construct deep reinforcement learning-based controllers with *provable* control-theoretic stability guarantees, and demonstrate on a range of synthetic tasks that such controllers improve performance over traditional robust control techniques while retaining similar guarantees. In Chapter 8, we employ similar frameworks within the context of two realistic energy optimization tasks, namely grid inverter control and energy-efficient building heating and cooling, and show that our methods are able to maintain important operational constraints while improving performance over existing methods.

1.1.3 Part III: Implicit differentiation in power systems

While the previous part demonstrates how power systems can yield fruitful directions for deep learning, in Part III, we show how insights from deep learning workflows can similarly yield insights for traditional power systems research. In particular, we show how combining insights from the literature on implicit layers, adversarially robust deep learning, and electrical engineering can provide novel and scalable approaches to address important power system optimization problems. Our contributions in this part of the thesis are as follows:

- Chapter 9 presents an implicit differentiation-based method to conduct data vulnerability assessments on the electric power grid. In particular, we formulate an inverse problem called "inverse optimal power flow" that relates publicly-available data to privately-held data; we then use fast implicit differentiation techniques to optimize this problem, and assess the extent to which privately-held data may be exposed. We find via experiments on synthetic systems that private electricity generation costs and (to some

extent) grid structural parameters may be exposed, with the aim of providing this input to decision-makers to inform robust market design and cybersecurity assessments.

- Chapter 10 presents a scalable approach to addressing stochastic optimal power flow and N-k security-constrained optimal power flow, two optimization problems that are particularly important for the integration of time-varying renewables and for the robust operation of power grids in the face of correlated failures due to climate extremes. We demonstrate the efficacy of this approach on realistic-scale (5,000- and 10,000-bus) systems, which have previously seldom been addressed due to high computational costs.

1.2 Summary of publications

The content of Part I appears in:

> Priya L. Donti, J. Zico Kolter, and Inês Lima Azevedo. "How Much Are We Saving After All? Characterizing the Effects of Commonly Varying Assumptions on Emissions and Damage Estimates in PJM." *Environmental Science & Technology* 53.16 (2019), 9905–9914.

> Priya L. Donti, Yajing Liu, Andreas J. Schmitt, Andrey Bernstein, Rui Yang, and Yingchen Zhang. "Matrix Completion for Low-Observability Voltage Estimation." *IEEE Transactions on Smart Grid* 11.3 (2019), 2520–2530.

The content of Part II appears in:

> Priya L. Donti, Brandon Amos, and J. Zico Kolter. "Task-based End-to-End Model Learning in Stochastic Optimization." *Advances in Neural Information Processing Systems*. 2017, 5490–5500.

> Priya L. Donti*, David Rolnick*, and J. Zico Kolter. "DC3: A Learning Method for Optimization with Hard Constraints." *International Conference on Learning Representations*. 2021.

> Priya L. Donti, Melrose Roderick, Mahyar Fazlyab, and J. Zico Kolter. "Enforcing Robust Control Guarantees within Neural Network Policies." *International Conference on Learning Representations*. 2021.

> Bingqing Chen*, Priya L. Donti*, Kyri Baker, J. Zico Kolter, and Mario Bergés. "Enforcing Policy Feasibility Constraints through Differentiable Projection for Energy Optimization." *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*. 2021, 199–210.

The content of Part III appears in:

Priya L. Donti, Inês Lima Azevedo, and J. Zico Kolter. "Inverse Optimal Power Flow: Assessing the Vulnerability of Power Grid Data." *NeurIPS Workshop on AI for Social Good* (2018).

Priya L. Donti*, Aayushya Agarwal*, Neeraj Vijay Bedmutha, Larry Pileggi, and J. Zico Kolter. "Adversarially Robust Learning for Security-Constrained Optimal Power Flow." *Advances in Neural Information Processing Systems* 34 (2021), 28677–28689.

Aayushya Agarwal, Priya L. Donti, J. Zico Kolter, and Larry Pileggi. "Employing Adversarial Robustness Techniques for Large-Scale Stochastic Optimal Power Flow." *Power Systems Computation Conference* (2022).

The following publications from my Ph.D. provide overviews on topics at the intersection of climate change and ML, and do not explicitly appear in the remainder of this thesis:

David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Sasha Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla P. Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. "Tackling Climate Change with Machine Learning." *ACM Computing Surveys* 55.2 (Feb. 2022, preprint 2019).

Priya L. Donti and J. Zico Kolter. "Machine Learning for Sustainable Energy Systems." *Annual Review of Environment and Resources* 46 (2021), 719–747.

Peter Clutton-Brock*, David Rolnick*, Priya L. Donti*, Lynn H. Kaack*, Tegan Maharaj, Alexandra Sasha Luccioni, Hari Prasanna Das, Cyrus Hodes, Virginia Dignum, Marta Kwiatkowska, Raja Chatila, and Nicolas Miailhe. *Climate Change and AI: Recommendations for Government Action.* Tech. rep. Global Partnership on AI, 2021.

Lynn H. Kaack, Priya L. Donti, Emma Strubell, George Kamiya, Felix Creutzig, and David Rolnick. "Aligning Artificial Intelligence with Climate Change Mitigation." *Nature Climate Change* (2022), 1–10.

I also co-authored the following publication, which introduces an implicit layer for maximum satisfiability solving:

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. "SATNet: Bridging Deep Learning and Logical Reasoning using a Differentiable Satisfiability Solver." *International Conference on Machine Learning.* 2019.

Chapter 2

Background and Preliminaries

This thesis employs ideas from deep learning and implicit layers to address climate-relevant problems in electric power systems, and is aimed at multiple audiences hailing from the various fields represented therein. In this chapter, we give high-level background on some of the relevant topics, with the aim of providing readers with a common foundation on which to approach the content of this thesis.

2.1 Machine learning

Machine learning can be viewed as a form of data-driven programming that automatically learns programs based on examples.[1] While there are many different types of machine learning techniques, at their core, most ML algorithms are based upon just three components:

1. A *model* or *hypothesis class* that specifies the set of functions the ML algorithm can represent. Informally, this can be thought of as the "skeleton" of the program that the algorithm produces. These models often have *free parameters* that can be adjusted to specialize to the task at hand.
2. An *objective* or *loss function*[2] that specifies desirable behavior of the model.
3. An *optimization* or *training procedure* that specifies how to choose or adjust the parameters of the model in order to improve performance on the objective.

As an illustrative example, consider the example of predicting electricity consumption for a given region throughout the day tomorrow. One might approach this by collecting data detailing electricity consumption during past days, along with features that correlate with this consumption (such as temperature or day of the week); they could then write an ML algorithm that attempts to find correlations between the past consumption data and their corresponding features, and then uses these correlations to predict future consumption given (estimates of) the relevant features at future times. In this example, the *model* might be a low-degree polynomial of temperature and day of the week, where the free parameters

[1] Note: This section is largely adapted from Donti and Kolter [DK21].

[2] For loss functions, by convention, lower values are considered better.

are the coefficients of the polynomial. The *objective* might be to minimize the absolute error of the predictions of future electricity consumption.[3] The *training procedure* might involve making small incremental adjustments to the parameters to iteratively improve the objective (e.g., via *gradient descent*, a common procedure in many ML algorithms).

Before diving further into the details, we first clarify machine learning's relationship to other relevant fields. Machine learning is a sub-field of artificial intelligence (AI), which describes a set of techniques concerned with making computers perform complex tasks traditionally associated with human intelligence (such as perception, speech, movement, and logical reasoning) [Rus10]. ML also shares a deep relationship to statistics, with a significant overlap in both history and techniques; the difference between these two fields is largely one of perspective [Bre+01], as machine learning is generally more concerned with *performance* on the task at hand (i.e., optimizing the objective), whereas statistics is generally more concerned with discovering *"truths"* in the underlying data (i.e., understanding the quality of the learned model parameters). Machine learning also has close ties to optimization (given its reliance on optimization procedures) and control theory (particularly in the case of reinforcement learning, as discussed below).

2.1.1 Notable paradigms

Within machine learning, there are different paradigms describing different ways in which ML can be employed. Particularly notable paradigms include:

Supervised learning. In supervised learning, the goal is for the ML algorithm to learn a function mapping from inputs (*features*) to their desired outputs (*labels*) given some "supervision" on what these input-output pairs should look like. (The previously-described load forecasting setting, in which we provided input/output pairs to a machine learning algorithm, is an example of supervised learning approach.) This is referred to as *regression* when the outputs are continuous, and *classification* when the outputs are discrete.

Supervised learning has to date seen many successes in areas such as image classification, automated speech recognition, and machine translation. Unfortunately, this paradigm is not applicable in all settings: in many cases, it is prohibitively expensive to get enough labeled data to use supervised learning, or the system of interest involves a decision-making process that cannot be sufficiently described by single input/output pairs.

Unsupervised learning. Unlike supervised learning, which requires labeled examples, unsupervised learning requires only that we provide inputs to the machine learning algorithm, without any corresponding outputs. As there is no output to produce, the algorithm merely attempts to find some form of structure over the inputs. For instance, *clustering* techniques aim to group data into similar categories ("clusters"). *Dimensionality reduction* techniques aim to find a low-dimensional subspace that captures most of the variation in the data

[3] It is worth noting that performance on the *training data* used to construct the model is related to, but different from, performance on data the model has not yet seen. In particular, it is important to avoid *overfitting*, which refers to the phenomenon where a data-driven model makes good predictions on training data, but does not *generalize* well to unseen data.

(similarly to techniques such as principal component analysis). *Generative modeling* aims to learn a probabilistic model representing the distribution of the underlying data, with the idea of sampling from this model to generate new data (e.g., generate a new picture of a horse given a dataset of horse pictures).

While useful for analyzing, partitioning, and/or generating data, a notable caveat with unsupervised methods is that key attributes (such as the number of clusters or dimensions, or parameters of the underlying dataset) are typically picked by the algorithm designer. As a result, the learned outputs may be an artifact of the algorithm itself rather than representing "true" attributes of the underlying data.

Reinforcement learning. Reinforcement learning (RL) is a paradigm where an agent must learn how to act in a sequential environment to maximize some reward [SB18]. The strategy the agent learns is called its *policy*. Unlike the previous paradigms of supervised or unsupervised learning, RL algorithms do not (usually) operate over a fixed dataset, but instead within a setting where the algorithm can take an action that affects future states of some system. This is a similar setting as considered in adaptive control, and indeed these fields have a great degree of shared history, though they often differ in the types of structural assumptions they make about the underlying system (see [Buş+18]). RL is also closely related to the area of agent-based modeling (ABM), though ABMs often involve manually specifying behavioral rules, whereas RL aims to learn such rules automatically.

While RL has seen a number of notable successes, such as beating humans in complex games such as Go [Sil+16], there have been comparatively few deployments of RL on real-world physical systems. This stems from the fact RL agents must often act sub-optimally (potentially for a long time) during the learning process; most successful applications thus require, at the very least, a (realistic) simulation environment on which to train the agent.

Other paradigms. The paradigms described above, while important, are not exhaustive. For instance, *semi-supervised learning* represents a mix between supervised and unsupervised learning paradigms. While supervised and unsupervised learning typically occur in the *offline* or *batch* setting (where the ML algorithm is provided a complete dataset up front), many algorithms must also operate in *online* or *streaming* settings (where datapoints arrive one at a time, and the algorithm must make a prediction before receiving the next datapoint). While our discussion above has implicitly assumed that ML models are trained in settings similar to those in which they ultimately operate, the paradigms of *transfer learning*, *multi-task learning*, and *meta-learning* focus on the ability of ML models to generalize to new settings and new kinds of tasks. For a more in-depth discussion of different ML paradigms, please see Bishop and Nasrabadi [BN06] and Murphy [Mur22].

2.1.2 Strengths, limitations, and alternatives

As perhaps hinted at by the discussion above, ML is a powerful paradigm for data-driven programming, and can facilitate the analysis of large and heterogeneous data streams in cases where they would be impossible to analyze manually. While conceptually simple, this paradigm can manifest itself in many forms. For instance, ML can be used to "scale

human intuition" by identifying patterns in comparatively small amounts of labeled data, and then applying these learned patterns at a much larger scale. It can also be used to glean actionable insights from unstructured data streams such as satellite imagery or text documents, and to optimize complex systems based on observations of the system's behavior, among many other applications.

At the same time, ML has a number of major limitations. For instance, ML algorithms are extremely dependent on the quality of the data they receive ("garbage in, garbage out"). More broadly, ML is fundamentally an amplifier of the systems in which it is deployed, meaning that while it is capable of amplifying the benefits of these systems, it is also equally capable of exacerbating biases [Meh+21], inequities [GD20], market failures [Vic19], and other systemic effects [Kaa+22] through its data, design, and applications. ML methods also generally assume that the data on which they are trained and tested are similar in distribution to each other, and have difficulty dealing with scenarios where this is not the case (known as *distribution shift*). ML tends to have difficulty enforcing any physics or hard constraints associated with the domains in which it operates (as further addressed in this thesis), and many methods also suffer from a lack of interpretability. In addition, recent trends show that the largest ML models are becoming increasingly computationally intensive (and thus financially costly) to run, which has implications for the accessibility of modern methods as well as implications for greenhouse gas emissions [Kaa+22; SGM19; Sch+20; Ben+21]. (Notably, many of these topics represent active areas of ML research.)

We note also that while ML is broadly powerful, complex or cutting-edge ML techniques may not always be best-suited or needed for every problem. For instance, linear regression may be a better alternative to more complex supervised ML techniques in cases where only small amounts of data are available, or where the structure of the relationships between the inputs is well-known. Techniques from classical control theory may be more appropriate than reinforcement learning when the dynamics of the underlying environment are simple or well-structured; techniques from agent-based modeling may be more appropriate when the rules governing agent behavior are well-known, and do not need to be learned.

In general, ML should be viewed not as a black box or a silver bullet, but as a tool to employ in a principled manner that is guided by an understanding of its strengths, limitations, and underlying assumptions, as well as of relevant technical and contextual considerations surrounding the problem at hand.

2.2 Deep learning and implicit layers

Deep learning [Goo+16; RS20] is currently one of the more prominent approaches to ML.[4] Using the framework presented in Section 2.1, deep learning approaches are characterized by the following components:

1. Model: Deep learning approaches employ a class of models called *artificial neural networks* (also often referred to as *neural networks*), which can be thought of as the composition of a sequence of functions (*layers*) with adjustable parameters.

[4]Note: Several sentences in this section are drawn verbatim from Donti and Kolter [DK21]. Parts of the subsection on implicit layers are adapted from Chen*, Donti*, Baker, Kolter, and Bergés [Che+21].

3. Training procedure: The model's parameters are optimized using an iterative procedure called "backpropagation and gradient descent." Given an initial set of neural network parameters, this procedure involves first efficiently computing gradients of the chosen neural network loss function with respect to all parameters (via *backpropagation*), then updating the parameters using these gradients (via *gradient descent* or its variants), and repeating until some stopping condition is reached.

The loss function used to evaluate performance can be adapted to the problem at hand, though there are "standard" losses for typical problems like classification.

This paradigm of composable layers trained via backpropagation and gradient descent has proven extremely powerful, capable of expressing very complex functions while also generalizing well in practice when presented with new data. In particular, deep learning methods have been widely applied within many different ML paradigms, including supervised, unsupervised, and reinforcement learning. In this section, we provide more details on neural network models and training procedures, as well as on a new class of neural network layers (called *implicit layers*) that we leverage in this thesis.

2.2.1 Neural network models

Let $h_\theta : \mathbb{R}^k \to \mathbb{R}^n$ be a neural network defined as the composition of L functions, such that for inputs $x \in \mathbb{R}^k$,

$$h_\theta(x) = h_{L,\theta_L} \circ h_{L-1,\theta_{L-1}} \circ \cdots \circ h_{1,\theta_1}(x). \tag{2.1}$$

Here, each h_{i,θ_i}, $i = 1, \ldots, L$ is a neural network layer with parameters θ_i, where h_{L,θ_L} is defined to have output dimension $\mathbb{R}^n$ and where h_{i,θ_i}, $i = 1, \ldots, L-1$ are defined such that the output dimension of h_{i,θ_i} is equal to the input dimension of $h_{i+1,\theta_{i+1}}$. We use θ to refer to the full collection of parameters of the neural network (i.e., $\theta_1, \ldots, \theta_L$). For L sufficiently large (e.g., $L \geq 4$), h_θ is considered a *deep* neural network (hence the term "deep learning").

The layers h_{i,θ_i} are generally chosen to be nonlinear, such that the resultant deep neural network is an expressive, nonlinear model. There are various standard kinds of layers that are often used in today's neural networks, based on functions such as rectified linear units (ReLUs), convolutions, sigmoid functions, and hyperbolic tangent functions. Several specialized forms of neural network models, or *architectures*, have also emerged over the years, such as generic feedforward networks (for unstructured data), convolutional networks (for image data), recurrent networks and transformer networks (for time-series or other sequential data), graph networks (for graph-structured data), and diffusion models (for generative modeling tasks).

2.2.2 Neural network training

We now describe the general procedure of training neural networks via backpropagation and gradient descent. For ease, we describe these concepts in the context of supervised learning.

Let $(x^{(j)}, y^{(j)})$, $j = 1, \ldots, m$ be a set of input-output pairs, comprising our training data. Further, let $\ell : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a loss function. As before, let h_θ be our neural network.

Our neural network training procedure then aims to solve the following problem:

$$\underset{\theta}{\text{minimize}} \sum_{j=1}^{m} \ell(h_\theta(x^{(j)}), y^{(j)}). \tag{2.2}$$

Solving this problem is known as *empirical risk minimization*. For deep learning, this problem is generally non-convex (as h_θ is generally a complex nonlinear function), and is therefore optimized with the aim of finding local (rather than global) minima.

In deep learning, empirical risk minimization is typically done using variants of an iterative algorithm called *gradient descent*. In particular, in gradient descent, given the current instantiation of parameters θ at a given iteration, the parameters are updated via

$$\theta \leftarrow \theta - \alpha \sum_{j=1}^{m} \nabla_\theta \ell(h_\theta(x^{(j)}), y^{(j)}), \tag{2.3}$$

where $\alpha > 0$ is a (potentially adaptive) design parameter called the *learning rate*. In practice, most modern deep learning algorithms use a variant called *stochastic gradient descent* (SGD), which only uses a subset of the training data for each update (in contrast to the full gradient descent update, which uses the full set of training data each time); more advanced versions of SGD, such as Adam [KB15], are also widely used today.

In all cases, implementing these updates requires computing the gradient $\nabla_\theta \ell(h_\theta(x, y))$ for each training sample (dropping the indices on x and y for brevity). These gradients are computed efficiently via *backpropagation*, which can be thought of as a computationally efficient application of the chain rule. In particular, let z_i and θ_i denote the outputs and parameters, respectively, of the layer h_{i,θ_i}, for $i = 1, \ldots, L$. (We note that $z_L \coloneqq h_\theta(x)$.) Assuming for simplicity that the parameters of each layer are disjoint, the gradient with respect to a given θ_i is then given by

$$\nabla_{\theta_i} \ell(h_\theta(x), y)) \coloneqq \frac{\mathrm{d}\ell(h_\theta(x), y))}{\mathrm{d}\theta_i} = \frac{\partial \ell(h_\theta(x), y))}{\partial \theta_i} + \frac{\partial \ell(h_\theta(x), y))}{\partial z_L} \frac{\mathrm{d}z_L}{\mathrm{d}z_{L-1}} \cdots \frac{\mathrm{d}z_{i+1}}{\mathrm{d}z_i} \frac{\mathrm{d}z_i}{\mathrm{d}\theta_i}. \tag{2.4}$$

Thus far, Equation (2.4) as written is just a direct application of the chain rule. However, since layer dimensions and subsequently the number of neural network parameters are often quite large, many of the terms in Equation (2.4) are generally large in size; thus, naively applying the chain rule can prove to be extremely time- and space-intensive. As such, efficient backpropagation entails employing two key tricks to make this computationally tractable:

- Jacobian-vector trick: The backpropagation algorithm entails cleverly ordering the computations in Equation (2.4) from left to right, as well as avoiding the explicit computation of expensive Jacobians of the form $\mathrm{d}z_{i+1}/\mathrm{d}z_i$ and $\mathrm{d}z_i/\mathrm{d}\theta_i$, in order to reduce the associated costs. To illustrate this, we note that for a neural network whose layer outputs z_i can be viewed as vectors (as is common), since $\ell(h_\theta(x), y))$ is a scalar, the gradient $\mathrm{d}\ell(h_\theta(x), y))/\mathrm{d}z_L$ is a vector; the product of this term with the Jacobian matrix $\mathrm{d}z_L/\mathrm{d}z_{L-1}$ is then a vector-Jacobian product, whose output is a vector; the product of *this* term with the Jacobian matrix $\mathrm{d}z_{L-1}/\mathrm{d}z_{L-2}$ is then *also* a vector-Jacobian product, whose output is a vector; and so forth. By always computing the terms in

Equation (2.4) from left to right, one can avoid computing expensive matrix-matrix products (e.g., of the form $(\mathrm{d}z_i/\mathrm{d}z_{i-1})(\mathrm{d}z_{i-1}/\mathrm{d}z_{i-2})$). Furthermore, many backpropagation approaches avoid *ever* solving for any of the Jacobians $\mathrm{d}z_{i+1}/\mathrm{d}z_i$ and $\mathrm{d}z_i/\mathrm{d}\theta_i$ explicitly, but instead simply directly compute the vector-Jacobian product (e.g., by finding an analytical expression for it); this further saves on computation time. (See also [GW08].)

- Caching intermediate quantities: Due to the modular nature of neural networks, most of the products in Equation (2.4) that are computed to obtain gradients with respect to some θ_i are also needed to obtain gradients with respect to θ_{i-1}, θ_{i-2}, and so forth. As such, most backpropagation approaches involve caching the results of relevant computations so that they can be reused. This trick improves the time complexity of backpropagation, with the tradeoff being increased space complexity (due to the fact that computational results must then be stored).

We will come back to the Jacobian-vector trick in the next section on implicit layers.

2.2.3 Implicit layers

Having discussed the basics of neural network models and training, we now discuss recent work presenting a new class of neural network layers, called *implicit layers* [KDJ20]. In particular, many of the layers commonly used within neural networks (e.g., ReLUs and convolutions) are based on *explicit functions* with a direct, closed-form mapping between inputs and outputs; these functions are cheap to compute, and admit analytical gradients for use within backpropagation. However, there has been increasing interest in enriching the toolkit of layers to capture *implicit functions*, whose outputs cannot be solved for in closed form and must therefore be obtained via an iterative solution procedure (e.g., Newton's method). This has included *optimization layers* [AK17; DK17; TSK18; DAK18; Wan+19; Agr+19; GHC21], differential equation-based layers [Che+18], and layers representing rigid-body physics [ABP+18], among other implicit layers [BKK19; LFK18; Jin+20].

Like all neural network layers, implicit layers must support a *forward procedure* to map from inputs to outputs, as well as a *backward procedure* to compute gradients of the outputs with respect to the inputs and layer parameters. As hinted at above, the forward procedure is implemented by using an iterative solution technique. While the backward procedure could in principle simply entail differentiating through the unrolled iterations of this solution procedure, in practice, this can be time-intensive, space-intensive, and numerically unstable. As such, the design of implicit layers generally involves writing down a set of equilibrium or fixed-point conditions for the implicit function under consideration, and then efficiently differentiating through these conditions using implicit differentiation techniques [KP12; DR09; GW08].

2.2.3.1 Example: Differentiable projection layer

As an example, consider the L_2-norm projection $\mathcal{P}_{\mathcal{C}} : \mathbb{R}^p \to \mathcal{C}$ that maps from some point in $\hat{u} \in \mathbb{R}^p$ to its closest point in some convex constraint set $\mathcal{C} \subseteq \mathbb{R}^p$ as follows:

$$\mathcal{P}_{\mathcal{C}}(\hat{u}) := \operatorname*{argmin}_{u \in \mathcal{C}} \frac{1}{2}\|u - \hat{u}\|_2^2. \tag{2.5a}$$

For simplicity, we consider the case where $\mathcal{C}$ characterizes linear constraints, i.e.,

$$\mathcal{C} \coloneqq \{u : Au = b, Gu \leq h\} \tag{2.5b}$$

for some $A \in \mathbb{R}^{n_{\text{eq}} \times p}$, $b \in \mathbb{R}^{n_{\text{eq}}}$, $G \in \mathbb{R}^{n_{\text{ineq}} \times p}$, and $h \in \mathbb{R}^{n_{\text{ineq}}}$. The implicit layer based on Equation (2.5) would have $\hat{u}, A, b, G, h$ as its inputs and/or parameters, and u (plus possibly the dual variables of the optimization problem) as its output. (While we assume for simplicity of exposition that all parameters are adjustable, it would also be possible to set certain parameters to fixed values as dictated by the needs of the specific setting.)

We note that the resultant problem is a convex quadratic optimization problem. The forward procedure can then be implemented by simply solving the optimization problem, e.g., using standard convex optimization solvers. Perhaps less evidently, it is also possible to construct a backward procedure for this problem by using the implicit function theorem, as described in previous work (e.g., [AK17; Agr+19]). In particular, it is possible to efficiently compute gradients through Equation (2.5) by implicitly differentiating through its KKT conditions (i.e., conditions that are necessary and sufficient to describe its optimal solutions).

As Equation (2.5) is a special case of the general differentiable quadratic programming layer described in Amos and Kolter [AK17], we closely follow the gradient derivation therein in our gradient derivation here. Specifically, the KKT conditions for stationarity, primal feasibility, and complementary slackness for this problem are given by

$$\begin{aligned} u^\star - \hat{u} + A^T\nu^\star + G^T\lambda^\star &= 0 \\ Au^\star - b &= 0 \\ \text{diag}(\lambda^\star)(Gu^\star - h) &= 0, \end{aligned} \tag{2.6}$$

where $u^\star$, $\lambda^\star$, and $\nu^\star$ are the optimal primal and dual solutions. By the implicit function theorem, we can then take derivatives through these conditions at the optimum to obtain relevant gradients. Specifically, the total differentials of these KKT conditions are given by

$$\begin{aligned} \mathrm{d}u - \mathrm{d}\hat{u} + \mathrm{d}A^T\nu^\star + A^T\mathrm{d}\nu + \mathrm{d}G^T\lambda^\star + G^T\mathrm{d}\lambda &= 0 \\ \mathrm{d}Au^\star + A\mathrm{d}u - \mathrm{d}b &= 0 \\ \text{diag}(Gu^\star - h)\mathrm{d}\lambda + \text{diag}(\lambda^\star)(\mathrm{d}Gu^\star + G\mathrm{d}z - \mathrm{d}h) &= 0, \end{aligned} \tag{2.7}$$

or in matrix form as

$$\begin{bmatrix} I & G^T & A^T \\ \text{diag}(\lambda^\star)G & \text{diag}(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{d}u \\ \mathrm{d}\lambda \\ \mathrm{d}\nu \end{bmatrix} = - \begin{bmatrix} -\mathrm{d}\hat{u} + \mathrm{d}G^T\lambda^\star + \mathrm{d}A^T\nu^\star \\ \text{diag}(\lambda^\star)\mathrm{d}Gz^\star - \text{diag}(\lambda^\star)\mathrm{d}h \\ \mathrm{d}Az^\star - \mathrm{d}b \end{bmatrix}. \tag{2.8}$$

The terms in these equations look somewhat complex, but fundamentally, the left side gives the Jacobian of the optimality conditions of the convex problem, and the right side gives the derivatives at the achieved solution with respect to the problem parameters $\hat{u}, A, b, G, h$.

As described in Amos and Kolter [AK17], these equations can be used to solve for the Jacobians of any solution variable $u^\star, \lambda^\star, \nu^\star$ with respect to any problem parameter $\varphi \in \{\hat{u}, A, b, G, h\}$, by setting the differential $\mathrm{d}\varphi$ associated with the problem parameter

to its identity value (and all other differentials to zero). As noted both there and in Section 2.2.2, however, we seldom actually want to compute these Jacobians explicitly, due to the potentially large time and space complexity of doing so. Instead, it is often desirable to directly compute $\mathrm{d}\ell/\mathrm{d}\varphi$ by employing the Jacobian-vector trick – that is, by directly computing the left vector-matrix product of these Jacobians with some backward pass vector in order to reduce time and space complexity.

In particular, for some loss function ℓ, if we are given the backward pass vectors $\mathrm{d}\ell/\mathrm{d}u^\star, \mathrm{d}\ell/\mathrm{d}\lambda^\star, \mathrm{d}\ell/\mathrm{d}\nu^\star$ and we want to compute the gradient $\mathrm{d}\ell/\mathrm{d}\varphi$ for some parameter φ, we note by the chain rule that

$$\frac{\mathrm{d}\ell}{\mathrm{d}\varphi} = \frac{\partial \ell}{\partial \varphi} + \frac{\partial \ell}{\partial u^\star}\frac{\mathrm{d}u^\star}{\mathrm{d}\varphi} + \frac{\partial \ell}{\partial \lambda^\star}\frac{\mathrm{d}\lambda^\star}{\mathrm{d}\varphi} + \frac{\partial \ell}{\partial \nu^\star}\frac{\mathrm{d}\nu^\star}{\mathrm{d}\varphi}. \tag{2.9}$$

Per the Jacobian-vector trick, we would then aim to compute the three products in this equation directly, rather than explicitly computing and storing the intermediate Jacobians $\mathrm{d}u^\star/\mathrm{d}\varphi, \mathrm{d}\lambda^\star/\mathrm{d}\varphi, \mathrm{d}\nu^\star/\mathrm{d}\varphi$ at any point during the process. In more detail, using Equation (2.8), we note that

$$\begin{aligned}
&\frac{\partial \ell}{\partial u^\star}\frac{\mathrm{d}u^\star}{\mathrm{d}\varphi} + \frac{\partial \ell}{\partial \lambda^\star}\frac{\mathrm{d}\lambda^\star}{\mathrm{d}\varphi} + \frac{\partial \ell}{\partial \nu^\star}\frac{\mathrm{d}\nu^\star}{\mathrm{d}\varphi} \\
&= \begin{bmatrix} \partial\ell/\partial u^\star \\ \partial\ell/\partial\lambda^\star \\ \partial\ell/\partial\nu^\star \end{bmatrix}^T \begin{bmatrix} \mathrm{d}u^\star/\mathrm{d}\varphi \\ \mathrm{d}\lambda^\star/\mathrm{d}\varphi \\ \mathrm{d}\nu^\star/\mathrm{d}\varphi \end{bmatrix} \\
&= -\begin{bmatrix} \partial\ell/\partial u^\star \\ \partial\ell/\partial\lambda^\star \\ \partial\ell/\partial\nu^\star \end{bmatrix}^T \begin{bmatrix} I & G^T & A^T \\ \mathrm{diag}(\lambda^\star)G & \mathrm{diag}(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\mathrm{d}\hat{u} + \mathrm{d}G^T\lambda^\star + \mathrm{d}A^T\nu^\star \\ \mathrm{diag}(\lambda^\star)\mathrm{d}Gz^\star - \mathrm{diag}(\lambda^\star)\mathrm{d}h \\ \mathrm{d}Az^\star - \mathrm{d}b \end{bmatrix}\Bigg|_{\substack{\mathrm{d}\varphi = I \\ \mathrm{d}\psi = 0, \forall \psi \neq \varphi}}.
\end{aligned} \tag{2.10}$$

Noting that the first two terms are consistent across all problem parameters, as shown in Amos and Kolter [AK17], we can pre-compute their product before computing all relevant gradients. Letting

$$\begin{bmatrix} d_u \\ d_\lambda \\ d_\nu \end{bmatrix}^T := -\begin{bmatrix} \partial\ell/\partial u^\star \\ \partial\ell/\partial\lambda^\star \\ \partial\ell/\partial\nu^\star \end{bmatrix}^T \begin{bmatrix} I & G^T & A^T \\ \mathrm{diag}(\lambda^\star)G & \mathrm{diag}(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1}, \tag{2.11}$$

the relevant gradients (2.9) with respect to all problem parameters are then given by

$$\begin{aligned}
\frac{\mathrm{d}\ell}{\mathrm{d}\hat{u}} &= \frac{\partial \ell}{\partial \hat{u}} - d_u \\
\frac{\mathrm{d}\ell}{\mathrm{d}A} &= \frac{\partial \ell}{\partial A} + d_\nu u^{\star T} + \nu^\star d_u^T & \frac{\mathrm{d}\ell}{\mathrm{d}b} &= \frac{\partial \ell}{\partial b} - d_\nu \\
\frac{\mathrm{d}\ell}{\mathrm{d}G} &= \frac{\partial \ell}{\partial G} + \mathrm{diag}(\lambda^\star)d_\lambda u^{\star T} + \lambda^\star d_u^T & \frac{\mathrm{d}\ell}{\mathrm{d}h} &= \frac{\partial \ell}{\partial h} - \mathrm{diag}(\lambda^\star)d_\lambda.
\end{aligned} \tag{2.12}$$

We note that most of the operations involved in the gradient computations (2.11)–(2.12) are relatively cheap. The notable exception involves the inversion of the KKT Jacobian matrix in Equation (2.11); however, in practice, this matrix inversion was likely already computed when solving the projection (2.5) in the first place (e.g., if the projection was solved using a KKT-based iterative method). As such, it is often possible to get this matrix inversion "for free" in the backward pass, significantly reducing the additional computational complexity associated with the gradient computation. In general, this example shows one way in which implicit gradients can be computed in an inexpensive and scalable manner.

While the above example is for a linearly-constrained projection operation, these kinds of gradients can be computed (or in some cases, approximately computed) for convex projection problems in general. For instance, Donti, Roderick, Fazlyab, and Kolter [Don+21b] compute gradients through a projection onto a second order cone by differentiating through the fixed point equations of a solver, and Agrawal, Amos, Barratt, Boyd, Diamond, and Kolter [Agr+19] provide a method and library for differentiable disciplined convex programs.

2.3 Electric power systems

Electric power systems are the backbone of modern society, and a major focus of many strategies to promote environmental, economic, and social sustainability.[5] For instance, moving to renewable and low-carbon electricity sources will be critical to achieving both climate change and air quality targets [IPC22b] and electricity access is a key pillar of economic development [Car+11]. In this section, we provide a brief overview of electric power systems; we refer readers to Von Meier [VM06], Kirschen and Strbac [KS04], and Wood, Wollenberg, and Sheblé [WWS14] for additional details.

2.3.1 Power system basics

Electric power systems (or, *power systems*) refer to the networks of electrical infrastructure that facilitate the production and transportation of electric power. In particular, electric power systems consist of three main parts:

- Generation: The production of electricity from *primary energy sources* such as fuels (e.g., fossil fuels or nuclear fuels) or renewable resources (e.g., sun, wind, or water).
- Transmission: The transportation of electricity over long distances from where it is produced to it is consumed, using high-voltage transmission lines. (The role of electricity transmission can be analogized to the role of highways in vehicle transportation, where the goal is to travel long distances efficiently.)
- Distribution: The transportation of electricity from transmission infrastructure to end-use consumers, using low-voltage distribution lines. (The role of electricity distribution

[5] The introductory paragraph of this section is adapted from Donti and Kolter [DK21]. Parts of the description of AC optimal power flow are adapted from Donti, Rolnick, and Kolter [DRK21]. Parts of the discussion on climate change mitigation and adaptation are adapted from Rolnick, Donti, Kaack, Kochanski et al. [Rol+22].

can be analogized to the role of local roads in vehicle transportation, where the goal is to safely travel short distances to a final destination.)

Together, transmission and distribution systems form the *electric grid* (or *power grid*).

Given the high fixed costs associated with operating transmission and distribution systems, these systems are generally considered *natural monopolies*, and are managed by either public or highly-regulated entities. In particular, the term *utility* is used to refer to the owner of transmission or distribution infrastructure, and the term *system operator* is used to refer to the entity that oversees the system to ensure it operates smoothly. Depending on the power market structure, the relationships between utilities, system operators, and *power generators* may vary. For instance, many regions in the United States have *regulated electricity markets*, where *vertically-integrated utilities* own and operate all of the generation, transmission, and distribution in a given region; other regions have *deregulated electricity markets*, where system operators are independent from transmission and distribution utilities, and where power generators can be privately owned [Fed15].

While power systems were historically operated in a primarily top-down manner – with power produced by large, centralized generators and then transported one-way to consumers via transmission and distribution infrastructure – the landscape has more recently started shifting towards the paradigm of *smart grids* that accommodate dynamic, two-way energy flows [Fan+11; TA16]. In particular, with the introduction of *distributed energy resources* such as rooftop solar panels (with potentially controllable *power inverters*), of *distributed energy storage* such as distribution-connected batteries and electric vehicles, and of *demand flexibility/demand response* strategies that aim to intelligently shift electricity consumption based on the needs of the power grid, electrical loads on distribution systems have also started to play a more active role in power system operations. This has thereby increased the scale at which power systems must be managed.

2.3.2 Power system operations

At a high level, the operation of power systems can be conceptualized through the lens of a problem called AC optimal power flow (ACOPF). ACOPF is a fundamental problem for the operation of the electrical grid, and is (in principle) used by transmission system operators to determine how much power must be produced by each controllable generator in order to meet demand. (This is also known as determining a *dispatch*.)

Formally, a power network may be considered as a graph on b nodes, representing different locations (*buses*) within the electric grid, and with edges weighted by complex *admittances* $w_{ij} \in \mathbb{C}$ that represent how easily current can flow on the corresponding power lines. Let $W \in \mathbb{C}^{b \times b}$ denote the graph's Laplacian matrix, which in power systems terminology is called the *nodal admittance matrix*,[6] and let $f_c : \mathbb{R}^b \to \mathbb{R}$ be a cost function parameterized by the costs of power generation c. Then, the problem of ACOPF can be defined as follows: Given inputs $p_d \in \mathbb{R}^b$, $q_d \in \mathbb{R}^b$ (representing the *real* and *reactive*

[6] In power systems, the notation Y is more canonically used for the admittance matrix; however, we adopt W here to avoid clashing with canonical machine learning notation, in which Y often stands for a collection of "ground truth" labels.

components of power demand at the various buses), we aim to output $p_g \in \mathbb{R}^b, q_g \in \mathbb{R}^b$ (representing the real and reactive power generation at each bus) and $v \in \mathbb{C}^b$ (representing complex voltage values at each bus)[7] according to the following optimization problem:

$$\underset{p_g \in \mathbb{R}^b,\, q_g \in \mathbb{R}^b,\, v \in \mathbb{C}^b}{\text{minimize}} \quad f_c(p_g) \tag{2.13a}$$

$$\text{subject to} \quad p_g^{\min} \leq p_g \leq p_g^{\max}, \quad q_g^{\min} \leq q_g \leq q_g^{\max}, \quad v^{\min} \leq |v| \leq v^{\max}, \tag{2.13b}$$

$$(p_g - p_d) + (q_g - q_d)i = \text{diag}(v)\overline{W}\overline{v}, \tag{2.13c}$$

where $i \coloneqq \sqrt{-1}$, where the notation $\overline{x}$ is used to denote the complex conjugate of some x, and where $|v|$ denotes the element-wise L_1 norm. The constraints (2.13c) are called the *AC power flow equations*, and ensure that the amount of power flowing into every bus equals the amount of power flowing out. The dual variables $\lambda \in \mathbb{R}^b$ corresponding to the AC power flow equations are (in principle) the wholesale power prices at each bus. While we use simplified notation here, more detailed notation is provided in Appendix B.1.

The ACOPF problem (2.13) is non-convex and NP-hard to solve, due to the nonlinear and non-smooth nature of the power flow equations. As a result, it is common to use less computationally expensive approximations of this problem in various contexts [WWS14; VM06]. In particular, one common approximation is *DC optimal power flow* (DCOPF), which makes various simplifying assumptions about power system quantities that cause the constraints of Equation (2.13) to become linear; while more computationally tractable to obtain, it is worth noting that solutions of DCOPF are not feasible with respect to the original system [Bak21]. Another common approximation omits the power flow constraints altogether, only requiring that the amount of power generation and power consumption be equal on an aggregate level; this variant, called *economic dispatch*, basically reduces to ranking power generation on a *merit order curve* from cheapest to most expensive, and then dispatching power in order, starting from the cheapest generators, until all power demand is met.

While ACOPF (and its cheaper approximations) reflect some basic considerations for the operation of power grids, more complex variants of these problems aim to reflect additional important considerations. In particular, *N-k security-constrained optimal power flow* (N-k SCOPF) aims to schedule power generation in a way that is robust to potentially k simultaneous outages of lines or generators (with N-1 SCOPF being the most common variant). The problem of *stochastic optimal power flow* (stochastic OPF) aims to schedule power generation in a way that is cognizant of demand being stochastic; in particular, while hidden in the notation above, "demand" in the context of optimal power flow refers to actual electricity demand *minus* non-controllable power generation (e.g., from solar and wind), making stochastic OPF a particularly important problem for the integration of time-varying renewable energy. While these problems provide a dispatch for one point in time, the mixed-integer *unit commitment* problem decides which generators should be turned on/off over multiple time steps, as well as how much power they should produce.

While optimal power flow problems provide a rough conceptual model for how power grids operate, in reality, there are many ways in which actual power systems are more complex

[7] Modern electric power systems are generally alternating current (AC) systems, in which all electrical quantities – e.g., powers and voltages – are considered to be complex-valued. In particular, the terms *real power* and *reactive power* refer to the real and imaginary components, respectively, of *complex power*.

[KS04]. For example, controllable power generation is typically only dispatched at fixed time intervals, despite the fact that power demand and time-varying renewable generation change continuously; to account for this (as well as any issues with the original dispatch), power generators and other controllable loads also participate in real-time *ancillary service* schemes such as *frequency control*, *voltage control*, and *reactive power control* to stabilize the grid. As another example, power prices (as in principle captured mathematically within the ACOPF dual variables) are in reality the result of a combination of complex multi-timescale market trading and of longer-term purchasing contracts called *power purchase agreements*. In addition, while we have presented a fundamentally top-down view of power system operations, as previously discussed, distribution systems have also started to play a more active role; this has in some cases led to the rise of *distributed energy markets* in which distributed energy resources and distributed loads can actively trade.

2.3.3 Climate change mitigation and adaptation in power systems

In the context of climate action, electric power systems play a key role in both *climate change mitigation* (reducing or preventing greenhouse gas emissions) and *climate change adaptation* (responding to the effects of a changing climate).

With respect to climate change mitigation, electric power systems are currently responsible for about a quarter of annual greenhouse gas (GHG) emissions [IPC22b]. Demand for low-carbon electricity is also further projected to grow as sectors such as buildings, transportation, and heavy industry seek to "electrify" greenhouse-gas-emitting loads (such as passenger vehicles) [IPC22b]. Therefore, reducing greenhouse gas emissions in the electric power sector is a keystone of societal climate action. In particular, this will entail [IPC22b]:

- Rapidly transitioning to low-carbon[8] electricity sources (such as solar, wind, hydro, nuclear, and geothermal) and away from carbon-emitting sources (i.e., fossil fuels).
- Reducing greenhouse gas emissions associated with existing fossil fuels and electricity infrastructure (e.g., by reducing waste, by preventing methane leaks, or via carbon capture and sequestration).
- Implementing these changes across all countries and contexts, including via potentially tighter integration of regional electricity and energy systems.

With respect to climate change adaptation, power grids both will be affected by climate change impacts and are a key part of strategies to build adaptive capacity in many regions. In particular, climate change adaptation in the power sector will entail the following:

- Making operations more robust and resilient in the face of climate extremes [Nat17]. This may include, e.g., developing robustness to correlated failures of grid components that may occur due to extreme heat or cold [Mur19], or enabling the quick repair of

[8] As a note, while the terms "renewable" and "low-carbon" are often used interchangeably, they are in reality distinct terms with different implied (though related) objectives. In particular, "renewable energy" is often used in service of general sustainability and circular economy goals, whereas "low-carbon energy" is largely used in service of climate change goals. While many forms of energy (such as solar and wind power) fall under both categories, others fall under only one; for instance, biomass-based energy may be renewable but not necessarily low-carbon [Cre16], whereas nuclear energy is low-carbon but not renewable.

grid infrastructure after outage-inducing storms.

- Adapting how power systems are planned to accommodate changing patterns in supply and demand [RF+21], as changes in weather impact both the production of renewable energy and energy consumption for (e.g.) heating and cooling.
- Expanding access to reliable electricity across additional countries and contexts; in particular, energy access and reliability are strong drivers of economic development [Mos+20], which itself is affects broader capacity to adapt to climate change [BCF12].

Achieving these objectives will require efforts to shape both power system operations and planning. On the operations side, power grids will need to be managed at increasing speed and scale – for instance, to accommodate the time-varying nature of renewable energy sources, to integrate additional distributed energy resources, and to explicitly account for different power system failure scenarios. This will require improvements in techniques such as forecasting, estimation, centralized optimization, and distributed control to cope with this speed and scale. On the planning side, power grids will need to be designed, built, and upgraded in ways that foster the tradition to clean energy sources and improve power system reliability, robustness, and resilience. This will require holistic, inter-regional planning of power grids, as well as coordination with the climate science community to build additional understanding of ways in which climate changes will affect power supply, demand, and infrastructure.

In the remainder of this thesis, we focus largely on power system operations, and how machine learning can help improve operations in service of climate change mitigation and adaptation goals. For more detail on the role of machine learning in sustainable energy systems more broadly, we refer readers to Rolnick, Donti, Kaack, Kochanski, et al. [Rol+22] and Donti and Kolter [DK21].

Part I

Estimation Tasks in Power Systems

Chapter 3

Assessing Emissions and Damage Factors in PJM

In recent years, several methods have emerged to estimate the emissions and associated damages (e.g., to health, the environment, and the climate) that are avoided by interventions such as energy efficiency, demand response, and the integration of renewables. However, differing assumptions employed in these analyses could yield contradicting recommendations regarding whether and how an intervention should be implemented. We test the magnitude of the effect of using different key assumptions – average vs. marginal emissions, year of calculation, temporal and regional scope, and inclusion of non-emitting generation – to estimate emissions and damage factors in the PJM Interconnection. We further highlight the importance of these differing assumptions by evaluating three illustrative 2017 power system examples in PJM. We find that for a simple building lighting intervention, using average emissions factors incorporating non-emitting generation *underestimates* avoided damages by 45% compared to marginal factors. For PJM demand response, outdated marginal emissions factors from 2016 *overestimate* avoided damages by 25% compared to 2017 factors. Our assessment of PJM summer load further suggests that fossil-only average emissions factors *overestimate* damages by 63% compared to average factors incorporating non-emitting generation. We recommend that energy modelers carefully select appropriate emissions metrics when performing their analyses. Furthermore, since the U.S. electric grid is rapidly changing, we urge decision-makers to frequently update (and consider forecasting) grid emissions factors.

The work in this chapter has previously been published in:[1]

> Priya L. Donti, J. Zico Kolter, and Inês Lima Azevedo. "How Much Are We Saving After All? Characterizing the Effects of Commonly Varying Assumptions on Emissions and Damage Estimates in PJM." *Environmental Science & Technology* 53.16 (2019), 9905–9914.

[1] Code for all analyses is available online: https://github.com/priyald17/emissions-assumptions.

3.1 Introduction

Power system interventions such as installing efficient building lighting, implementing demand response, and increasing renewables integration can potentially reduce emissions of CO_2, SO_2, NO_x, and $PM_{2.5}$, as well as their associated damages to the climate, environment, and human health. As decision-makers design such interventions, it is important to understand the actual benefits achieved by their implementation. Extensive prior work has evaluated interventions such as electric vehicle charging [GZKM14; Tam+15; Wei+15; RJK16; Yuk+16], battery storage management [CN13; HA15; FA17; HA17], data center load shifting [Hor16], building energy efficiency [GAJ14], and changes in fuel mix [KML13; SE+13; KA16; Lue+16; HA17; HL17] using emissions and damage factors. However, such analyses have differed in terms of the key assumptions they have made when calculating emissions and damage factors.

One key distinction involves average vs. marginal emissions factors (AEFs vs. MEFs). AEFs measure the average emissions intensity of all electricity generation at a given time and are widely used due to their simplicity of calculation. Alternatively, MEFs measure the emissions intensity of marginal generators, which are the last to be dispatched to meet demand and thus the first to respond to power system interventions.

Another important assumption involves the potential inclusion of non-emitting (i.e., renewable and nuclear) electricity sources. While most studies have examined marginal emissions factors using only fossil fuel data, recent analyses suggest that renewables may influence marginal emissions factors in regions such as the U.S. Midcontinent Independent System Operator (MISO) region [Li+17].

Other assumptions in calculating emissions factors involve regional and temporal scope. In the U.S., studies have employed emissions factors nationally [Sch+17a] and for grid interconnections [GZKM14], North American Electric Reliability Corporation (NERC) regions [SEAM12], independent system operator (ISO) and regional transmission organization (RTO) regions [Li+17; PJM17a; Thi+17], and even custom-defined regions [Uni17c]. Temporally, emissions factors have been calculated at the annual [Uni17b], monthly [PJM17a], and time of day [GZKM14] levels, and using data from different years. Additional assumptions may include using non-temporal partitions such as load bins to calculate emissions factors [SEAM12] and using generation vs. demand data in emissions factor calculations [GZKM14].

We seek to answer the question of how influential a number of these assumptions are on emissions and damage assessments in the PJM RTO. PJM operates the largest competitive wholesale market in the U.S., encompassing utilities from all or part of 13 states and servicing about 770 TWh of annual demand [PJM18]. In 2017, PJM accounted for roughly 20% of emissions of criteria pollutants and carbon dioxide from U.S. electricity generators (based on data from the U.S. Environmental Protection Agency's Continuous Emissions Monitoring System), making it an important candidate for power system interventions. As a deregulated entity, PJM can shape market participation of power system interventions via incentives or reduction of barriers to entry, warranting careful analysis as to these interventions' potential costs and emissions effects.

Our study serves as an initial input to such analysis by seeking to understand the implications of choosing different types of emissions and damage factors when evaluating

Table 3.1: Emissions and generation data employed in our analysis.

Data	Source	Description	Processing done by authors
Generator labels			
Generator ISO/RTO membership	eGRID (EPA)	Generator's ISO/RTO of membership (if any) (2011)	Selected the generators associated with PJM
Emissions			
Hourly CO_2, SO_2, NO_x emissions	CEMS (EPA)	Hourly emissions for fossil-fueled generators larger than 25 MW (2006-17)	Aggregated to PJM
Annual primary $PM_{2.5}$ emissions	NEI (EPA)	Annual $PM_{2.5}$ emissions for subset of CEMS generators (2008, 2011, 2014)	Converted to generator-specific and average by-fuel-type emissions rates, and then multiplied by hourly generation
Generation			
Hourly fossil generation	CEMS (EPA)	Gross hourly generation for fossil-fueled generators larger than 25 MW (2006-17)	Aggregated to PJM
Hourly non-emitting generation	PJM (Data Miner 2)	System-wide hourly generation by fuel type for PJM (mid-2015 onward)	Aggregated generation from nuclear, wind, hydro, solar, and "other renewables"
Hourly marginal generator proportions by fuel type	PJM (via Monitoring Analytics)	Hourly fraction of marginal generators by fuel type (2006-17)	Converted to dummy variables for hourly presence of nuclear, wind, hydro, solar, or "other renewables"

interventions. Specifically, we compare emissions and damage factors calculated using different key assumptions – average vs. marginal, with or without non-emitting generation, and at different regional and temporal scopes – and evaluate the effects of using these factors for evaluations of a building lighting efficiency intervention, economic demand response, and summer load in PJM. While previous work has addressed the effects of some of these assumptions (for CO_2) [RJK16; Rya+18], to our knowledge, our work is the first to systematically quantify the effects of varying these key assumptions in historical emissions and damage factor estimates (across multiple pollutants).

3.2 Data and methods

We estimate generation-based average and marginal emissions and damage factors for CO_2, SO_2, NO_x, and $PM_{2.5}$ in PJM and the Reliability First Corporation (RFC), the NERC region most similar in spatial scope to PJM. We estimate these factors from hourly data for multiple temporal breakdowns: by year (annual), by month (monthly), by hour of day for a given month (monthly time of day, i.e., monthly TOD), and for each hour of the year (hourly; for average factors only). We additionally consider various generation fuel mixes: for 2006-17, we calculate emissions factors using fossil generation only, and for PJM in 2016-17, we additionally calculate factors incorporating non-emitting generation.

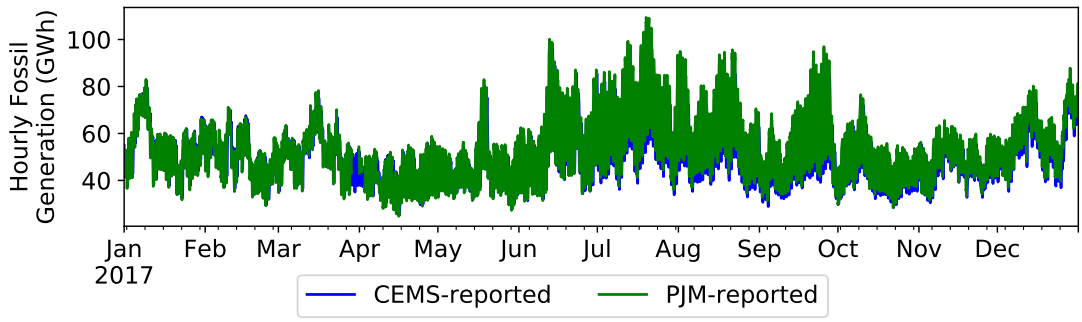


Figure 3.1: Hourly fossil generation in PJM in 2017, as reported by CEMS and by PJM. CEMS includes generation for generators larger than 25 MW, whereas PJM reports the entirety of fossil generation within its footprint. The CEMS- and PJM-reported values are similar, with PJM-reported values on average less than 6% higher than CEMS values (and in the median about 4% higher). In other words, CEMS-excluded fossil generators do not contribute significantly to fossil generation in PJM, and we hypothesize that the exclusion of such generators (by virtue of our use of the CEMS data) is not a significant limitation to our analysis; that said, it may be worth including CEMS-excluded fossil generators in future analyses, pending data availability.

3.2.1 Data

We employ hourly emissions and electricity generation data from the U.S. Environmental Protection Agency (EPA) and PJM (see Table 3.1).

We obtain hourly-level emissions and fossil generation data for 2006-17 from the Continuous Emissions Monitoring System (CEMS) [Uni09] and the National Emissions Inventory (NEI) [Uni17d] through the EPA. CEMS reports hourly CO_2, SO_2, and NO_x emissions as well as total hourly generation for all U.S. fossil-fueled generators larger than 25 MW. (We estimate that smaller fossil generators contribute less than 6% of PJM's total fossil generation, and thus that the exclusion of such generators is not a significant limitation to our analysis; see Figure 3.1 for more details.)

NEI reports primary annual $PM_{2.5}$ emissions for a subset of these fossil generators, but only for the years 2008, 2011, and 2014. To estimate primary hourly $PM_{2.5}$ emissions for different years, we first find generator-specific emissions rates by dividing total annual emissions from NEI by total annual generation from CEMS at the generator level in each NEI year. We further use these generator-specific rates to calculate average $PM_{2.5}$ emissions rates by generator fuel type. To then estimate $PM_{2.5}$ emissions for each individual generator in all hours of different years, we multiply CEMS-reported hourly generation either by (a) the generator-specific emissions rate (for generators in NEI) or (b) the applicable average by-fuel-type emissions rate (for generators not in NEI), using the rates calculated from the 2008, 2011, and 2014 NEIs for the years 2006-2010, 2011-2013, and 2014-2017, respectively. (More discussion about our use of NEI data is provided in Appendix A.1.) We aggregate PJM hourly emissions (CO_2, SO_2, NO_x, or $PM_{2.5}$) and fossil generation by summing across all generators eGRID 2011 reports as being associated with PJM.

We further analyze the effects of including recent nuclear and renewable generation data from PJM in our estimates. PJM reports system-wide hourly generation by fuel type starting mid-2015, from which we extract nuclear, wind, hydro, solar, and "other renewable"

sources as being non-emitting. PJM also reports the fraction of marginal generators in each hour that belong to each fuel type (where this hourly data is aggregated from the 5-minute level). As this data does not report the *extent* of marginal generation from each fuel type–only the *number* of marginal generators–we convert it to dummy variables $I_s(t)$ indicating the hourly presence of each non-emitting fuel type s in the marginal generator data at time t: that is, we let $I_s(t) = 1$ if fuel s is ever marginal during hour t and set it to 0 otherwise. For 2016-17 in PJM, we find that wind, nuclear, and solar are marginal in 27%, 15%, and 0.9% of hours, respectively, and other non-emitting sources are never marginal.

3.2.2 Calculating AEFs

For a given grouping of hours, we calculate the AEF as the ratio of total emissions to total generation for that grouping. Explicitly, for a pollutant p, generation mix C, region R, and set of hours T corresponding to a given temporal breakdown, the AEF is given by

$$\text{AEF}^{p,C}_{R,T} = \sum_{t \in T} E^{p}_{R,t} \; / \; \sum_{t \in T} G^{C}_{R,t}, \tag{3.1}$$

where $E^{p}_{R,t}$ is the total emissions of pollutant p (kg) and $G^{C}_{R,t}$ is the total generation for fuel mix C (MWh), both in region R in hour t.

3.2.3 Calculating MEFs

We estimate MEFs via regression of marginal emissions against marginal generation, following a similar approach as in previous work [Hor16; HA17; SE+13; SEAM12]. This approach entails estimating the emissions effects of potential *instantaneous* changes in power generation (i.e., counterfactuals we do not observe) using the emissions effects of changes in generation *between* adjacent hours (which we do empirically observe). The fundamental assumption, here, is that the merit order curve associated with economic dispatch stays roughly the same between adjacent hours; this means that the generators that *would have* been dispatched to serve an instantaneous change in load in a given hour are roughly the same as those that were *actually* dispatched to serve changes in load between adjacent hours. We note that this is indeed an assumption, and does not account for power system congestion or other factors that may cause the merit order to change (see, e.g., the discussion in Section 2.3.2). However, we posit that this approach still provides a reasonable first-order approximation for marginal emissions factor values, and we employ this approach in keeping with prior work.

Specifically, for each pollutant p and hour t, we estimate the hourly marginal emissions $\Delta E^{p}_{R,t}$ (kg) as the hourly change in total regional emissions:

$$\Delta E^{p}_{R,t} = E^{p}_{R,t} - E^{p}_{R,t-1}. \tag{3.2}$$

Marginal generation $\Delta G^{C}_{R,t}$ (MWh) is estimated as the hourly change in total regional fossil generation plus (potentially) in marginal non-emitting generation. That is, we estimate

marginal generation for fossil-only factors (C = fossil) and factors incorporating non-emitting generation (C = f+ne), respectively, as

$$\Delta G^{\text{fossil}}_{R,t} = G^{\text{fossil}}_{R,t} - G^{\text{fossil}}_{R,t-1}, \tag{3.3}$$

$$\Delta G^{\text{f+ne}}_{R,t} = \Delta G^{\text{fossil}}_{R,t} + \sum_{s \in \left\{ \begin{smallmatrix} \text{nuclear, wind, hydro,} \\ \text{solar, “other renewables”} \end{smallmatrix} \right\}} I_s(t)(G^s_{R,t} - G^s_{R,t-1}). \tag{3.4}$$

We note that this calculation implicitly makes two assumptions regarding the role of fossil fuels in marginal generation: (a) that fossil fuels are marginal in every hour, and (b) that *all* changes in fossil generation are marginal changes. We believe these assumptions to be reasonable for the present PJM system. On the first point, in 2016 and 2017 (the latest years in which we calculate fossil-plus-non-emitting factors), PJM reports fossil fuels as being marginal in all but five hours; in those five remaining hours, the marginal fuels are ambiguous but likely include fossil fuels, as the marginal generation is reported as being partially wind and partially “min gen/dispatch reset” (i.e., events in which power plants–likely fossil-fuel power plants–need to account for over-generation in the system). On the second point, we expect that non-marginal changes in fossil fuel generation are relatively insignificant compared to marginal changes; in particular, maintenance events are relatively infrequent and changes attributed to ancillary services are relatively small. In a future system where fossil fuels are *not* always marginal (i.e. where non-emitting sources may often be marginal), these assumptions may not hold; in this future case, we would suggest that those estimating marginal emissions factors modify Equations (3.2)–(3.4) to exclude those changes in fossil-fuel generation and emissions that are not explicitly associated with marginal generators.

We further note that Equation (3.4) computes the marginal generation for each non-emitting source s by differencing its hourly generation *before* selecting whether to use this difference in hour t (i.e., computes $I_s(t)(G^s_{R,t} - G^s_{R,t-1})$) instead of differencing *after* selection (i.e., computing $I_s(t)G^s_{R,t} - I_s(t-1)G^s_{R,t-1}$ as in Li, Smith, Yang, and Wilson [Li+17]). We employ this method because PJM has many contiguous hours in which wind (the dominant non-emitting marginal fuel) switches between marginal and not marginal, which means differencing after selection would lead us to incorrectly include *all* wind generation from the marginal hour in our marginal generation estimate. The latter method would thus overestimate the magnitude of marginal generation from wind. However, we suspect that Equation (3.4) likely still slightly overestimates the magnitude of non-emitting marginal generation, as some hour-to-hour changes in non-emitting generation that we count as marginal are likely due to natural resource variation rather than active response to changing demand.

Using the estimates (3.2)–(3.4), given a set of hours T, we then calculate MEFs via regression for hours $t \in T$, i.e.,

$$\Delta E^p_{R,t\in T} = \text{MEF}^{p,C}_{R,T}\, \Delta G^C_{R,t\in T} + \alpha^{p,C}_{R,T} + \epsilon^{p,C}_{R,T}, \tag{3.5}$$

where the slope $\text{MEF}^{p,C}_{R,T}$ of the regression corresponds to the MEF estimate (kg/MWh) for pollutant p and generation mix C in region R for the set of hours T, and where $\alpha^{p,C}_{R,T}$

and $\epsilon_{R,T}^{p,C}$ are the regression intercept and noise, respectively. We note that this calculation likely overestimates the magnitude of difference between fossil-only and fossil+non-emitting factors, given the prior discussion that the estimate of $\Delta G_{R,t}^{\text{f+ne}}$ calculated via Equation (3.4) likely overaccounts for non-emitting generation.

3.2.4 Calculating average and marginal damage factors

We additionally calculate average and marginal damage factors (ADFs and MDFs), i.e., the monetized health, environmental, and climate change damages associated with average and marginal emissions (2010 dollars). For CO_2, we simply multiply the average and marginal emissions factors $\text{AEF}_{R,T}^{\text{CO}_2,C}$ and $\text{MEF}_{R,T}^{\text{CO}_2,C}$ by a social cost of carbon of \$40/ton CO_2 to get their respective damage factors $\text{ADF}_{R,T}^{\text{CO}_2,C}$ and $\text{MDF}_{R,T}^{\text{CO}_2,C}$ (\$/MWh); at the time our analysis was conducted, this social cost of carbon was approximately the value recommended for U.S. regulatory impact analyses under a 3% average discount rate [Uni16a].

For SO_2, NO_x, and $PM_{2.5}$, damages vary by region and population density. Thus, following Siler-Evans, Azevedo, Morgan, and Apt [SE+13], we convert from emissions to damages *at the individual generator level* by multiplying hourly emissions by a location-specific damage intensity. We aggregate damages to the regional level only after this generator-level conversion. Generator-level damage intensities are obtained from two distinct models: an integrated assessment model of U.S. air pollution called AP2 [Mul14] and a reduced-form air quality model called EASIUR [HAG16]. More information about our use of these models is included in Appendix A.2.

To calculate pollutant-specific ADFs and MDFs from our regional aggregations of damages and generation, we employ similar methods as for AEFs and MEFs. That is, we simply substitute damages $D_{R,t}^{p}$ for emissions $E_{R,t}^{p}$ in all previous equations to yield the average and marginal damage factors $\text{ADF}_{R,T}^{p,C}$ and $\text{MDF}_{R,T}^{p,C}$ (\$/MWh).

To get a total marginal damage factor (in \$/MWh) across all pollutants, we sum the per-pollutant marginal damage factors as $\text{MDF}_{R,T}^{C} = \sum_{p \in \{\text{CO}_2,\ \text{SO}_2,\ \text{NO}_x,\ \text{PM}_{2.5}\}} \text{MDF}_{R,T}^{p,C}$. The aggregated average damage factor $\text{ADF}_{R,T}^{C}$ is calculated similarly.

3.2.5 Selecting factors for emissions/damage assessments

We employ the calculated emissions and damage factors to estimate the effects of interventions and loads in PJM. To analyze how commonly-varying assumptions affect these estimates, we first identify a baseline factor that a decision-maker may reasonably use in their assessments. In all cases, the choice of baseline factor depends on how the intervention or load in question affects the grid. We describe the rules-of-thumb used in this work when ascribing this effect, while recognizing that application-specific considerations (including data availability) may warrant different choices of baseline factor in practice. For additional discussion on these topics, please see Ryan, Johnson, and Keoleian [RJK16] and Ryan, Johnson, Keoleian, and Lewis [Rya+18].

Average vs. marginal factors. Marginal factors measure the emissions intensity of

marginal generators, so it is appropriate to use them for interventions or loads that cause changes in marginal generation (e.g., small changes in demand or the introduction of renewables [RJK16]). Average factors measure the emissions intensity of all generation and are thus appropriate for attributing interventions or loads whose effects are approximately distributed throughout the generation mix (e.g., large existing loads). We note that in the presence of cap-and-trade, marginal factors may be zero if emissions caps are binding [TCC11]; however, since most of PJM is not part of CO_2 cap-and-trade schemes such as the Regional Greenhouse Gas Initiative (RGGI) and since NO_x caps under the Clean Air Interstate Rule/Cross-State Air Pollution Rule (CAIR/CSAPR) are non-binding, marginal factors are nonzero for our case study.

Inclusion of non-emitting sources. In most cases, choosing to include all grid electricity generation sources (emitting and non-emitting) in factor estimates most accurately captures the grid effects of a particular intervention or load. If there is a strong case for why a particular intervention or load will not affect non-emitting sources, then it may be reasonable to use fossil-only factors.

Temporal granularity. In general, factors should be chosen granularly to capture the actual time frame on which the given intervention or load occurs, while also acknowledging that factors calculated at more granular level will exhibit more variance (i.e., more uncertainty) than those calculated at aggregate levels.

Regional scope. Regional boundaries should be chosen to match the locations on the grid that a given intervention or load affects. These boundaries should be sufficiently coarse to capture regional import/export effects [RJK16; Uni17c] (as, e.g., loads in one region may cause generators in another region to ramp), but sufficiently granular so as to not capture grid locations largely unaffected by a particular intervention or load. Such exact attributional analysis may be difficult, however, leading to analyses within administrative boundaries (e.g., ISOs/RTOs) in practice.

3.3 Results and discussion

We analyze average and marginal emissions and damage factors for various temporal scopes and generation fuel mixes in PJM and RFC. We then examine these different factors' implications for estimating the emissions and damage effects of PJM power system interventions and loads – specifically, building lighting efficiency, economic demand response, and summer load. As all results were similar under the AP2 and EASIUR damage models, results reported in the text are under AP2, unless otherwise stated; results under EASIUR are reported in Appendix A.3.

3.3.1 Annual and monthly emissions factors over time

We analyze average and marginal emissions and damage factors in PJM and RFC from 2006 to 2017, using annual factors (T contains all hours in a given year) and monthly factors (T contains all hours in a given month of a given year). Annual and monthly total damage factors under AP2 (including CO_2, SO_2, NO_x, and $PM_{2.5}$) are shown in Figure 3.2. Numerical comparisons of annual and monthly emissions and damage factors under both AP2 and EASIUR are shown in Table 3.2.

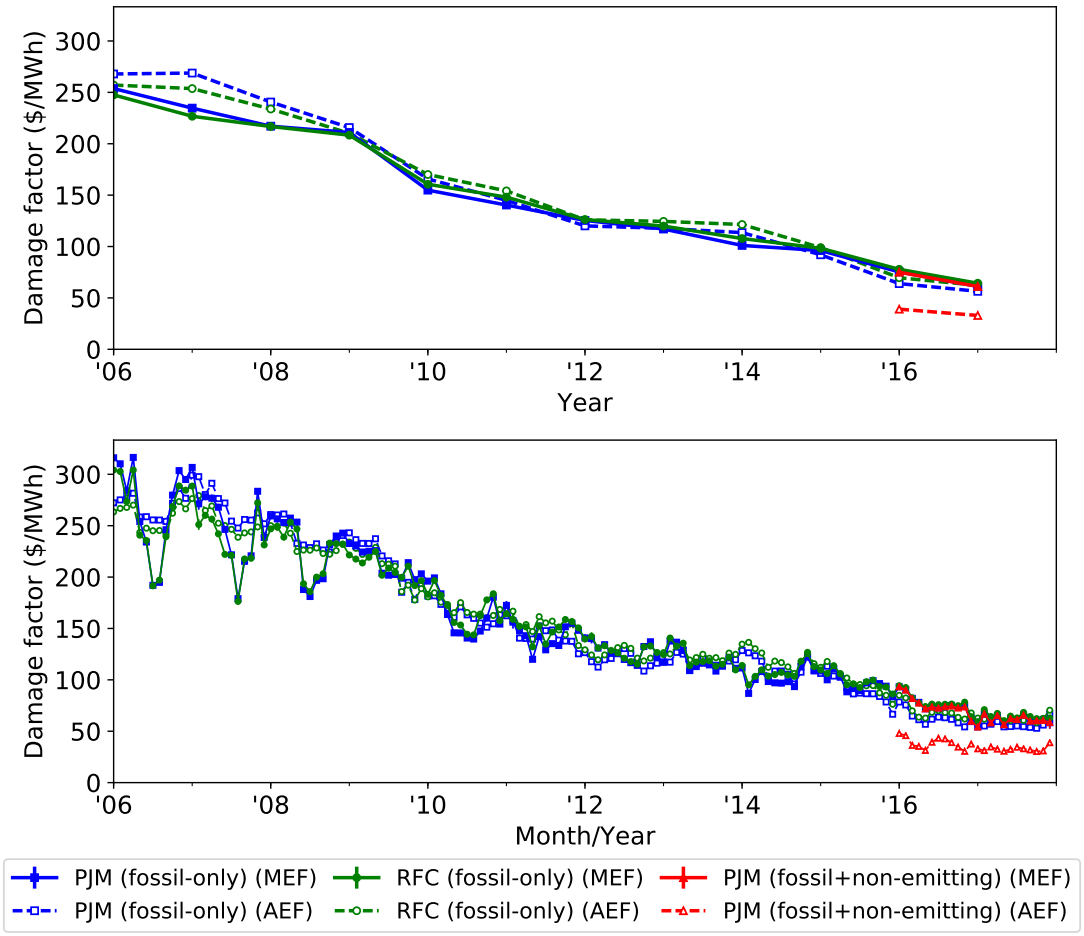


Figure 3.2: Annual (top) and monthly (bottom) average and marginal factors over time for total damages under the AP2 damage model (i.e., health damages from SO_2, NO_x, and $PM_{2.5}$, and climate change damages from CO_2 in 2010 dollars) in PJM and RFC. Error bars for marginal factors (narrow) represent regression standard errors and do not reflect the uncertainty in the underlying data. PJM (fossil-only) = emissions factor estimates using only fossil fuel generation in PJM; RFC (fossil-only) = emissions factor estimates using only fossil fuel generation in RFC; PJM (fossil+non-emitting) = emissions factor estimates using fossil fuel and non-emitting generation in PJM; (MEF) = marginal emissions factor estimate; (AEF) = average emissions factor estimate. 2016 was the first full year in which non-emitting generation data was available, so we only show 2016 and 2017 results for "PJM (fossil+non-emitting)" scenarios.

Our analysis of annual factors shows that fossil-only total damage factors in PJM decreased significantly over time. For example, by 2017, total damage factors had decreased

Table 3.2: Comparison of annual and monthly emissions and damage factors (in %). Total damage factors include CO_2 climate change damages and SO_2, NO_x, and $PM_{2.5}$ health and environmental damages. Per-pollutant ranges capture comparisons for the pollutant's emissions and damage factors. The labels "(M)" and "(A)" indicate marginal and average factor comparisons, respectively.

Comparison	***Period***	***Total damage factors***		***Per-pollutant emissions and damage factors***			
		AP2	**EASIUR**	**CO_2**	**SO_2**	**NO_x**	**$PM_{2.5}$**
% decrease over time (annual factors, PJM *fossil-only*)	2006-2017	76 (M) 79 (A)	73 (M) 76 (A)	12 (M) 17 (A)	86 to 87 (M) 90 to 91 (A)	57 to 73 (M) 65 to 75 (A)	65 to 72 (M) 65 to 71 (A)
	2016-2017	19 (M) 12 (A)	15 (M) 8 (A)	5 (M) 1 (A)	24 to 30 (M) 16 to 23 (A)	22 to 31 (M) 21 to 26 (A)	7 to 17 (M) 2 to 12 (A)
% decrease over time (annual factors, PJM *fossil+non-emitting*)	2016-2017	18 (M) 16 (A)	15 (M) 12 (A)	5 (M) 6 (A)	24 to 30 (M) 19 to 26 (A)	22 to 31 (M) 24 to 29 (A)	7 to 16 (M) 7 to 16 (A)
Average vs. marginal mean % difference (annual factors, PJM *fossil-only*)	2006-2017	2	6	12	-1 to 4	-5 to 6	-6 to 9
	2017	-8	-3	7	-24 to -21	-21 to 2	-5 to 10
Average vs. marginal mean % difference (annual factors, PJM *fossil+non-emitting*)	2016	-48	-45	-37	-58 to -56	-51 to -41	-45 to -36
	2017	-46	-43	-37	-56 to -54	-54 to -40	-45 to -36
RFC vs. PJM % difference (annual factors, fossil-only)	2006-2017	2 (M) 3 (A)	5 (M) 7 (A)	2 (M) 4 (A)	2 to 7 (M) 3 to 11 (A)	1 to 9 (M) 4 to 15 (A)	1 to 8 (M) -2 to 7 (A)
	2017	5 (M) 10 (A)	8 (M) 15 (A)	2 (M) 7 (A)	2 to 7 (M) 9 to 17 (A)	4 to 20 (M) 11 to 30 (A)	19 to 31 (M) 19 to 34 (A)
Monthly vs. annual mean absolute % difference (PJM fossil-only factors; ranges are across years)	2006-2017	4 to 16 (M) 3 to 12 (A)	3 to 15 (M) 2 to 11 (A)	1 to 5 (M) 0 to 2 (A)	7 to 18 (M) 2 to 23 (A)	4 to 43 (M) 3 to 39 (A)	2 to 13 (M) 1 to 8 (A)
	2017	5 (M) 4 (A)	4 (M) 3 (A)	2 (M) 2 (A)	9 to 10 (M) 9 to 11 (A)	4 to 11 (M) 10 to 15 (A)	5 to 8 (M) 2 to 5 (A)
Monthly vs. annual mean absolute % difference (PJM fossil+non-emitting factors; ranges are across years)	2016	8 (M) 11 (A)	6 (M) 9 (A)	2 (M) 6 (A)	12 to 15 (M) 13 to 19 (A)	8 to 16 (M) 11 to 15 (A)	3 to 6 (M) 6 to 10 (A)
	2017	5 (M) 5 (A)	4 (M) 4 (A)	2 (M) 3 (A)	9 to 10 (M) 9 to 11 (A)	4 to 11 (M) 8 to 13 (A)	6 to 8 (M) 3 to 6 (A)
Fossil+non-emitting vs. fossil-only mean % difference (annual factors, PJM)	2016	0 (M) -39 (A)	0 (M) -39 (A)	0 (M) -39 (A)	-1 to 0 (M) -39 to -38 (A)	-1 to 0 (M) -39 to -38 (A)	-1 to 0 (M) -39 to -38 (A)
	2017	0 (M) -42 (A)	0 (M) -42 (A)	0 (M) -42 (A)	-1 to 0 (M) -42 to -41 (A)	-1 to 0 (M) -42 to -41 (A)	-1 to 0 (M) -42 to -41 (A)
Pollutant share in total damage factor (annual factors, PJM fossil-only)	2006	-	-	12 (AP2) 15 (EAS)	75 (AP2) 68 (EAS)	3 (AP2) 5 (EAS)	10 (AP2) 12 (EAS)
	2017	-	-	48 (AP2) 53 (EAS)	33 (AP2) 26 (EAS)	4 (AP2) 6 (EAS)	14 (AP2) 15 (EAS)

by 76-79% (across marginal and average factors) when compared to 2006 levels. Criteria pollutants contributed significantly to this decrease, whereas CO_2 contributed only modestly. (As a result, criteria pollutants comprised 52% of PJM fossil-only total damage factors in 2017, compared to 88% of total damage factors in 2006.) This discrepancy between CO_2 and criteria pollutants is likely due to the replacement of coal with natural gas plants, which are about half as CO_2-intensive as coal plants[2] but much less SO_2- and NO_x-intensive [DG+14]. Specifically, coal's share in PJM dropped from 60 to 50 GW of capacity between 2007 and 2016 (from 50 to 35% of generation between 2006 and 2014), with natural gas' share increasing from 50 to 65 GW of capacity (from 20 to 30% of generation) during the same time periods [PJM17e]. Additionally, SO_2 and NO_x emissions trading programs such as CAIR and CSAPR prompted installation of emissions controls for criteria pollutants [Uni12b; DG+14; Cen17].

These steep changes over time highlight the importance of updating emissions factor estimates on a regular basis. Using outdated emissions factors may imply significantly inflated conclusions as to the effectiveness of modern power system interventions, especially for criteria pollutants. Similarly, this finding underscores the need for accurate estimation of *future* emissions factors (via data- or optimization-driven approaches) when designing future interventions.

Comparing marginal and average factors, we find that while total fossil-only annual average damage factors overall slightly exceeded their marginal counterparts during 2006-17, average factors decreased at a faster rate than marginal factors. In fact, average total damage factors dropped *below* their marginal counterparts during 2015-2017. We suspect this is because *marginal* natural gas units were less clean than *average* natural gas units, even though natural gas replaced coal at similar rates in both baseload[3] and marginal generation [PJM17a; PJM17e].

Adding non-emitting generation to our factors in 2016 and 2017 (the two full years for which we have the relevant data) exacerbates differences between marginal and average factors, with *average* annual fossil+non-emitting total damage factors close to half their *marginal* counterparts. While this result is not surprising (as fossil+non-emitting average factors require dividing emissions by *total* generation rather than just fossil generation, and fossil+non-emitting marginal factors capture the fact that nuclear and renewable sources are often non-marginal), it has tremendous implications for policy implementation. For instance, evaluating a marginal intervention using a fossil+non-emitting average emissions factor (the type of AEF reported by PJM [PJM17c]) could underestimate the effects of that intervention by 36-58% (across all pollutants and damage models in both years), potentially flipping a policy decision about whether that intervention should be pursued. This underscores the importance of rigor and transparency when publishing emissions/damage factors and applying them to system analyses, whether using average or marginal factors.

Using PJM vs. RFC boundaries also leads to differences in total damage factors. While RFC factors followed qualitatively similar temporal trends to PJM factors during this time

[2] Our statement about the reduced GHG emissions of natural gas power vs. coal power does not account for the potential effect of methane leaks in natural gas infrastructure (which are not reflected in our emissions data).

[3] *Baseload* refers to power generation that consistently stays "on" to meet minimum power demand.

period, RFC's total damage factors were generally higher than PJM's (by 2-7% across all years and by 5-15% in 2017, across AP2 and EASIUR marginal and average factors). Across all years, *damage* factors for NO_x diverged the most of any individual damage factor despite relatively similar *emissions* factors. In 2017, $PM_{2.5}$ factors were also much higher in RFC than PJM. This is possibly due to the presence of coal plants near population centers in Michigan that are captured in RFC but not in PJM, as suggested by EIA and census data [Bur10; Uni17a].

Finally, we find that annual factors mask some of the intra-annual variability in emissions factors: monthly total damage factors exhibit mean absolute differences of 3-16% from their corresponding annual total damage factors (for PJM fossil-only factors across marginal and average factors in 2006-17), with most of the variability due to SO_2 and NO_x. In other words, using an annual factor to evaluate or design interventions could potentially miss important differences between emissions and damage factors in different months and hours, especially for criteria pollutants.

Overall, we find that the distinction between marginal and average factors drives the largest changes in emissions and damage factors, specifically when incorporating non-emitting generation (as is common for average factors). Using outdated factors can also significantly change estimated intervention results, and thus power system interventions should be evaluated using emissions factors for the year in which they took or will take place (which is sometimes not the case in existing literature and policy evaluations). Both the use of PJM vs. RFC regional boundaries and intra-annual variability are also somewhat influential in driving emissions and damage factor values, especially for criteria pollutants.

On the other hand, we find that adding non-emitting generation has virtually no (less than 1%) effect on marginal factor estimates. This is because while non-emitting sources (particularly wind and nuclear) are marginal in a nontrivial number of hours, the amount of marginal *generation* (i.e. the magnitude of hour-to-hour generation changes in marginal hours) attributed to these sources is small. These results imply that fossil-only marginal factors are adequate to evaluate marginal interventions in PJM at this time, though we emphasize that this latter finding is specific to the PJM region during the time period evaluated (2016-17) and thus may not be appropriate in modeling system emissions in other regions or time periods.

3.3.2 Intra-annual variability in emissions and damage factors

We now examine the intra-annual temporal variability in emissions factors in 2017, the latest year for which we have a full year of both fossil and non-emitting generation data. Specifically, we examine average and marginal emissions and damage factors partitioned by month and time of day (i.e., monthly TOD). Figure 3.3 illustrates AP2 total damage factors in all months of 2017 for three pairings of emissions and generation: PJM with fossil generation only, PJM with non-emitting generation (fossil+non-emitting), and RFC with fossil generation only.

We find ample intraday variability in monthly TOD marginal emissions factors. Monthly TOD marginal emissions factors for CO_2, SO_2, NO_x, and $PM_{2.5}$ vary on average by 3-8%, 25-186%, 21-36%, and 6-27%, respectively, from their monthly means, with maximum variations of 26%, 761%, 114%, and 67%, respectively (where these ranges are across all months in

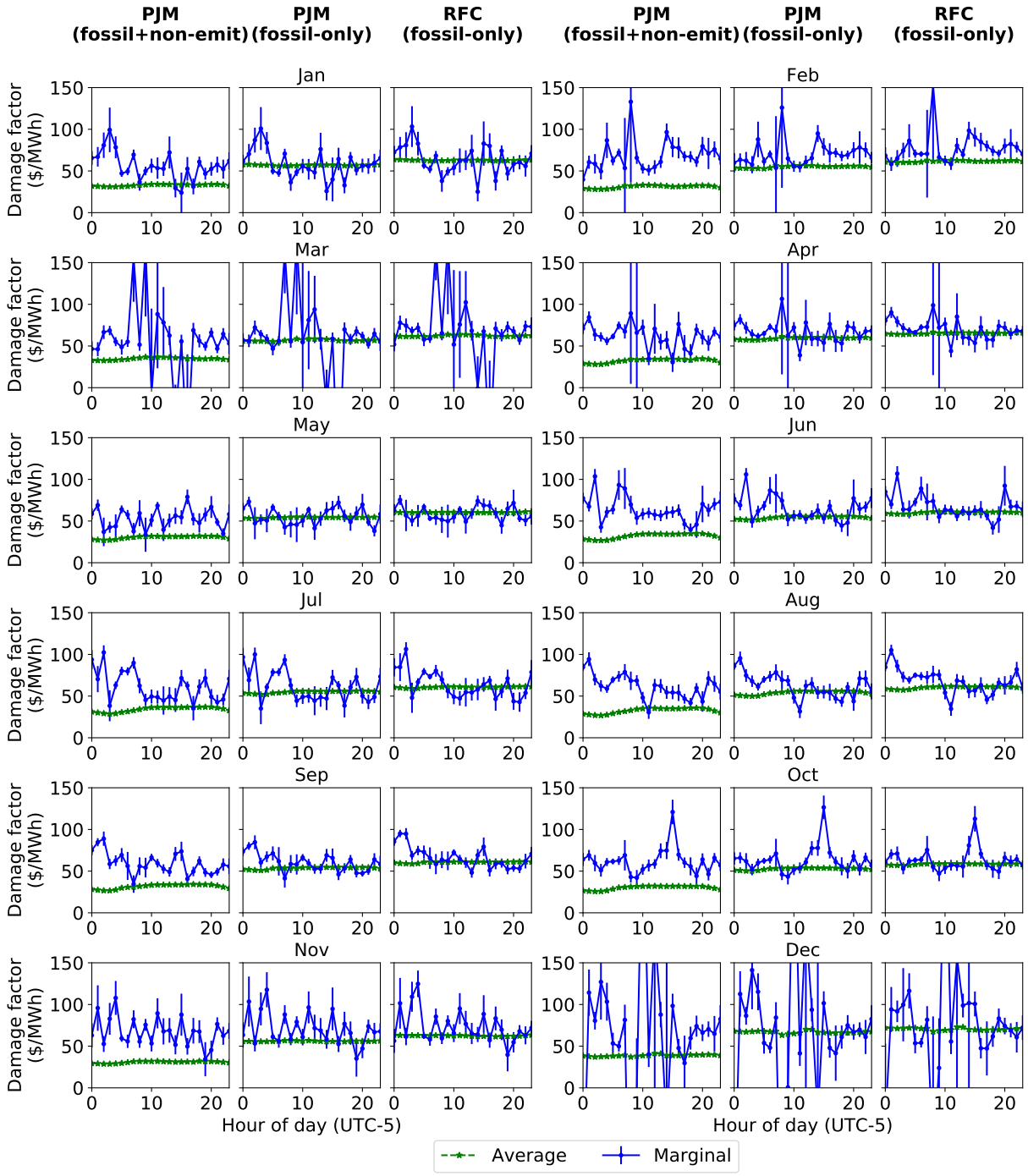


Figure 3.3: Monthly time of day total damage factors under the AP2 damage model (incorporating health and climate change damages for CO_2, SO_2, NO_x, and $PM_{2.5}$, in 2010 dollars), in all months of 2017. Error bars for marginal factors represent regression standard errors and do not reflect the uncertainty in the underlying data. PJM (fossil+non-emitting), PJM (fossil-only), and RFC (fossil-only) are as defined in Figure 3.2.

2017 for PJM fossil-only factors). The corresponding total damage factors from AP2 vary by 13-100% from their monthly means, with a maximum variation of 458%. The shapes of the intraday marginal emissions curves also differ between months (see Figure 3.3). *Average* monthly TOD factors vary much less by hour of day, likely because average factors are primarily determined by baseload (which does not change significantly throughout the day).

We find that differences between PJM and RFC marginal factors are exacerbated at this level of granularity, with RFC fossil-only marginal total damage factors on average 20% higher than their PJM counterparts in 2017 under AP2. (This number is 94% under EASIUR.) This result indicates that PJM and RFC factors are not necessarily interchangeable at this level of temporal granularity.

At monthly TOD granularity, we also do find a modest difference between fossil-only and fossil+non-emitting marginal monthly TOD factors in PJM, with fossil+non-emitting damage factors on average 6% lower than their fossil-only counterparts under AP2, and 13% lower under EASIUR.

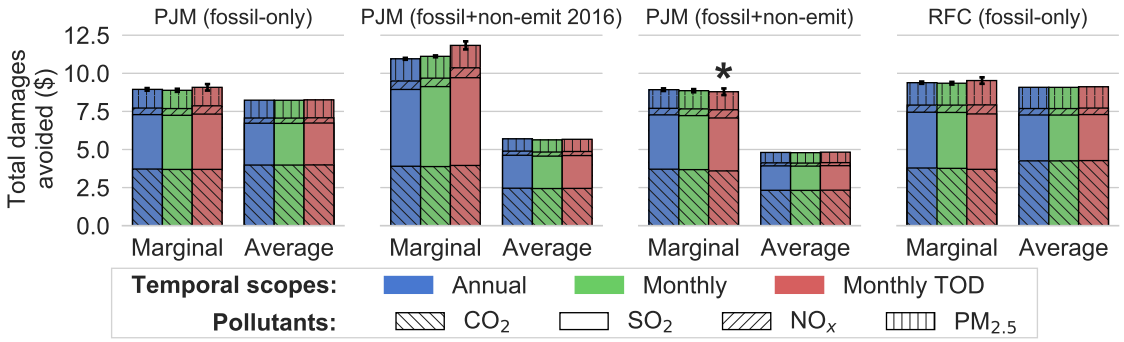
Overall, we see a great deal of intra-annual and intraday variability in PJM marginal emissions factors that is not captured by annual or monthly level factors. As such, we recommend using granular factors when appropriate in order to design and evaluate the effects of marginal interventions. However, as a caveat, monthly TOD factors exhibit much greater uncertainty (i.e. have higher standard errors) than monthly or annual factors, due to higher susceptibility to outliers. As such, modelers who use monthly TOD or other granular factors should explicitly propagate the associated standard errors in their estimates in order to be transparent about the underlying uncertainty in these estimates.

3.3.3 Effects of a building-level lighting intervention

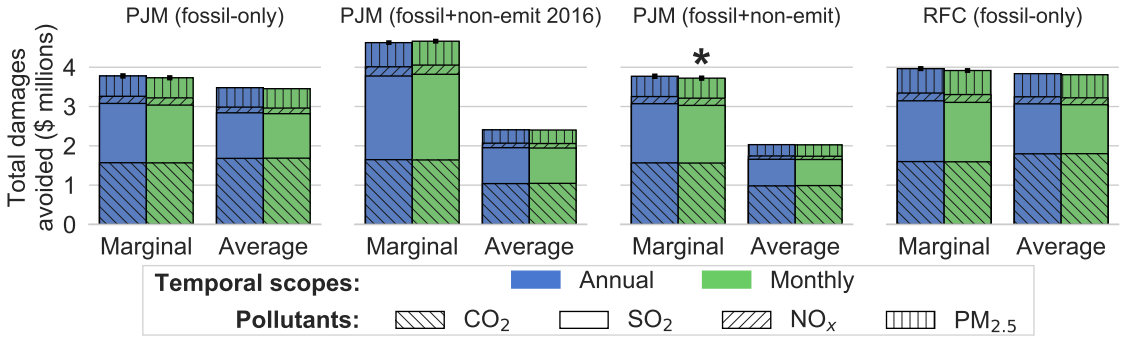
To illustrate the importance of different emissions and damage factor assumptions, we assess the effects of a simple building-level lighting intervention in 2017 under the different factor types discussed above. Specifically, we estimate the effects of reducing residential consumption by 100W from 8pm to midnight, e.g. by switching two 60W incandescent bulbs to 10W LEDs for indoor residential nighttime use. This scenario is simply meant to be illustrative of the magnitude of emissions and damage reductions possible, as more detailed assumptions about lamp types and usage patterns would be needed for an actual lighting policy evaluation.

Figure 3.4a shows the health, environmental, and climate change damages of this intervention under different emissions factors, using the AP2 damage model and a \$40/ton CO_2 cost of carbon. As we are measuring the hourly effects of marginal changes in this example, we use marginal monthly TOD fossil+non-emitting damage factors (for PJM in 2017) as a baseline. All percentage comparisons in this section are relative to this baseline.

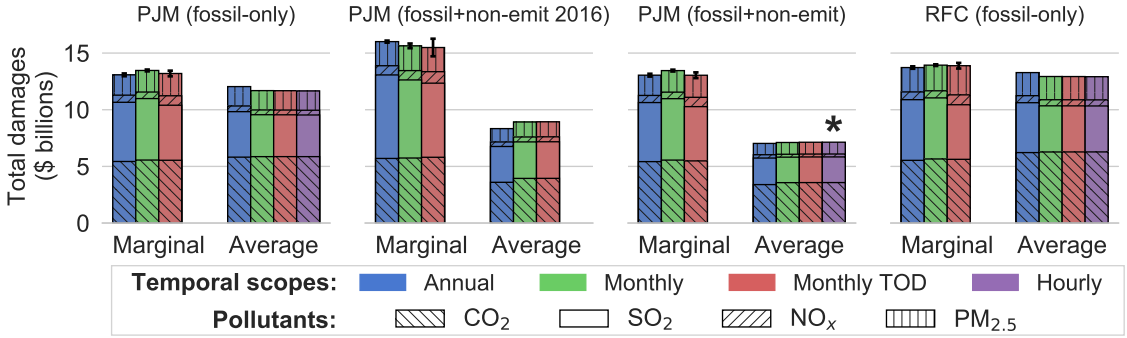
Several important points are highlighted with this simple example. First, we see a wide range of estimates for the damages avoided, ranging from \$4-12 depending on the factor used, with a baseline estimate of \$9. This difference is important: assuming this intervention were implemented in the approximately 25 million households in PJM (estimated given a population of 65 million in PJM [PJM18] and an average of 2.58 people per household [Uni12a]), the estimated annual social damages avoided would range from \$100 to 300 million.



(a) Total annual damages avoided for a 2017 nighttime building level lighting intervention in PJM that induces a daily 100W reduction from 8pm to midnight.



(b) Total damages avoided for 2017 PJM historical demand response, assuming complete load shedding. For context, PJM demand response revenue was $2.4 million (2010 dollars) in 2017 [McA18].



(c) Total damages from 2017 PJM summer metered load (June-August). For context, PJM's annual billings in 2016 were approximately $40 billion (2010 dollars) [PJM18]. As we do not estimate hourly-level marginal factors, and since hourly level 2016 factors should not be applied to 2017, we omit hourly-level estimates in these cases.

Figure 3.4: Effects of interventions and loads evaluated as assessed with damage factors under the AP2 damage model (incorporating health and climate change damages for CO_2, SO_2, NO_x, and $PM_{2.5}$, in 2010 dollars). Baseline factor effects are indicated with an asterisk. Error bars (narrow) represent propagated regression standard errors. PJM (fossil-only) = assessment with 2017 emissions factor estimates using only fossil fuel generation in PJM; PJM (fossil+non-emitting) = assessment with 2017 emissions factor estimates using fossil fuel and non-emitting generation in PJM; PJM (fossil+non-emitting 2016) = assessment with 2016 emissions factor estimates using fossil fuel and non-emitting generation in PJM; RFC (fossil-only) = assessment with 2017 emissions factor estimates using only fossil fuel generation in RFC.

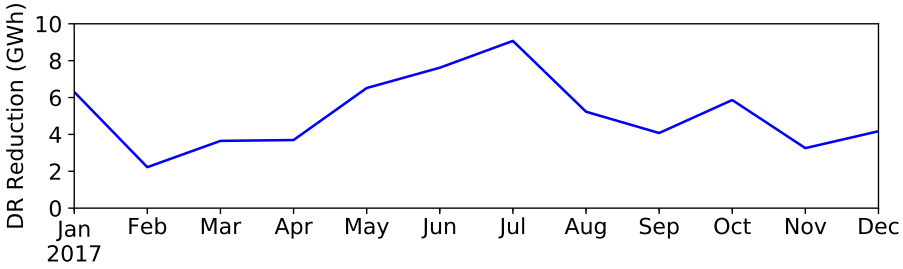


Figure 3.5: PJM's reported economic demand response by month in 2017. In 2017, monthly DR reductions ranged from 2 GWh (in February) to 9 GWh (in July), with a total of 62 GWh throughout the year [McA18]. (This amount corresponds to less than 0.01% of PJM's annual load.)

Our results also highlight that the distinction between marginal and average factors is the most influential in driving results for this intervention. Specifically, using average emissions factors grossly underestimates the damages avoided by this marginal intervention, especially with the inclusion of non-emitting generation. Indeed, the *average fossil+non-emitting* counterparts to the baseline underestimate damage reductions by 45%, and even the *average fossil-only* counterparts to the baseline underestimate damage reductions by 6%.

While average factors *underestimate* marginal intervention effectiveness in 2017, outdated factors significantly *overestimate* effectiveness. For instance, if a decision-maker employing the baseline monthly TOD fossil+non-emitting factors were to use (relatively recent) data from 2016 instead of 2017, they would overestimate the total damages avoided in this example by 35%. We choose 2016 for comparison since, at the time of this analysis (which we conducted over 2017-18), it was the data year for the EPA's most recent eGRID factors, which may have viably been used by policymakers for intervention design and emissions targets.

As suggested by our factor comparisons, the distinctions between PJM and RFC factor assessments are more modest (within 8%), as are the distinctions between assessments using PJM fossil-only vs. fossil+non-emitting marginal factors (within 3%). In this case, the distinction between annual, monthly, and monthly TOD factors is also less influential (less than 2%) since the intervention is implemented uniformly across the year and for multiple hours each day, causing intra-annual and intraday variations in factors to average out.

3.3.4 Effects of historical demand response

Demand response (DR) entails the participation of demand resources in energy and capacity markets, and can benefit the U.S. power system by curbing electricity emissions and costs [Wal+08] given real-time power grid information. Numerous studies have explored DR implementation in different settings [AES08; PD11; FIN18; Zha+15]. There has been a great deal of recent regulatory activity in this area; for instance, a 2016 Supreme Court order potentially encouraged DR in energy markets by allowing it to be compensated as electricity generation [Uni16b], but PJM meanwhile curbed DR's participation in the capacity market by imposing capacity performance requirements [PJM17b].

We analyze the historical emissions and damage effects of PJM's existing DR program as input to decision-making. PJM reports aggregated historical demand response at the

monthly level. In 2017, monthly DR reductions ranged from 2 GWh (in February) to 9 GWh (in July) with a total of 62 GWh throughout the year [McA18], which corresponds to less than 0.01% of PJM's annual load (see Figure 3.5). Since demand response in reality occurs at a granular time scale, monthly-level analyses may miss important effects stemming from intraday variation in marginal factors. We note further that we evaluate DR reductions as reported by PJM assuming complete load shedding, and do not account for potential load shifting or social damages from behind-the-meter generation used for DR (due to the lack of access to granular data). However, these effects can potentially have large impacts on demand response assessments (see Appendix A.4), warranting the release of more granular data to enable accurate assessments.

In the absence of more granular data, the assessed effects of monthly-level demand response for total damages under AP2 (assuming complete load shedding) are shown in Figure 3.4b. As we are measuring the monthly effects of marginal changes, we use marginal monthly fossil+non-emitting damage factors (for PJM in 2017) as our baseline. All percentage comparisons in this section are relative to this baseline.

We find that the annual health, environmental, and climate change damages avoided by demand response range from $2.0 to 4.7 million depending on the factor used, with a baseline estimate of $3.7 million. For context, PJM economic DR revenue – which gives a lower bound on avoided electricity sales – was $2.4 million (2010 dollars) in 2017 [McA18].

Similarly to the previous intervention, the average vs. marginal factor distinction and year of calculation are the most important assumptions driving results. The *average fossil+non-emitting* counterparts to the baseline underestimate total damages avoided by 46%, and even the *average fossil-only* counterparts to the baseline underestimate damages by 7%. On the other hand, the 2016 counterparts to the baseline *overestimate* total damages avoided by 25%.

In this case, we again see that the distinctions between PJM and RFC factors and fossil-only vs. fossil+non-emitting factors do not significantly drive estimates of damages avoided, leading to differences of less than 5% and less than 1%, respectively. The distinctions between annual and monthly factors are also not influential (less than 1% difference) in this case, as monthly-level demand response reductions are relatively evenly distributed throughout the year.

3.3.5 Effects of historical summer load

As a final example, we assess the emissions effects from historical load in PJM in the summer (June-August 2017), when demand is highest. Specifically, peak load during these summer months is 146 GW, but only 132 GW in the rest of the year (see Figure 3.6).

The assessed effects of this load for total damages under AP2 are shown in Figure 3.4c. Since we are measuring the hourly effects of *total load* as opposed to a marginal change, we use *average* fossil+non-emitting damage factors (for PJM in 2017) *computed for each hour of the year* as a baseline for this example. All percentage comparisons in this section are relative to this baseline.

We find that the health, environmental, and climate change damages of summer load range from $7.0 to 16 billion depending on the factor used, with a baseline estimate of $7.1

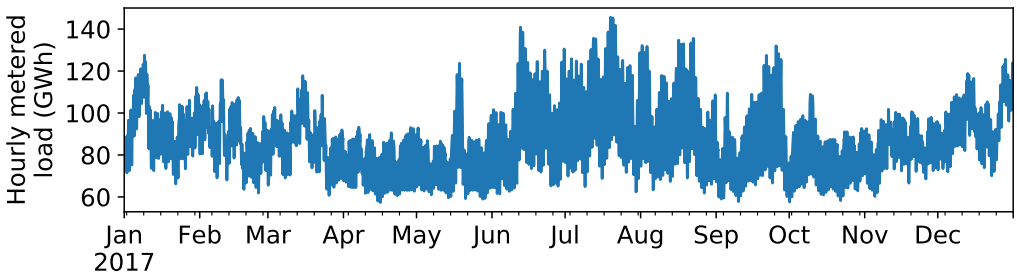


Figure 3.6: PJM hourly metered load in 2017. (Data from PJM Data Miner 2 [PJM17d].)

billion. For context, PJM's total billings in 2017 were approximately $40 billion [PJM18].

The choice to include or exclude non-emitting generation is *extremely* influential for assessments of total loads such as summer demand, as the fossil-only counterparts to the baseline overestimate damages avoided by 63%. The distinction between marginal and average factors also becomes more important with the inclusion of non-emitting generation, as the marginal monthly TOD counterparts to the baseline overestimate total damages by 83%. It is thus extremely important that policymakers consider whether including non-emitting generation makes sense for their particular application (instead of simply using published fossil+non-emitting average factors by default) and be particularly mindful of the distinction between average and marginal factors in the presence of nuclear and renewables.

As in the previous cases, using the 2016 monthly TOD counterparts to the baseline overestimates damages by 25%. The annual counterparts to the baseline only negligibly underestimate damages (since intra-annual variations in average fossil+non-emitting factors are averaged out over the summer months), and as per our analysis of monthly TOD factors, granularity beyond the monthly level does not affect damage assessments.

3.4 Policy implications

Our analysis of emissions and damage factors for PJM highlights the great importance of using appropriate assumptions about average vs. marginal factors, year of calculation, and inclusion of non-emitting generation (and the potential importance of assumptions such as temporal and regional scope) when evaluating interventions and loads. Such assumptions can significantly drive power system intervention assessments, potentially flipping policy decisions about intervention implementation, design, and incentives. In particular, we suggest that decision-makers employ marginal factors for marginal interventions and average factors for total load assessments, especially since public marginal emissions factor estimates are now readily available for the United States [Aze+21]. If there is ambiguity as to which factor is appropriate, we recommend conducting a sensitivity analysis across both types of factors. We also suggest that regulators and PJM decision-makers update emissions and damage factors frequently (e.g., at least once per year) and always use emissions factors from the year in which the intervention under evaluation was implemented. Similarly, for the design of future interventions, we recommend that decision-makers develop and

use accurate forecasting methods for grid emissions factors. These recommendations are particularly important given that average factors change at different rates than marginal factors, leading the relative effectiveness of average vs. marginal interventions to differ over time. The interactions between cap-and-trade and marginal factor calculations should also be examined over time, as caps should be set to reflect marginal (not average) impacts.

While we find that intra-annual and intraday variability is not influential in our assessments of interventions and loads, we find that both marginal and average factors do vary throughout the year, and marginal factors vary quite a bit throughout the day. We suggest that policymakers examine the intraday and intra-annual variability in emissions factors to design incentives that target interventions at high-intensity (and thereby high-impact) times of the day and year.

Nuclear and renewable generation currently only minimally affect PJM MEFs, but this conclusion may change with the increasing prevalence of renewables. Our method of regression with dummy variables is also only one potential way to include non-emitting sources in marginal emissions estimates, and relies on data about when different fuel sources are marginal (which may not be available in all desired regions of analysis). Important future work would involve exploring different methods of including non-emitting sources in marginal factor estimates, particularly ones that can detect whether non-emitting generation is at the margin given hourly generation and emissions from system generators.

While our work assesses sensitivities for generation-based emissions and damage factors, we do not consider the differences driven by distinctions between *generation*-based factors and *demand*-based factors [GZKM14]. In 2017, PJM total (real-time and day-ahead) gross imports and exports were 44 TWh and 87 TWh, respectively [Ana17], which corresponds to about 6% and 11% of PJM's annual load served. The distinction between generation-side and demand-side factors could thus potentially impact intervention estimates and would be worth examining in future work.

We finally note that the kinds of emissions-reducing interventions and loads studied here are equally important from both the climate change and public health perspectives. In particular, climate change-related damages and health-related damages each comprise about half of the total damage factors in 2017. As such, we suggest that policymakers in both the climate change and health domains consider the insights provided by our work when selecting baseline factors for the assessment of power system interventions.

Chapter 4

Matrix Completion for Distribution System Voltage Estimation

With the rising penetration of distributed energy resources, distribution system control and enabling techniques such as state estimation have become essential to distribution system operation. However, traditional state estimation techniques have difficulty coping with the *low-observability* conditions often present on the distribution system due to the paucity of sensors and heterogeneity of measurements in most systems. To address these limitations, we propose a distribution system state estimation algorithm that employs matrix completion (a tool for estimating missing values in low-rank matrices) augmented with noise-resilient power flow constraints. This method operates under low-observability conditions where standard least-squares-based methods cannot operate, and flexibly incorporates any network quantities measured in the field. We empirically evaluate our method on the IEEE 33- and 123-bus test systems, and find that it provides near-perfect state estimation performance (within 1% mean absolute percent error) across many low-observability data availability regimes. This work is now being used within collaborations between the National Renewable Energy Laboratory (NREL) and the Hawaiian Electric Company (HECO), as HECO contends with increased penetrations of distributed solar power on its system.

The work in this chapter has previously been published in:

> Priya L. Donti, Yajing Liu, Andreas J. Schmitt, Andrey Bernstein, Rui Yang, and Yingchen Zhang. "Matrix Completion for Low-Observability Voltage Estimation." *IEEE Transactions on Smart Grid* 11.3 (2019), 2520–2530.

4.1 Introduction

State estimation is one of the most critical inference tasks in power systems. Classically, it entails estimating voltage phasors at all buses in a network given some noisy and/or bad data from the network [Lia82]. Estimates are obtained via the (generally non-linear) measurement model:

$$z = h(x) + \epsilon, \tag{4.1}$$

where $z \in \mathbb{C}^m$ is a vector of measurements, $x \in \mathbb{C}^n$ is a vector of quantities to estimate (typically, voltage phasors), $h(\cdot)$ is a vector of functions representing the system physics (i.e., power-flow equations), and ϵ is a vector of measurement noise. The state-estimation task is then to estimate x given z and some knowledge of $h(\cdot)$ (e.g., its Jacobian matrix). State estimation has been thoroughly addressed in transmission networks, wherein system (4.1) is typically *overdetermined* and *fully observable*: that is, (i) the number of measurements m is at least the number of unknowns n, and (ii) the Jacobian $J \in \mathbb{C}^{m \times n}$ of $h(\cdot)$ is (pseudo) invertible in the sense that $(J^T J)^{-1}$ exists. As transmission systems conventionally have redundant measurements that satisfy the observability requirement, classical least-squares estimators are applicable and can operate efficiently [AE04].

In contrast, the use of state estimation has historically been limited in distribution networks [Deh+18]. Due to limited availability of real-time measurements from Supervisory Control and Data Acquisition (SCADA) systems, Equation (4.1) is typically *underdetermined* ($m < n$), rendering standard least-squares methods inapplicable. Accurate distribution system state estimation was also previously unnecessary since distribution networks only delivered power in one direction towards the customer, requiring minimal distribution system control. This led industry to in practice use only simple heuristics (e.g. based on simple load-allocation rules [DHZ02; PSM04]) to roughly calculate power flow.

However, due to the increasing adoption of distributed energy resources (DERs) at the edge of the network [DK08], distribution system state estimation has become increasingly important [PL17]. There is thus a large focus in the literature on low-observability state estimation techniques. Many existing methods attempt to improve system observability, e.g., by optimizing the placement of additional system sensors [SPV09; BKV18; JZ16] or by deriving pseudo-measurements from existing sensor data [Man+12; WHJ13]. Unfortunately, installation of additional sensors may be expensive or slow, and pseudo-measurements can introduce estimation errors [Cle11] or be extremely data-intensive to obtain [Man+12; JZ16]. Other methods seek to perform state estimation using neural networks, without constructing an underlying system model [Per+16]. While such machine learning methods can obtain accurate estimation results, training these methods requires a significant amount of historical data, which may not be available. As such, there is a need for state estimation methods that can exploit problem structure to perform state estimation at current levels of data availability and observability.

In this chapter, we propose a low-observability state estimation algorithm based on *matrix completion* [CR09], a tool for estimating missing values in low-rank matrices. We apply this tool to state estimation for a given time step by forming a structured data matrix whose rows correspond to measurement locations, and whose columns correspond to measurement

types (e.g., voltage or power). While some methods require collecting data over large time windows [BKV18], our approach enables "single shot" state estimation that employs only data from a single time instance. Our approach is closely related to recent works [Gao+16; Gen+18; Lia+19] that use matrix completion to estimate lost phasor measurement unit (PMU) data over a time series, but while these works estimate missing quantities exclusively at measurement points, we consider the problem of estimating quantities even at non-measurement points where the quantities to be estimated may never have been measured.

The main contributions of this chapter are:

- A novel distribution system state estimation method based on constrained matrix completion. By augmenting matrix completion with noise-resilient power flow constraints, the proposed method can accurately estimate voltage phasors under low-observability conditions where standard (least-squares) methods cannot.
- A flexible framework for employing various types of distribution system measurements into state estimation. Whereas many works (e.g. [KZ15; WHJ13; BKV18]) require specific measurements for estimation, our approach can accommodate any quantities measured in the field.
- An empirical demonstration of the robustness of our method to data availability and measurement loss.

4.2 Matrix completion methods

We start by introducing *constrained matrix completion*, a method that is central to our proposed approach.

4.2.1 Matrix completion

Given an incomplete matrix that is assumed to be low-rank, the matrix completion problem aims to determine the unknown elements in this matrix. Formally, let $M \in \mathbb{R}^{n_1 \times n_2}$ be a real-valued data matrix; $\Psi \subseteq \{1, \dots, n_1\} \times \{1, \dots, n_2\}$ describe the known elements in M; and $M_\Psi \in \mathbb{R}^{n_1 \times n_2}$ denote the observation matrix, where $(M_\Psi)_{j,k} = M_{j,k}$ for $(j,k) \in \Psi$ and 0 otherwise. Matrix completion can then be formulated as a rank minimization [CR09]:

$$\begin{aligned} \underset{X \in \mathbb{R}^{n_1 \times n_2}}{\text{minimize}} \quad & \operatorname{rank}(X) \\ \text{subject to} \quad & X_\Psi = M_\Psi, \end{aligned} \tag{4.2}$$

where the decision variable X estimates M. As the optimization problem (4.2) is NP-hard due to the non-convexity of the rank function, it is common to use a heuristic approach that instead minimizes the *nuclear norm* of the matrix [CR09]:

$$\begin{aligned} \underset{X \in \mathbb{R}^{n_1 \times n_2}}{\text{minimize}} \quad & \|X\|_* \\ \text{subject to} \quad & X_\Psi = M_\Psi, \end{aligned} \tag{4.3}$$

where $\|X\|_*$ sums the singular values of X. Given a sufficient number of randomly-sampled entries in M_Ψ (depending on the matrix size and rank), problem (4.3) often has a unique minimizer X that equals M [CR09]. In practice, this problem can be solved efficiently using truncated nuclear norm regularization or other methods [Hu+13; KMO10; Gro11].

Due to the nature of the equality constraint, formulation (4.3) is highly susceptible to noise. To alleviate this problem, Candes and Plan [CP10] proposed an algorithm to handle noisy measurements. The algorithm modifies the equality constraint in (4.3) to

$$\|X_\Psi - M_\Psi\|_F \leq \delta, \tag{4.4}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\delta \geq 0$ is a parameter that can be tuned based on the extent of measurement noise.

4.2.2 Constrained matrix completion

Now suppose that the values in the matrix M come from some physical system (e.g., a power system). It is then natural to extend formulation (4.3)/(4.4) to incorporate system physics via the following *constrained* optimization problem:

$$\begin{aligned} \underset{X \in \mathbb{R}^{n_1 \times n_2}}{\text{minimize}} \quad & \|X\|_* \\ \text{subject to} \quad & \|X_\Psi - M_\Psi\|_F \leq \delta, \\ & \|g(X)\|_2 \leq \beta, \end{aligned} \tag{4.5}$$

for $\delta, \beta \geq 0$, where $g(\cdot)$ is a vector of functions representing system physics (e.g., power-flow equations). We note that:

- The additional constraint $\|g(x)\|_2 \leq \beta$ incentivizes low-rank solutions that respect the system physics.
- The choice of δ and β is problem-dependent. These parameters can be chosen based on the extent of measurement noise, or the objective function can be augmented with terms that try to minimize their values.
- If $g(\cdot)$ is nonlinear, (4.5) is typically non-convex and computationally challenging.

4.3 Low-observability state estimation

We now present our low-observability state estimation algorithm, which employs the constrained matrix completion model (4.5). We describe our power system model, possible formulations of M, and possible physical constraints $g(\cdot)$ before showing our full formulation.

4.3.1 Power system model

Let $\mathcal{B}$ denote the set of buses, where bus 1 is the slack bus and the remaining $|\mathcal{B}| - 1$ buses are PQ buses.[1] Further, let $m \subseteq \mathcal{B} \times \mathcal{B}$ denote the set of distribution lines. We describe

[1] The *slack (or reference) bus* is a power generation bus that, by mathematical convention, provides a reference voltage angle for the rest of the system; in the context of distribution systems, the slack

the nodal admittance matrix $W \in \mathbb{C}^{|\mathcal{B}|\times|\mathcal{B}|}$ in block form as

$$W = \begin{bmatrix} W_{11} \in \mathbb{C} & W_{1L} \in \mathbb{C}^{1\times(|\mathcal{B}|-1)} \\ W_{L1} \in \mathbb{C}^{(|\mathcal{B}|-1)\times 1} & W_{LL} \in \mathbb{C}^{(|\mathcal{B}|-1)\times(|\mathcal{B}|-1)} \end{bmatrix}.$$

Let $v \in \mathbb{C}^{|\mathcal{B}|}$ and $s \in \mathbb{C}^{|\mathcal{B}|}$ be the vectors of (partially unknown) voltage phasors and net complex power injections, respectively, at each bus. We denote the slack bus voltage phasor and power injection as v_1 and s_1, respectively, and similarly denote the vectors of non-slack bus voltages and power injections as v_{-1} and s_{-1}. Finally, let $\iota \in \mathbb{C}^{|\mathcal{L}|}$ be the vector of (partially unknown) complex currents in each *branch* (i.e., line on the power system), where ι_{ft} is the current in line $(f,t) \in \mathcal{L}$.

4.3.2 Data matrix formulation

The formulation of the data matrix M (and thus the optimization variable X) can vary based on the particular attributes of the problem setting, e.g., the kinds of measurements available and problem scale. We present two possible formulations here, one indexed by branches and one indexed by buses. However, we emphasize that *the proposed method is not limited to using these matrix structures*. The matrix M can be flexibly structured to accommodate available measurements, as long as these measurements are correlated so that M is (approximately) low rank.

4.3.2.1 Branch formulation

M can be structured such that each row represents a power system branch and each column represents a quantity relevant to that branch. This structure allows us to take advantage of both bus- and branch-related measurements. Specifically, for every line $(f,t) \in \mathcal{L}$, the corresponding row in the matrix $M \in \mathbb{R}^{n_1\times n_2}$ contains:

$$\begin{aligned}[&\operatorname{Re}(v_f),\ \operatorname{Im}(v_f),\ |v_f|,\ \operatorname{Re}(s_f),\ \operatorname{Im}(s_f),\ \operatorname{Re}(v_t),\\ &\operatorname{Im}(v_t),\ |v_t|,\ \operatorname{Re}(s_t),\ \operatorname{Im}(s_t),\ \operatorname{Re}(\iota_{ft}),\ \operatorname{Im}(\iota_{ft})\ ],\end{aligned}$$

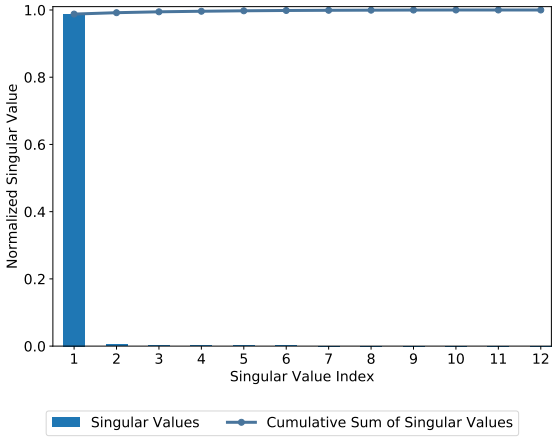
where $n_1 = |\mathcal{L}|$ and we employ $n_2 = 12$ quantities per row, and where Re and Im refer to the real and imaginary parts, respectively, of complex phasors.

4.3.2.2 Bus formulation

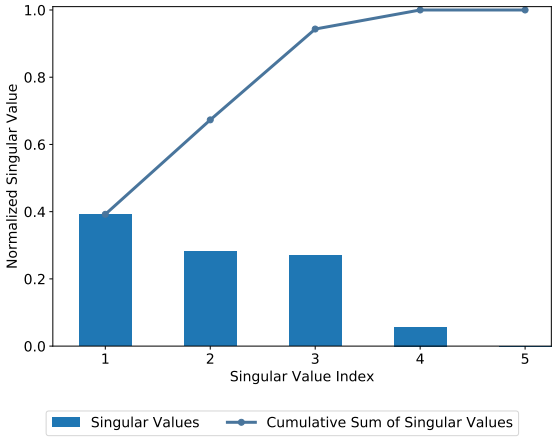
M can also be structured such that each row represents a bus and each column represents a quantity relevant to that bus. That is, for every bus $b \in \mathcal{B}$, the corresponding row in the matrix $M \in \mathbb{R}^{n_1\times n_2}$ contains:

$$[\operatorname{Re}(v_b),\ \operatorname{Im}(v_b),\ |v_b|,\ \operatorname{Re}(s_b),\ \operatorname{Im}(s_b)],$$

bus usually refers to the distribution substation, i.e., the bus connecting the distribution system to the transmission system. The terminology *PQ bus* refers to a demand bus without any controllable power generation capacity (but which may have, e.g., variable rooftop solar power).



(a) IEEE 33-bus feeder (branch formulation; dim. 32×12). The largest singular value (out of 12) comprises >98% of the singular value sum.



(b) IEEE 123-bus feeder (multi-phase bus formulation; dim. 263×5). The first 3 (of 5) largest singular values comprise 95% of the singular value sum.

Figure 4.1: Singular values of data matrices for the IEEE 33-bus and 123-bus systems. The bars show the individual singular values (normalized by the singular value sum), and the circles show the cumulative sums of the (normalized) singular values.

where $n_1 = |\mathcal{B}|$ and we employ $n_2 = 5$ quantities per row. While this structure only employs bus-related measurements, its advantage is that it yields small matrices that can be used for efficient estimation on large-scale problems.

4.3.2.3 On the low-rank assumption

To employ the matrix completion framework described in Section 4.2, it is necessary that M be (approximately) low rank. Roughly speaking, this means that M must be structured such that its rows and/or columns are correlated.

The branch and bus formulations we propose here structure M such that each row represents a location (branch or bus) and each column represents a type of measurement (voltage, power, etc.) There are two types of correlations that are possible in this case. The first is among the rows, representing the spatial correlation between different locations in a power grid; this property is system dependent (e.g., correlation between loads at neighboring buses). The second type of possible correlation is among columns, representing the correlation between different types of measurements; this type of correlation is more general, as implied by the fact that the power system equations employing these measurements can be expressed in (approximate) linear form [BW89; BD17] (see Section 4.3.3). Therefore, these data matrix formulations should be approximately low-rank.

We observe empirically that this low-rank property holds in practice. Figure 4.1 shows the cumulative percentage distribution of singular values for the IEEE 33-bus feeder [MAT20] (using a branch formulation) and the IEEE 123-bus feeder [IEE10b] (using a bus formulation). In both cases, we see that a few singular values comprise much of the singular value sum, implying these matrices are (approximately) low-rank.

4.3.3 Physical power flow constraints

As described in Section 4.2.2, we augment matrix completion with power system constraints to encourage physically meaningful solutions. These constraints are linear to ensure that problem (4.5) is convex. We describe the constraints we use below, but note that constraints can be added, removed, or modified depending on the types of measurements in M.

4.3.3.1 Duplication constraint

Depending on the formulation, some quantities may appear in more than one location in M. For example, in our branch formulation, quantities related to a given bus appear in multiple rows if the bus is in multiple branches. We thus constrain equivalent quantities in the matrix to be equal. Formally, let Λ be a set containing the indices of all pairs of duplicated quantities in M, such that $((\lambda_1, \lambda_2), (\lambda_1', \lambda_2')) \in \Lambda \iff M_{\lambda_1,\lambda_2} \equiv M_{\lambda_1',\lambda_2'}$. We require that

$$X_{\lambda_1,\lambda_2} = X_{\lambda_1',\lambda_2'},\ \forall\, ((\lambda_1, \lambda_2), (\lambda_1', \lambda_2')) \in \Lambda. \tag{4.6}$$

4.3.3.2 Ohm's law constraint

When M contains both bus- and branch-related quantities (as in the branch formulation), we can apply Ohm's Law, defined as

$$(v_f - v_t)w_{ft} = \iota_{ft},\ \forall\, (f,t) \in \mathcal{L}, \tag{4.7}$$

where w_{ft} is the line admittance. However, using an exact equality constraint may cause the matrix completion problem to become infeasible, e.g., due to measurement noise. We thus employ a noise-resilient version of Ohm's Law, i.e.,

$$\begin{bmatrix} -\xi_{r,ft} \\ -\xi_{c,ft} \end{bmatrix} \leq \begin{bmatrix} \mathrm{Re}\left((v_f - v_t)w_{ft} - \iota_{ft}\right) \\ \mathrm{Im}\left((v_f - v_t)w_{ft} - \iota_{ft}\right) \end{bmatrix} \leq \begin{bmatrix} \xi_{r,ft} \\ \xi_{c,ft} \end{bmatrix}, \tag{4.8}$$

where $\xi_{r,ft},\ \xi_{c,ft} \in \mathbb{R}_+$ are respective error tolerances for the real and complex parts of Ohm's Law on line $(f,t) \in \mathcal{L}$.

4.3.3.3 Linearized power flow constraints

As the exact AC power flow equations are non-linear (see Section 2.3.2), we employ Cartesian linearizations of these equations. For non-slack voltages and power injections, we employ approximations of the form

$$v_{-1} \approx A \begin{bmatrix} \mathrm{Re}(s_{-1}) \\ \mathrm{Im}(s_{-1}) \end{bmatrix} + u, \tag{4.9a}$$

$$|v_{-1}| \approx C \begin{bmatrix} \mathrm{Re}(s_{-1}) \\ \mathrm{Im}(s_{-1}) \end{bmatrix} + |u|. \tag{4.9b}$$

For example, using the method proposed in Bernstein and Dall'Anese [BD17], we can let $u = -v_1 W_{LL}^{-1} W_{L1} \in \mathbb{C}^{|\mathcal{B}|-1}$ be the vector of non-slack zero-load voltages and $A,\ C \in \mathbb{C}^{(|\mathcal{B}|-1)\times 2(|\mathcal{B}|-1)}$ be defined for some non-slack voltage estimates $\hat{v}_{-1}$ as

$$A = \begin{bmatrix} W_{LL}^{-1} \operatorname{diag}(\overline{\hat{v}}_{-1})^{-1} & -jW_{LL}^{-1} \operatorname{diag}(\overline{\hat{v}}_{-1})^{-1} \end{bmatrix}, \tag{4.10a}$$

$$C = \operatorname{diag}(|\hat{v}_{-1}|)^{-1} \operatorname{Re}\left(\operatorname{diag}(|\overline{\hat{v}}_{-1}|) A\right). \tag{4.10b}$$

For our case, we let $\hat{v}_{-1} = u$. We note, however, that other methods to obtain the linear approximations (4.9) (e.g., data-driven regression methods [Liu+18]) can also be leveraged.

To relate voltages with the power injection at the slack bus, we employ the exact power flow equation

$$s_1 = v_1(\overline{W}_{11}\overline{v}_1 + \overline{W}_{1L}\overline{v}_{-1}). \tag{4.11}$$

This equation is linear in voltages since v_1 is known.

As in Section 4.3.3.2, we relax these constraints into noise-resilient versions as

$$\begin{bmatrix} -\tau_r \\ -\tau_c \end{bmatrix} \leq \begin{bmatrix} \operatorname{Re}\left(v_{-1} - \left(A \begin{bmatrix} \operatorname{Re}(s_{-1}) \\ \operatorname{Im}(s_{-1}) \end{bmatrix} + u\right)\right) \\ \operatorname{Im}\left(v_{-1} - \left(A \begin{bmatrix} \operatorname{Re}(s_{-1}) \\ \operatorname{Im}(s_{-1}) \end{bmatrix} + u\right)\right) \end{bmatrix} \leq \begin{bmatrix} \tau_r \\ \tau_c \end{bmatrix}, \tag{4.12a}$$

$$-\gamma \leq |v_{-1}| - \left(C \begin{bmatrix} \operatorname{Re}(s_{-1}) \\ \operatorname{Im}(s_{-1}) \end{bmatrix} + |u|\right) \leq \gamma, \tag{4.12b}$$

$$\begin{bmatrix} -\alpha_r \\ -\alpha_c \end{bmatrix} \leq \begin{bmatrix} \operatorname{Re}\left(s_1 - \left(v_1(\overline{W}_{11}\overline{v}_1 + \overline{W}_{1L}\overline{v}_{-1})\right)\right) \\ \operatorname{Im}\left(s_1 - \left(v_1(\overline{W}_{11}\overline{v}_1 + \overline{W}_{1L}\overline{v}_{-1})\right)\right) \end{bmatrix} \leq \begin{bmatrix} \alpha_r \\ \alpha_c \end{bmatrix}, \tag{4.12c}$$

where $\tau_r, \tau_c, \gamma \in \mathbb{R}_+^{|\mathcal{B}|-1}$, $\alpha_r, \alpha_c \in \mathbb{R}_+$ are error tolerances, and inequalities are evaluated elementwise.

4.3.4 Full problem formulation

Given these power flow constraints, we collect our error tolerances into the set $\mathcal{E} = \{\xi_r, \xi_c, \tau_r, \tau_c, \gamma, \alpha_r, \alpha_c\}$ and form our constrained matrix completion problem (4.5) as

$$\underset{X \in \mathbb{R}^{n_1 \times n_2},\, \mathcal{E}}{\text{minimize}} \quad \|X\|_* + \sum_{\epsilon \in \mathcal{E}} w_\epsilon \|\epsilon\|_2 \tag{4.13a}$$

$$\text{subject to} \quad \|X_\Psi - M_\Psi\|_F \leq \delta, \tag{4.13b}$$

$$(4.6), (4.8), (4.12a), (4.12b), (4.12c), \tag{4.13c}$$

$$\epsilon \geq 0, \quad \forall \epsilon \in \mathcal{E}, \tag{4.13d}$$

where in this case we add each constraint tolerance $\epsilon \in \mathcal{E}$ to the objective with an associated weight w_ϵ. Each weight is chosen to reflect the relative importance of its constraint. For a branch-formulated M, the above formulation can be used as-is. For a bus-formulated M,

equations (4.6) and (4.8) are removed from the constraints (and their associated parameters ξ_r, ξ_c are removed from $\mathcal{E}$) since M does not contain duplicated quantities or branch current measurements.

Formulation (4.13) allows the matrix completion optimization to explicitly trade off between the low-rank assumption and fidelity to the power flow constraints, without requiring tuning of each entry of each constraint tolerance vector. We further observe that, since the objective is convex and all constraints are linear in the entries of X, this formulation is a convex optimization problem and can be solved efficiently.

We note that our approach is closely related to *regularized least-squares* methods. Indeed, for underdetermined systems, regularization is typically used to bring structure into the problem and identify a unique meaningful solution. For example, ridge regression employs ℓ_2-norm regularization, whereas the popular LASSO regression technique employs ℓ_1-norm regularization to promote sparse solutions. Our approach can be viewed as a variant of a least-squares problem with *nuclear norm regularization* in order to promote solutions that are low rank.

In practice, we formulate (4.13) as a semidefinite program (SDP) to solve it using an SDP solver. In particular, as described in [CR09], the constrained matrix completion problem (4.13) can be rewritten as

$$\begin{aligned}
\underset{\substack{X\in\mathbb{R}^{n_1\times n_2},\,\mathcal{E},\\ D_1\in\mathbb{R}^{n_1\times n_1},\\ D_2\in\mathbb{R}^{n_2\times n_2}}}{\text{minimize}} \quad & \operatorname{trace}(D_1)+\operatorname{trace}(D_2)+\sum_{\epsilon\in\mathcal{E}} w_\epsilon\|\epsilon\|_2 && \text{(4.14a)}\\
\text{subject to} \quad & \|X_\Psi - M_\Psi\|_F \le \delta, && \text{(4.14b)}\\
& (4.6), (4.8), (4.12a), (4.12b), (4.12c), && \text{(4.14c)}\\
& \epsilon \ge 0, \quad \forall \epsilon \in \mathcal{E}, && \text{(4.14d)}\\
& \begin{bmatrix} D_1 & X \\ X^T & D_2 \end{bmatrix} \succeq 0, && \text{(4.14e)}
\end{aligned}$$

where the optimization variables are now X and $\mathcal{E}$ (as before) and additionally the matrices D_1 and D_2. As this formulation is an SDP, it can be solved using a standard SDP solver.

4.3.5 Extension to the multi-phase setting

While for brevity we formally present only a single-phase, balanced formulation, our approach can easily be extended to the general *multi-phase* setting. Specifically, M can be structured to include phase-wise quantities, and the constraints presented in Section 4.3.3 can be replaced with multi-phase versions (see, e.g., [BD17; Ber+18]). We empirically illustrate the application of our method to a multi-phase 123-bus test case in Section 4.4.2.

4.4 Simulation and results

We demonstrate the voltage estimation performance of our matrix completion method on the IEEE 33-bus [MAT20] and 123-bus [IEE10b] test cases. For the 33-bus feeder, we employ

a branch formulation matrix and show that our method performs well in low-observability settings (where traditional state estimation techniques cannot operate) as well as in full-observability settings. For the multi-phase 123-bus feeder, we use a bus formulation matrix and demonstrate that our method scales robustly to larger systems.

Implementation details. In all cases, we implement the SDP formulation (4.14) of our matrix completion algorithm using the CVXPY Python library [DB16], and solve it using CVXPY's SCS solver. For the 33-bus test case, we set the weights w_ϵ, $\epsilon \in \{\xi_r, \xi_c\}$ of the Ohm's Law parameters to be 100, and the weights w_ϵ, $\epsilon \in \{\tau_r, \tau_c, \gamma, \alpha_r, \alpha_c\}$ of the linearized power flow parameters to be 10, with the rationale that the Ohm's Law equations are exact whereas the linearized power flow equations are approximate. For the 123-bus case, we set the weights of the linearized power flow parameters to be 20 (and do not employ Ohm's Law constraints since we use a bus-formulated data matrix). For both the 33- and 123-bus test cases, we let δ be the value of the noise standard deviation (e.g. for 1% noise, we let $\delta = 0.01$).

4.4.1 33-bus system

We test our branch-formulated algorithm on a modified version of the IEEE 33-bus test case with solar panels added at buses 16, 23, and 31. On this system, voltage magnitudes range from approximately 0.99-1.02 p.u., and all angles (relative to the substation voltage angle) are close to 0. We assume that voltage phasors are known exactly at the slack bus, and must be estimated everywhere else. Non-voltage phasor quantities (i.e., voltage magnitude, power injections, and current flows) are assumed to be known exactly at the slack bus, and are "potentially measured" at other buses.

We compare our matrix completion method on this system against a state-of-the-art weighted least squares (WLS) state estimation algorithm. This algorithm sets up a (nonlinear) system of equations that captures relationships between voltage magnitudes/angles and measured quantities using Ohm's Law and the (original, nonlinear) power flow equations. Starting from an initial guess of 1 p.u. for unknown voltage magnitudes and 0 degrees for unknown voltage angles, WLS iteratively solves linearizations of this system of equations (using the Jacobian about the previous guess for voltage magnitudes and angles) until convergence. We use a base weight of 100 for each observation (since both Ohm's Law and the power flow equations are exact in this case), and adjust these weights to reflect the extent of measurement noise (i.e. down-weight each observation in direct proportion to its variance, as is standard in WLS). We implement the WLS baseline using the (nonlinear) least squares function from the Python library SciPy. For each experimental run during which matrix completion and WLS are compared, these algorithms are run in parallel using the same inputs.

4.4.1.1 Randomly-sampled data

In one set of experiments, we model data unavailability among "potentially measured" quantities via random sampling. That is, we randomly choose sets of buses (ranging between 0-100% of buses) at which all "potentially measured" quantities are measured, and set these

quantities to be unknown at all other buses. Results under 1% Gaussian measurement noise are shown in Figure 4.2c.

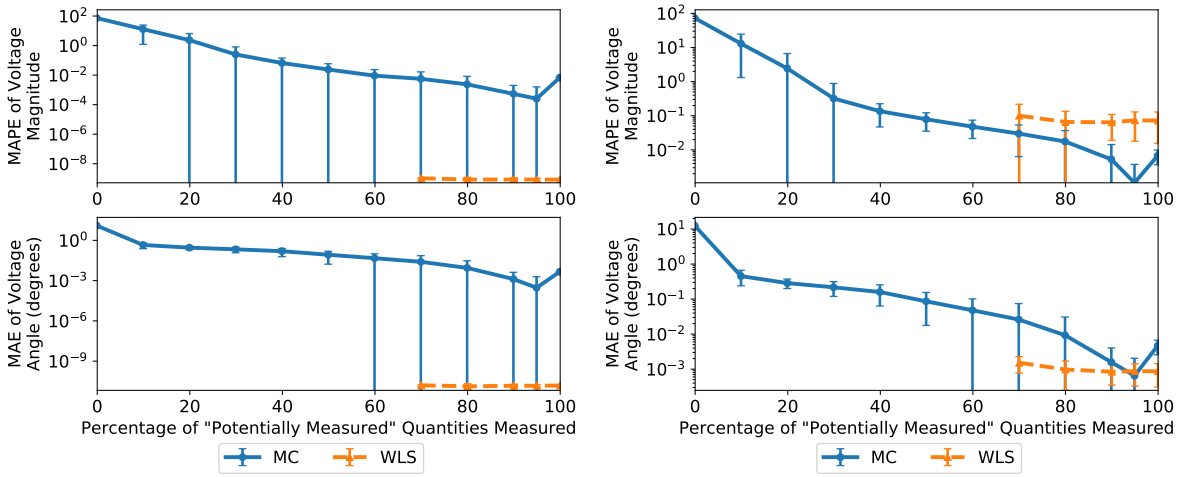
Under 1% measurement noise, we see that the mean absolute percent error (MAPE) of our matrix completion algorithm's voltage magnitude estimates drops to below 3% and the mean absolute error (MAE) of our voltage angle estimates drops to below 0.29 degrees when 20% or more of measurements are known. (We report MAEs rather than MAPEs for voltage angles since all angles are close to 0.) When 30% or more of measurements are known, voltage magnitude MAPEs are much less than 1%, and voltage angle MAEs are less than 0.23 degrees. This result calibrates well with guarantees for matrix completion performance under random data removal [CP10]. In contrast, we find that WLS is not able to operate on this system until approximately 70% of "potentially measured" quantities are measured, as the WLS Jacobian was never (pseudo) invertible at lower data availabilities during our experiments (see condition (ii) for observability described in Section 4.1). At the full observability conditions under which WLS is able to operate, our method's estimates are statistically similar to those of WLS, with voltage magnitude MAPEs and angle MAEs within 0.2% and 0.03 degrees, respectively, for our matrix completion algorithm and within 0.5% and 0.008 degrees, respectively, for WLS.

Given that grid sensors may exhibit varying degrees of measurement noise, we further examine the performance of our matrix completion algorithm and the WLS baseline under different noise levels (though assuming, as previously stated, that non-voltage phasor quantities are known exactly at the slack bus). Figures 4.2a, 4.2b, and 4.2d show the performance of these algorithms on the 33-bus test case (given randomly sampled measurements) under 0%, 0.2%, and 10% noise. While estimation performance does degrade as measurement noise increases, it does not degrade linearly in the amount of measurement noise; in other words, both our matrix completion algorithm and WLS are able to buffer against measurement noise to some extent.

4.4.1.2 Data-driven assumptions

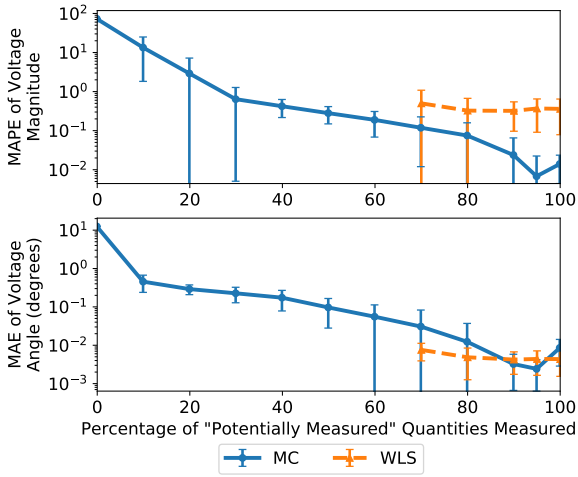
In practice, data unavailability is not uniformly random, but instead correlated and system-specific. For instance, a utility may only have certain types of sensors at certain types of buses. We thus run a second set of experiments where we classify the buses into four categories: slack bus (1 bus), solar generators (3 buses), large loads (6 buses), and small loads (23 buses). As before, all quantities are known exactly at the slack bus. At solar PV generators, only real power injections, reactive power injections, and voltage magnitudes may be measured. At loads (large or small), only real power injections may be measured. Since actual sensor availability may vary between utilities, we model different scenarios in which these groups of non-slack buses have real, pseudo-, or no measurements. In the best case (when all three groups of buses have some measurements), 23% of "potentially measured" quantities are measured. Thus, *all our data-driven scenarios are at low observability*, and traditional full-observability state estimation methods cannot be used.

The performance of our method on these scenarios is shown in Figure 4.3, assuming measurements have 1% Gaussian sensor noise and pseudomeasurements have 10% Gaussian error. Our algorithm achieves less than 1% MAPE in its magnitude estimates and less than

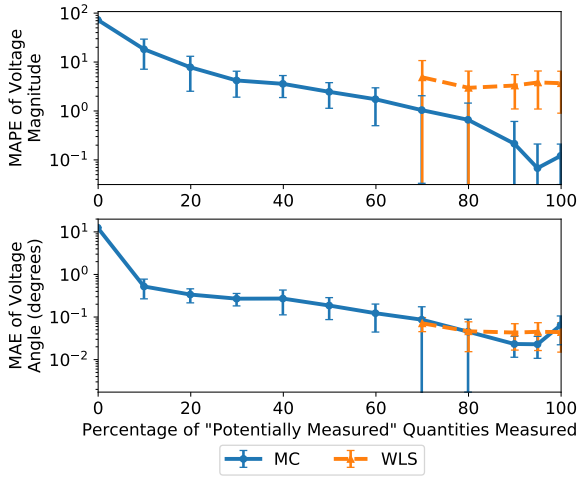


(a) 0% noise

(b) 0.2% noise



(c) 1% noise



(d) 10% noise

Figure 4.2: Performance on 33-bus test case with different levels of noise and random sampling of data. (Each point represents 50 runs.)

0.5 degrees MAE in its angle estimates when accurate measurements are available for solar generators and either measurements or pseudomeasurements are available for loads. Results for a representative run in this scenario are shown in Figure 4.4. More generally, our estimates (averaged across all runs) have at most $10.2 \pm 0.2\%$ voltage magnitude MAPE and 0.50 ± 0.26 degrees voltage angle MAE in *any* scenario where solar generators are measured, and at most $16.0 \pm 3.4\%$ magnitude MAPE and 3.3 ± 1.9 degrees angle MAE in any scenario where solar generators have pseudomeasurements (where angle errors are high in this latter case due to solar generator measurement noise). If solar generator measurements are unknown, magnitude MAPEs range from 57-73%, and angle MAEs range from 2.7-30 degrees.

A potential alternative to using low-observability state estimation techniques is to enable full-observability techniques by deploying additional sensors. To model this alternative, we randomly add AMI sensors (which collect coarse-granularity load data, modeled as pseu-

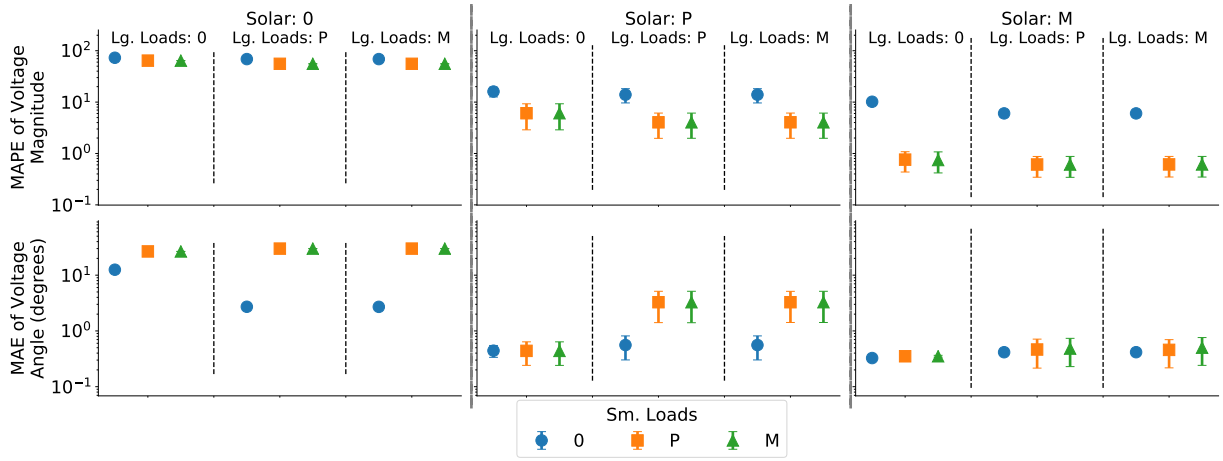


Figure 4.3: Performance on the 33-bus test case with 1% noise over different data availabilities for solar, large load, and small load buses (0 = not measured, P = some pseudomeasurements, M = some measurements; each point represents 50 runs). Our method achieves good estimation error in many scenarios, even though all scenarios shown exhibit low observability.

domeasurements) and "magnitude sensors" (which collect voltage and current magnitudes) to our system until full observability is achieved. We then compare our method to WLS on this augmented system. Voltage phasor estimates for a representative run are shown in Figure 4.5. In this case, both our matrix completion algorithm and WLS quite accurately estimate voltages, with voltage magnitude MAPEs and angle MAEs within 0.6% and 0.53 degrees, respectively, for our matrix completion algorithm and within 0.8% and 0.67 degrees, respectively, for WLS. However, achieving full observability requires adding 28-63 sensors to the system (depending on the baseline data availability scenario), which represents a potentially high cost to the distribution utility.

Overall, our results demonstrate that our matrix completion algorithm can provide accurate state estimation performance in the low-observability case (where traditional state estimation techniques cannot operate), as well as in the full-observability case.

4.4.2 123-bus feeder

We next demonstrate that our matrix completion method effectively scales to larger systems via experiments on the IEEE 123-bus feeder. The 123-bus feeder is a multi-phase unbalanced radial distribution system, in which buses are single-, double-, or three-phase (with 263 phases in total). For our simulations, we add PV systems at each load node of the feeder. We employ a bus-formulation matrix for this test case, where each matrix row represents a phase at one bus. To validate our approach, we use two hours of system voltage and power injection data. This data was simulated at one-minute resolution using power flow analysis with diversified load and solar profiles created for each bus using realistic solar irradiance and load consumption data. The corresponding voltage magnitudes range from 0.95-1 p.u., and voltage angles are around 0 or ± 120 degrees.

As in the previous section, we assume that voltage phasors are known exactly at the

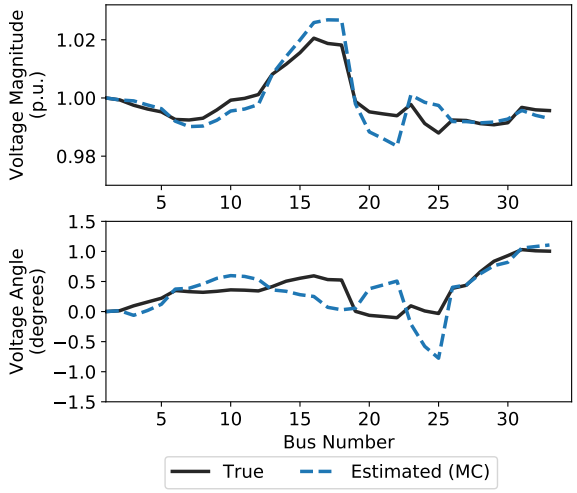


Figure 4.4: Representative voltage phasor estimates for the 33-bus test case with 1% noise in the low-observability scenario with solar, large loads, and small loads partially measured.

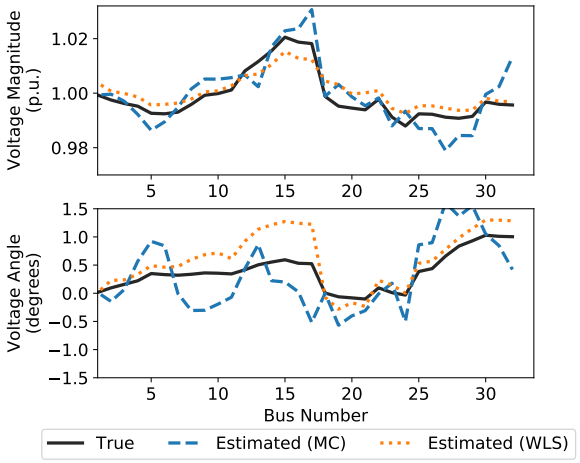


Figure 4.5: Representative estimates for the 33-bus test case with 1% noise and data-driven sampling at full observability.

Table 4.1: Average running time over 50 runs for our matrix completion method ($w_\epsilon = 20$, $\forall \epsilon \in \mathcal{E}$) on the IEEE 123-bus Feeder, using the CXVPY SCS Solver on a laptop with 1.9 GHz CPU and 32 GB RAM.

% "potentially measured" quants. avail.	10	30	50	70	90
Running time for 0.2% noise (s)	20	22	23	41	50
Running time for 1% noise (s)	78	85	88	89	93

slack bus (which here has three phases) and must be estimated everywhere else. All other measurements (i.e., voltage magnitudes and power injections) are known exactly at the slack bus and are "potentially measured" for non-slack system phases. The running time of our implementation for different percentages of known measurements and noise levels on the IEEE 123-bus feeder is shown in Table 4.1; as a note, we found that tuning the parameters w_ϵ did not affect the performance of our method, but did affect the running time.

We first employ our algorithm at one point in the time series during which solar injections are nonzero. In our experiments, we vary the percentage of "potentially measured" quantities that are measured, and note that there are fewer measurements than unknown voltage phasor quantities if less than two-thirds of these quantities are measured (see condition (i) for observability in Section 4.1). Results for the cases of 0.2% and 1% measurement noise are shown in Figure 4.6. We also show representative results for one run under 1% measurement noise and 50% measurement availability (which is in the low-observability realm) in Figure 4.7.

These results show that our algorithm estimates voltage phasors with relatively high accuracy across all levels of data availability. The MAPE of our voltage magnitude estimates is less than 2.6% even when no "potentially measured" quantities are measured, and falls to less than 1% once 10% or more of these quantities are available. Unsurprisingly, our voltage

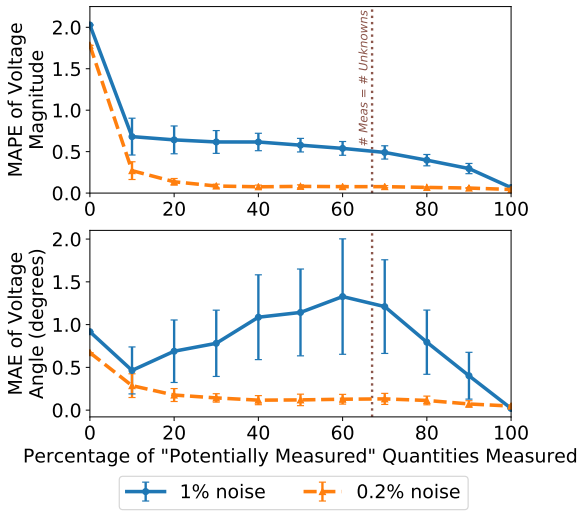


Figure 4.6: Performance on the 123-bus test case for one time step. (Each point represents 50 runs.)

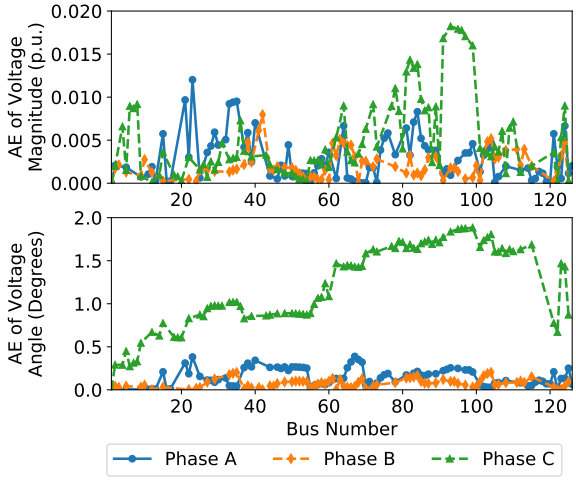


Figure 4.7: Performance on the 123-bus test case for a representative run with 50% data availability and 1% noise.

magnitude estimates are better when measurement noise is lower. We do note, however, that the MAPE is not equal to zero even when 100% of measurements are available, since we use approximate linear power flow equations as constraints in our formulation (4.13).

For voltage angle estimates, we again report MAEs rather than MAPEs since the angles at some phases are close to zero. For 1% measurement noise, we see that the MAE is at most 1.5 degrees across all data availability levels, which is small given that most voltage angles are around ±120 degrees. However, the accuracy of our angle estimates does not decrease monotonically as more measurements are added. This is because our algorithm directly estimates the real and imaginary parts of voltage phasors; while we estimate these quantities accurately, their errors are not correlated (especially under large measurement noise), leading to inaccuracies in their implied angle estimates. At 0.2% measurement noise, the MAE for voltage angle decreases as more measurements are available, dropping below 1 degree even when no measurements are available.

To demonstrate the scalability of our method, we next implement our matrix completion algorithm on the entire two-hour time series. We model data availability by placing sensors at fixed sets of buses comprising 30%, 50%, or 70% of all buses (with larger sets of buses inclusive of smaller sets). The MAPEs of our voltage magnitude estimates under 1% measurement noise are shown in Figure 4.8. We find that under 30% data availability, the MAPEs of these estimates are within 1% for 80% of time steps, and within 1.5% for 99% of time steps, with a maximum MAPE of 2.7%. Under 50% and 70% data availability, the MAPEs of our voltage magnitude estimates are within 1% across all points in the time series. The MAEs of our voltage angle estimates at each time step (not pictured) are similar to those in Figure 4.6.

Finally, we demonstrate the robustness of our algorithm to dynamic measurement loss, which commonly occurs on the distribution system due to failures such as packet drops and equipment malfunctions. We model the baseline data availability as being 50% (low-observability), and randomly remove 20% of the available measurements at each point

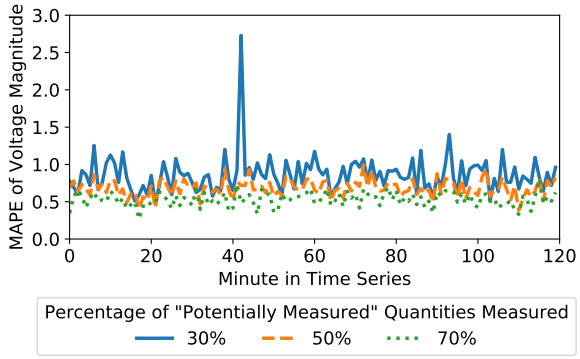


Figure 4.8: Time-series performance for the 123-bus test case with 1% noise. Voltage magnitude MAPEs are at most 2.7% with 30% of data, and at most 1% with 50% or 70% of data.

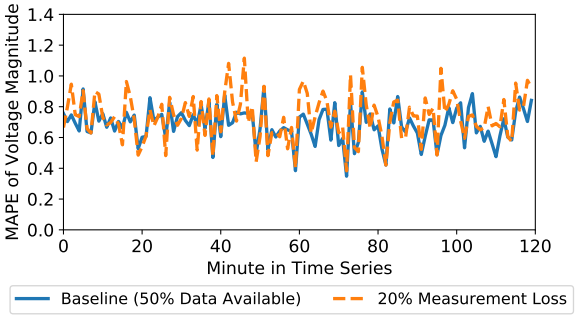


Figure 4.9: Time-series performance for the 123-bus test case with 1% noise when measurements are randomly lost. Our magnitude estimates are relatively robust to data loss.

in the two-hour time series. The MAPEs of our voltage magnitude estimates under 1% measurement noise are shown in Figure 4.9. We find that in most cases, our estimates under data loss are similar to the estimates without data loss, with larger deviations in some cases when critical measurements are lost. In all cases, the MAPEs of our voltage magnitude estimates are less than 1.2%, demonstrating the robustness of our algorithm to real-world operating conditions.

4.5 Conclusion

We present an algorithm for low-observability distribution system voltage estimation based on constrained low-rank matrix completion. This method can accurately estimate voltage phasors under low-observability conditions where standard state estimation methods cannot operate, and can flexibly accommodate any distribution network measurements available in the field. Our empirical evaluations of this method demonstrate that it produces accurate and robust voltage phasor estimates on the IEEE 33- and 123-bus test systems under a variety of data availability conditions. As such, we believe that our algorithm is a useful mechanism for voltage estimation on modern distribution systems.

Part II

Optimization-in-the-Loop Deep Learning

Chapter 5

Decision-Cognizant Learning for Stochastic Optimization

With the increasing popularity of machine learning techniques, it has become common to see prediction algorithms operating within some larger decision-making process. However, the criteria by which we train these algorithms (e.g., mean squared error of a prediction) often differ from the ultimate criteria on which we evaluate them (e.g., the quality of the decision made on the basis of the prediction). In this chapter, we propose an end-to-end approach for training probabilistic machine learning models in a manner that directly captures the objective of the ultimate decision-making process in which they will be used. Specifically, we propose to train a probabilistic model not (solely) for predictive accuracy, but so that – when it is later used within the loop of a stochastic programming procedure – it produces solutions that minimize the ultimate decision-relevant (task-based) loss. We present three experimental evaluations of the proposed approach: a classical inventory stock problem, a real-world electrical grid scheduling task, and a real-world energy storage arbitrage task. We show that the proposed approach can outperform both traditional modeling and purely black-box policy optimization approaches in these applications.

The work in this chapter has previously been published in:[1]

> Priya L. Donti, Brandon Amos, and J. Zico Kolter. "Task-based End-to-End Model Learning in Stochastic Optimization." *Advances in Neural Information Processing Systems*. 2017, 5490–5500.

[1] Code for all experiments is available at: https://github.com/locuslab/e2e-model-learning.

5.1 Introduction

Predictive algorithms commonly operate within some larger decision-making process. However, the criteria by which we generally train these algorithms (often related to predictive accuracy) tend to differ from the ultimate decision-related criteria on which we *actually* evaluate them (the performance of the full "closed-loop" system on the ultimate task at hand). For instance, instead of merely classifying images in a standalone setting, one may want to use these classifications within planning and control tasks such as autonomous driving. While a typical image classification algorithm might optimize accuracy or log likelihood, in a driving task, we may ultimately care more about the difference between classifying a pedestrian as a tree vs. classifying a garbage can as a tree. Similarly, when we use a probabilistic prediction algorithm to generate forecasts of upcoming electricity demand, we then want to use these forecasts to minimize the costs of a scheduling procedure that allocates power generation on the grid. As these examples suggest, instead of using a "generic" loss, we instead may want to learn a model that approximates the ultimate decision-relevant (or "task-based") loss.

This chapter considers an end-to-end approach for training probabilistic machine learning models that directly capture the objective of their ultimate task. Formally, we consider probabilistic models in the context of stochastic programming, where the goal is to minimize some expected cost over a model's probabilistic predictions, subject to some (potentially also probabilistic) constraints. In general, it is common to approach these problems in a two-step fashion: first to fit a predictive model to observed data by minimizing some criterion such as negative log-likelihood, and then to solve (or approximate) the stochastic programming problem using this model's predictions. While this procedure can work well in many instances, it neglects the fact that the *true* cost of the end-to-end system (the stochastic optimization objective evaluated on actual instantiations of uncertainty in the real world) may benefit from a model that actually attains worse overall likelihood, but makes more accurate predictions over certain manifolds of the underlying space.

In this work, we therefore propose to train a probabilistic model not (solely) for predictive accuracy, but so that – when it is later used within the loop of a stochastic programming procedure – it produces solutions that minimize the ultimate task-based loss. This formulation may seem somewhat counterintuitive, given that a "perfect" predictive model would of course also be an optimal model to use within a stochastic programming framework. However, the reality that all models *do* make errors illustrates that we should indeed look to a final task-based objective to determine the proper error tradeoffs within a machine learning setting. This chapter proposes one way to evaluate task-based tradeoffs in a fully automated fashion, by computing derivatives through the solution to the stochastic programming problem in a manner that can improve the underlying model.

In the rest of this chapter, we describe our task-based end-to-end learning approach within the context of stochastic programming, and give a generic method for propagating task loss through these stochastic programming problems in a manner that can update the underlying predictive models. We provide three experimental evaluations of the proposed approach: a classical inventory stock problem, a real-world electrical grid scheduling task, and a real-world energy storage arbitrage task. We show that the proposed approach outperforms traditional modeling and purely black-box policy optimization approaches.

5.2 Related work

Stochastic programming. Stochastic programming is a method for making decisions under uncertainty by modeling or optimizing objectives governed by a random process. It has applications in many domains such as energy [WF03], finance [ZV06], and manufacturing [BS93], where the underlying probability distributions are either known or can be estimated. Common considerations include how to best model or approximate the underlying random variables, how to solve the resulting optimization problem, and how to then assess the quality of the resulting (approximate) solution [SP07].

In cases where the underlying probability distribution is known but the objective cannot be solved analytically, it is common to use Monte Carlo sample average approximation methods, which draw multiple iid samples from the underlying probability distribution and then use deterministic optimization methods to solve the resultant problems [LSW06]. In cases where the underlying distribution is not known, it is common to learn or estimate some model from observed samples [RW91].

End-to-end training Recent years have seen a dramatic increase in the number of systems building on so-called "end-to-end" learning. Generally speaking, this term refers to systems where the end goal of the machine learning process is directly predicted from raw inputs [e.g. LeC+05; Tho+06]. In the context of deep learning systems, the term now traditionally refers to architectures where, for example, there is no explicit encoding of hand-tuned features on the data, but the system directly predicts what the image, text, etc. is from the raw inputs [WBB11; He+16; Wan+12; GJ14; Amo+15]. The context in which we use the term end-to-end is similar, but slightly more in line with its older usage: instead of (just) attempting to learn a predictive output to optimize some standalone notion of accuracy (using known and typically straightforward loss functions), we are specifically attempting to learn a predictive model based upon an end-to-end *task* that the user is ultimately trying to accomplish. We feel that this concept – of describing the entire "closed-loop" performance of the system as evaluated on the real task at hand – is beneficial to add to the notion of end-to-end learning.

Also highly related to our work are recent efforts in end-to-end policy learning [Lev+16], in embedding value iteration procedures within neural networks [Tam+16], and in multi-objective optimization [HSK06; VMN14; Mos+16; WWD14]. These lines of work fit more with the "pure" model-free end-to-end approach we discuss in Section 5.3.1 (where predictive models are eschewed for pure function approximation methods), but conceptually the approaches have similar motivations in modifying typically-optimized policies to address some task(s) directly.

Optimizing alternative loss functions There has been a great deal of work in recent years on using machine learning procedures to optimize different loss criteria than those "naturally" optimized by the algorithm. For example, Stoyanov, Ropson, and Eisner [SRE11] and Hazan, Keshet, and McAllester [HKM10] propose methods for optimizing loss criteria in structured prediction that are *different* from the inference procedure of the prediction algorithm; this work has also recently been extended to deep networks [Son+16]. Recent work has also explored using auxiliary prediction losses to satisfy multiple objectives [Jad+17],

learning dynamics models that maximize control performance in Bayesian optimization [Ban+17], and learning adaptive predictive models via differentiation through a meta-learning optimization objective [FAL17].

The prior work we have found that most closely resembles our approach is Bengio [Ben97], which uses a neural network model for predicting financial prices, and then optimizes the model based on returns obtained via a hedging strategy that employs it. We view this approach – of both using a model and then tuning that model to adapt to a (differentiable) procedure – as a philosophical predecessor to our own work. In particular, our approach likewise tunes a model to adapt to a (differentiable) decision-making procedure, though we consider the more general case where the decision-making process can be captured by a general stochastic optimization problem, as opposed to needing to hand-craft a differentiable decision-making procedure. (Despite the work of Bengio [Ben97] being over 20 years old, virtually all direct follow-on work has focused on the financial application, and not on what we feel is the core idea of learning a predictive model within a task-driven optimization procedure.) In concurrent work, Elmachtoub and Grigas [EG22] also propose an approach for tuning model parameters within linear regression given the results of downstream decisions, by aiming to minimize a *regret*-related loss function; however, their work only applies to the context of linear programming (as opposed to more general optimization problems). Since the work in this chapter was first published, it has also inspired a number of extensions, including work in the area of combinatorial optimization [WDT18], as well as deployed applications in inventory optimization and power scheduling by a number of industry players.

5.3 End-to-end model learning in stochastic programming

We first formally define the stochastic modeling and optimization problems with which we are concerned. Let $(x \in \mathcal{X}, y \in \mathcal{Y}) \sim \mathcal{D}$ denote standard input-output pairs drawn from some (real, unknown) distribution $\mathcal{D}$. We also consider actions $z \in \mathcal{Z}$ that incur some expected loss $L_{\mathcal{D}}(z) = E_{x,y\sim\mathcal{D}}[f(x, y, z)]$. For instance, a power systems operator may try to allocate power generation z given past electricity demand x and future electricity demand y; this allocation's loss corresponds to the over- or under-generation penalties incurred given future demand instantiations.

If we knew $\mathcal{D}$, then we could select optimal actions $z^\star_{\mathcal{D}} = \text{argmin}_z\, L_{\mathcal{D}}(z)$. However, in practice, the true distribution $\mathcal{D}$ is unknown. In this work, we are thus interested in modeling the conditional distribution $y|x$ using some parameterized model $p(y|x;\theta)$, with the goal of minimizing the real-world cost of the policy implied by our parameterization. Specifically, we find some parameters θ to parameterize $p(y|x;\theta)$ (as in the standard statistical setting) and then determine the optimal actions $z^\star(x;\theta)$ under our observed input x and the specific choice of parameters θ in our probabilistic model (via stochastic optimization). Upon observing the costs of these actions $z^\star(x;\theta)$ relative to *true* instantiations of x and y under $\mathcal{D}$, we update our parameterized model $p(y|x;\theta)$ accordingly, calculate the resultant new $z^\star(x;\theta)$, and repeat. The goal is to find parameters θ such that the corresponding policy $z^\star(x;\theta)$ optimizes the loss under the *true* joint distribution of x and y.

Explicitly, we wish to choose θ to minimize the *task loss* $L(\theta)$ in the context of $x, y \sim \mathcal{D}$,

i.e.,

$$\underset{\theta}{\text{minimize}} \quad L(\theta) \coloneqq \mathbf{E}_{x,y\sim\mathcal{D}}[f(x, y, z^\star(x;\theta))]. \tag{5.1}$$

Since in reality we do not know the distribution $\mathcal{D}$ and the true associated optimal decision $z^\star_{\mathcal{D}}$, this loss employs the optimal decision $z^\star(x;\theta)$ under $p(y|x;\theta)$ (for a fixed instantiation of parameters θ estimated by our model), computed via the proxy stochastic optimization problem

$$z^\star(x;\theta) \coloneqq \underset{z}{\text{argmin}} \quad \mathbf{E}_{y\sim p(y|x;\theta)}[f(x, y, z)]. \tag{5.2}$$

The above setting specifies $z^\star(x;\theta)$ using a simple (unconstrained) stochastic program, but in reality our decision may be subject to both probabilistic and deterministic constraints. We therefore consider more general decisions produced through a generic stochastic programming problem[2]

$$\begin{aligned} z^\star(x;\theta) \coloneqq \underset{z}{\text{argmin}} \quad & \mathbf{E}_{y\sim p(y|x;\theta)}[f(x, y, z)] \\ \text{subject to} \quad & \mathbf{E}_{y\sim p(y|x;\theta)}[g_i(x, y, z)] \leq 0, \quad i = 1, \ldots, n_{ineq} \\ & h_i(z) = 0, \quad i = 1, \ldots, n_{eq}. \end{aligned} \tag{5.3}$$

In this setting, the full task loss is more complex, since it captures both the expected cost and any deviations from the constraints. We can write this, for instance, as

$$L(\theta) = \mathbf{E}_{x,y\sim\mathcal{D}}[f(x, y, z^\star(x;\theta))] + \sum_{i=1}^{n_{ineq}} I\{\mathbf{E}_{x,y\sim\mathcal{D}}[g_i(x, y, z^\star(x;\theta))] \leq 0\} + \sum_{i=1}^{n_{eq}} \mathbf{E}_x[I\{h_i(z^\star(x;\theta)) = 0\}] \tag{5.4}$$

(where $I(\cdot)$ is the indicator function that is zero when its constraints are satisfied and infinite otherwise). However, the basic intuition behind our approach remains the same for both the constrained and unconstrained cases: in both settings, we attempt to learn parameters of a probabilistic model not to produce strictly "accurate" predictions, but such that *when we use the resultant model within a stochastic programming setting, the resulting decisions perform well under the true distribution.*

Actually solving this problem requires that we differentiate through the "argmin" operator $z^\star(x;\theta)$ of the stochastic programming problem. This differentiation is not possible for all classes of optimization problems (the argmin operator may be discontinuous), but as we will show shortly, in many practical cases – including cases where the function and constraints are strongly convex – we can indeed efficiently compute these gradients even in the context of constrained optimization.

5.3.1 Discussion and alternative approaches

We highlight our approach in contrast to two alternative existing methods: traditional model learning and model-free black-box policy optimization. In traditional machine learning

[2]It is standard to presume in stochastic programming that equality constraints depend only on decision variables (not random variables), as non-trivial random equality constraints are typically not possible to satisfy.

Algorithm 1 Task Loss Optimization

input: $\mathcal{D}$ *// samples from true distribution*
initialize θ *// some initial parameterization*
for $t = 1, \ldots, T$ **do**
  sample $(x, y) \sim \mathcal{D}$
  compute $z^\star(x; \theta)$ via Equation (5.3)
  // step in violated constraint or objective
  if $\exists i$ s.t. $g_i(x, y, z^\star(x; \theta)) > 0$ **then**
    update θ with $\nabla_\theta g_i(x, y, z^\star(x; \theta))$
  else
    update θ with $\nabla_\theta f(x, y, z^\star(x; \theta))$
  end if
end for

approaches, it is common to use θ to minimize the (conditional) log-likelihood of observed data under the model $p(y|x; \theta)$. This method corresponds to approximately solving the optimization problem

$$\underset{\theta}{\text{minimize}} \quad \mathbf{E}_{x,y\sim\mathcal{D}} \left[-\log p(y|x; \theta) \right]. \tag{5.5}$$

If we then need to use the conditional distribution $y|x$ to determine actions z within some later optimization setting, we commonly use the predictive model obtained from (5.5) directly. This approach has obvious advantages, in that the model-learning phase is well-justified independent of any future use in a task. However, it is also prone to poor performance in the common setting where the true distribution $y|x$ cannot be represented within the class of distributions parameterized by θ, i.e., where the procedure suffers from model bias. Conceptually, the log-likelihood objective *implicitly* trades off between model error in different regions of the input/output space, but does so in a manner largely opaque to the modeler, and may ultimately *not* employ the correct tradeoffs for a given task.

In contrast, there is an alternative approach to solving (5.1) that we describe as the model-free "black-box" policy optimization approach. Here, we would forgo learning any model at all of the random variable y. Instead, we would attempt to learn a policy mapping directly from inputs x to actions $z^\star(x; \tilde{\theta})$ that minimize the loss $L(\tilde{\theta})$ presented in (5.4) (where here $\tilde{\theta}$ defines the form of the policy itself, not a parameterized predictive model). While such model-free methods can perform well in many settings, they are often very data-inefficient, as the policy class must have enough representational power to describe sufficiently complex policies without recourse to any underlying model.[3]

Our approach offers an intermediate setting, where we *do* still use an intermediate model to determine an optimal decision $z^\star(x; \theta)$, yet we adapt this model based on the task loss instead of model prediction accuracy. (In practice, we often want to minimize some

[3]This distinction is roughly analogous to the policy search vs. model-based settings in reinforcement learning. However, for the purposes of this work, we consider much simpler stochastic programs without the multiple rounds that occur in RL; extension of these techniques to a full RL setting remains as future work.

weighted combination of log-likelihood *and* task loss, which can easily be accomplished.)

5.3.2 Optimizing task loss

To solve the generic optimization problem (5.1)/(5.4), we can in principle adopt a straightforward (constrained) stochastic gradient approach, as detailed in Algorithm 1. At each iteration, we solve the proxy stochastic programming problem (5.3) to obtain $z^\star(x, \theta)$, using the distribution defined by our current values of θ. Then, we compute the true loss $L(\theta)$ using the observed value of y. If any of the inequality constraints g_i in $L(\theta)$ are violated, we take a gradient step in the violated constraint; otherwise, we take a gradient step in the optimization objective f. We note that if any inequality constraints are probabilistic, Algorithm 1 must be adapted to employ mini-batches in order to determine whether these probabilistic constraints are satisfied. Alternatively, it is common in practice to simply move a weighted version of these constraints to the objective, i.e., we modify the objective by adding some appropriate penalty times the positive part of the function, $\lambda g_i(x, y, z)_+$, for some $\lambda > 0$. In practice, this has the effect of taking gradient steps jointly in all the violated constraints and the objective in the case that one or more inequality constraints are violated, often resulting in faster convergence. Note that we need only move stochastic constraints into the objective; deterministic constraints on the policy itself will always be satisfied by the optimizer, as they are independent of the model.

5.3.3 Differentiating the stochastic optimization solution

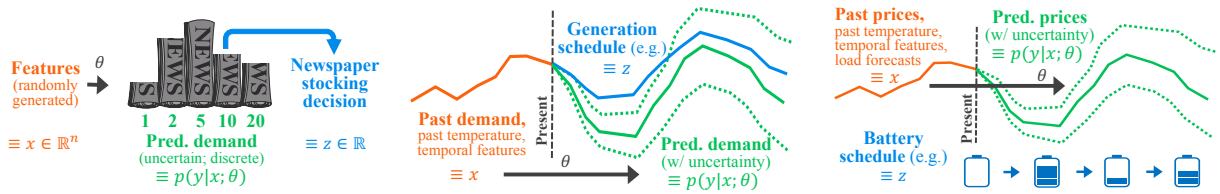
While the above presentation highlights the simplicity of the proposed approach, it avoids the issue of chief technical challenge, which is computing the gradient of an objective that depends upon the argmin operation $z^\star(x; \theta)$. Specifically, we need to compute the term

$$\frac{\mathrm{d}L}{\mathrm{d}\theta} = \frac{\partial L}{\partial \theta} + \frac{\partial L}{\partial z^\star}\frac{\mathrm{d}z^\star}{\mathrm{d}\theta} \tag{5.6}$$

which involves the Jacobian $\mathrm{d}z^\star/\mathrm{d}\theta$. This is the Jacobian of the optimal solution with respect to the distribution parameters θ. Previous approaches have looked into similar argmin differentiations [Gou+16; AXK17] (see also Section 2.2.3), though the methodology we present here is more general and handles the stochasticity of the objective.

To obtain this Jacobian, we begin by writing the KKT optimality conditions of the general stochastic programming problem (5.3), where all expectations are taken with respect to the modeled distribution $y \sim p(y|x; \theta)$ (for compactness, denoted here as $\mathbf{E}_{y_\theta}$). Further, assuming the problem is convex, we can replace the general equality constraints $h(z) = 0$ with the linear constraint $Az = b$. A point (z, λ, ν) is a primal-dual optimal point if it satisfies

$$\begin{aligned} \mathbf{E}_{y_\theta} g(z) &\leq 0 \\ Az &= b \\ \lambda &\geq 0 \\ \lambda \circ \mathbf{E}_{y_\theta} g(z) &= 0 \\ \nabla_z \mathbf{E}_{y_\theta} f(z) + \lambda^T \nabla_z \mathbf{E}_{y_\theta} g(z) + A^T \nu &= 0, \end{aligned} \tag{5.7}$$



(a) Inventory stock problem (b) Load forecasting problem (c) Price forecasting problem

Figure 5.1: Features x, model predictions y, and policy z for the three experiments.

where g denotes the vector of all inequality constraints, and where we wrap the dependence on x and y into the functions f and g_i themselves.

Differentiating these equations and applying the implicit function theorem gives a set of linear equations that we can solve to obtain the necessary Jacobians:

$$\begin{bmatrix} \nabla_z^2 \mathbf{E}_{y_\theta} f(z) + \sum_{i=1}^{n_{ineq}} \lambda_i \nabla_z^2 \mathbf{E}_{y_\theta} g_i(z) & (\nabla_z \mathbf{E}_{y_\theta} g(z))^T & A^T \\ \operatorname{diag}(\lambda)(\nabla_z \mathbf{E}_{y_\theta} g(z)) & \operatorname{diag}(\mathbf{E}_{y_\theta} g(z)) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}z}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\lambda}{\mathrm{d}\theta} \\ \frac{\mathrm{d}\nu}{\mathrm{d}\theta} \end{bmatrix} = - \begin{bmatrix} \frac{\mathrm{d}\nabla_z \mathbf{E}_{y_\theta} f(z)}{\mathrm{d}\theta} + \frac{\mathrm{d}\sum_{i=1}^{n_{ineq}} \lambda_i \nabla_z \mathbf{E}_{y_\theta} g_i(z)}{\mathrm{d}\theta} \\ \operatorname{diag}(\lambda) \frac{\mathrm{d}\mathbf{E}_{y_\theta} g(z)}{\mathrm{d}\theta} \\ 0 \end{bmatrix}. \tag{5.8}$$

As described in Section 2.2.3, we rarely solve for the Jacobians explicitly, but instead use the Jacobian-vector trick to directly solve for the gradient $\mathrm{d}L/\mathrm{d}\theta$. The above equations will take slightly different forms depending on how the stochastic programming problem is solved, but are usually fairly straightforward to compute if the solution is solved in some "exact" manner (i.e., where second order information is used). In practice for the experiments in this chapter, we calculate the right-hand terms by employing sequential quadratic programming [BT95] to find the optimal policy $z^\star(x;\theta)$ for the given parameters θ, using an existing approach for fast solution of argmin differentiation in QPs [AK17] to solve the necessary linear equations; we then take the derivatives at the optimum produced by this strategy.

5.4 Experiments

We consider three applications of our approach: a synthetic inventory stock problem, a real-world energy scheduling task, and a real-world battery arbitrage task. We demonstrate that our task-based end-to-end approach can substantially improve upon alternatives.

Implementation notes For all linear models, we use a one-layer linear neural network with the appropriate input and output layer dimensions. For all nonlinear models, we use a two-hidden-layer neural network, where each "layer" is actually a combination of linear, batch norm [IS15], ReLU, and dropout ($p = 0.2$) layers with dimension 200. In both cases, we add an additional softmax layer in cases where probability distributions are being predicted.

All models are implemented using PyTorch [Pas+19] and employ the Adam optimizer [KB15]. All QPs are solved using a recently-developed differentiable batch QP solver [AK17], and Jacobians are also computed automatically using backpropagation via the same.

5.4.1 Inventory stock problem

Problem definition To highlight the performance of the algorithm in a setting where the true underlying model is known to us, we consider a "conditional" variation of the classical inventory stock problem [SP07]. In this problem, a company must order some quantity z of a product to minimize costs over some stochastic demand y, whose distribution in turn is affected by some observed features x (Figure 5.1a). There are linear and quadratic costs on the amount of product ordered, plus different linear/quadratic costs on over-orders $[z-y]_+$ and under-orders $[y-z]_+$. The objective is given by

$$f_{stock}(y,z) = c_0 z + \frac{1}{2}q_0 z^2 + c_b[y-z]_+ + \frac{1}{2}q_b([y-z]_+)^2 + c_h[z-y]_+ + \frac{1}{2}q_h([z-y]_+)^2, \quad (5.9)$$

where $[v]_+ \equiv \max\{v, 0\}$, and $(c_0, q_0), (c_b, q_b)$, and (c_h, q_h) are linear and quadratic costs on the amount of product ordered, over-orders, and under-orders, respectively. For a specific choice of probability model $p(y|x;\theta)$, our proxy stochastic programming problem can then be written as

$$\underset{z}{\text{minimize}} \quad L(\theta) \coloneqq \mathbf{E}_{y\sim p(y|x;\theta)}[f_{stock}(y,z)]. \quad (5.10)$$

To simplify the setting, we further assume that the demands are discrete, taking on values $d_1, \ldots, d_k$ with probabilities (conditional on x) $(p_\theta)_i \equiv p(y = d_i|x;\theta)$. Thus our stochastic programming problem (5.10) can be written succinctly as a joint quadratic program[4]

$$\begin{aligned} \underset{z\in\mathbb{R}, z_b, z_h \in \mathbb{R}^k}{\text{minimize}} \quad & c_0 z + \frac{1}{2}q_0 z^2 + \sum_{i=1}^{k}(p_\theta)_i \left(c_b (z_b)_i + \frac{1}{2}q_b (z_b)_i^2 + c_h (z_h)_i + \frac{1}{2}q_h (z_h)_i^2 \right) \\ \text{subject to} \quad & d - z\mathbf{1} \le z_b, \quad z\mathbf{1} - d \le z_h, \quad z, z_h, z_b \ge 0. \end{aligned} \quad (5.11)$$

To demonstrate the explicit formula for argmin operation Jacobians for this particular case (e.g., to compute the terms in (5.8)), note that we can write the above QP in inequality form as $\text{minimize}_{\{\boldsymbol{z}: G\boldsymbol{z}\le h\}} \frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + c^T \boldsymbol{z}$ with

$$\boldsymbol{z} = \begin{bmatrix} z \\ z_b \\ z_h \end{bmatrix}, \; Q = \begin{bmatrix} q_0 & 0 & 0 \\ 0 & q_b p_\theta & 0 \\ 0 & 0 & q_h p_\theta \end{bmatrix}, \; c = \begin{bmatrix} c_0 \\ c_b p_\theta \\ c_h p_\theta \end{bmatrix}, \; G = \begin{bmatrix} -1 & -I & 0 \\ 1 & 0 & -I \\ -1 & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & -I \end{bmatrix}, \; h = \begin{bmatrix} -d \\ d \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (5.12)$$

Thus, for an optimal primal-dual solution $(\boldsymbol{z}^\star, \lambda^\star)$, we can compute the Jacobian $\mathrm{d}\boldsymbol{z}^\star/\mathrm{d}p_\theta$ (the Jacobian of the optimal solution with respect to the probability vector p_θ mentioned above), via the formula

$$\begin{bmatrix} \frac{\mathrm{d}\boldsymbol{z}^\star}{\mathrm{d}p_\theta} \\ \frac{\mathrm{d}\lambda^\star}{\mathrm{d}p_\theta} \end{bmatrix} = \begin{bmatrix} Q & G^T \\ \mathrm{diag}(\lambda^\star)G & \mathrm{diag}(G\boldsymbol{z}^\star - h) \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ q_b z_b^\star + c_b \mathbf{1} \\ q_h z_h^\star + c_h \mathbf{1} \\ 0 \end{bmatrix}, \quad (5.13)$$

[4]This is referred to as a two-stage stochastic programming problem (though a very trivial example of one), where first stage variables consist of the amount of product to buy before observing demand, and second-stage variables consist of how much to sell back or additionally purchase once the true demand has been revealed.

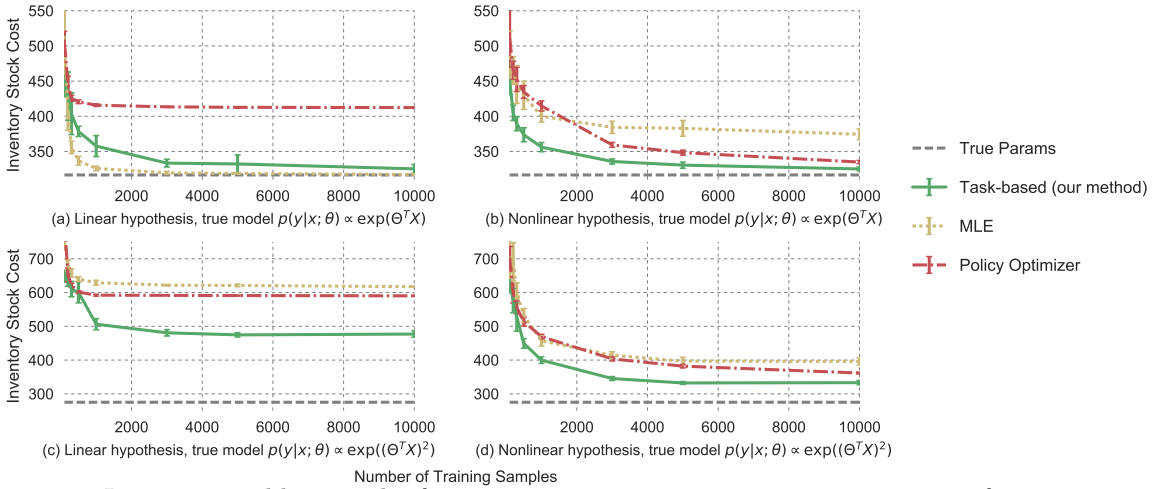


Figure 5.2: Inventory problem results for 10 runs over a representative instantiation of true parameters ($c_0 = 10, q_0 = 2, c_b = 30, q_b = 14, c_h = 10, q_h = 2$). Cost is evaluated over 1000 testing samples (lower is better). The linear MLE performs best for a true linear model. In all other cases, the task-based models outperform their MLE and policy counterparts.

where $\mathrm{diag}(\cdot)$ denotes a diagonal matrix for an input vector. After solving the problem and computing these Jacobians, we can compute the overall gradient with respect to the task loss $L(\theta)$ via the chain rule

$$\frac{\mathrm{d}L}{\mathrm{d}\theta} = \frac{\partial L}{\partial \theta} + \frac{\partial L}{\partial \boldsymbol{z}^\star} \frac{\mathrm{d}\boldsymbol{z}^\star}{\mathrm{d}p_\theta} \frac{\mathrm{d}p_\theta}{\mathrm{d}\theta} \tag{5.14}$$

where $\mathrm{d}p_\theta / \mathrm{d}\theta$ denotes the Jacobian of the model probabilities with respect to its parameters, which are computed in the typical manner.

Experimental setup We examine our algorithm under two main conditions: where the true model is linear, and where it is nonlinear. In all cases, we generate problem instances by randomly sampling some $x \in \mathbb{R}^n$ and then generating $p(y|x;\theta)$ according to either $p(y|x;\theta) \propto \exp(\Theta^T x)$ (linear true model) or $p(y|x;\theta) \propto \exp((\Theta^T x)^2)$ (nonlinear true model) for some $\Theta \in \mathbb{R}^{n \times k}$. We compare the following approaches on these tasks: 1) the QP allocation based upon the true model (which performs optimally); 2) MLE approaches (with linear or nonlinear probability models) that fit a model to the data, and then compute the allocation by solving the QP; 3) pure end-to-end policy-optimizing models (using linear or nonlinear hypotheses for the policy); and 4) our task-based learning models (with linear or nonlinear probability models).[5] In all cases, we evaluate test performance by running on 1000 random examples, and evaluate performance over 10 folds of different true $\theta^\star$ parameters.

Figures 5.2(a) and (b) show the performance of these methods given a linear true model, with linear and nonlinear model hypotheses, respectively. As expected, the linear MLE approach performs best, as the true underlying model is in the class of distributions that it can represent and thus solving the stochastic programming problem is a very strong proxy

[5]Here, our method's the loss function for the inventory stock problem involves a weighted combination of the task loss *and* a negative log-likelihood regularization term.

for solving the true optimization problem under the real distribution. While the true model is also contained within the nonlinear MLE's generic nonlinear distribution class, we see that this method requires more data to converge, and when given less data makes error tradeoffs that are ultimately not the correct tradeoffs for the task at hand; our task-based approach thus outperforms this approach. The task-based approach also substantially outperforms the policy-optimizing neural network, highlighting the fact that it is more data-efficient to run the learning process "through" a reasonable model. Note that here it does not make a difference whether we use the linear or nonlinear model in the task-based approach.

Figures 5.2(c) and (d) show performance in the case of a nonlinear true model, with linear and nonlinear model hypotheses, respectively. Case (c) represents the "non-realizable" case, where the true underlying distribution cannot be represented by the model hypothesis class. Here, the linear MLE, as expected, performs very poorly: it cannot capture the true underlying distribution, and thus the resultant stochastic programming solution would not be expected to perform well. The linear policy model similarly performs poorly. Importantly, the task-based approach with the *linear* model performs much better than these approaches: despite the fact that it still has a misspecified model, the task-based nature of the learning process lets us learn a *different* linear model than the MLE version, which is particularly tuned to the distribution and loss of the task. Finally, also as to be expected, the nonlinear models perform better than the linear models in this scenario, but again with the task-based nonlinear model outperforming the nonlinear MLE and end-to-end policy approaches.

5.4.2 Load forecasting and generator scheduling

We next consider a more realistic grid-scheduling task, based upon over 8 years of real electrical grid data. In this setting, a power system operator must decide how much electricity generation $z \in \mathbb{R}^{24}$ to schedule for each hour in the next 24 hours based on some (unknown) distribution over electricity demand (Figure 5.1b). Given a particular realization y of demand, we impose penalties for both generation excess (γ_e) and generation shortage (γ_s), with $\gamma_s \gg \gamma_e$. We also add a quadratic regularization term, indicating a preference for generation schedules that closely match demand realizations. Finally, we impose a ramping constraint c_r restricting the change in generation between consecutive timepoints, reflecting physical limitations associated with quick changes in electricity output levels. These are reasonable proxies for the actual economic costs incurred by electrical grid operators when scheduling generation (under a multi-time-step economic dispatch framework; see Section 2.3.2), and can be written as the stochastic programming problem

$$\begin{aligned} \underset{z \in \mathbb{R}^{24}}{\text{minimize}} \quad & \sum_{i=1}^{24} \mathbf{E}_{y \sim p(y|x;\theta)} \left[\gamma_s [y_i - z_i]_+ + \gamma_e [z_i - y_i]_+ + \frac{1}{2}(z_i - y_i)^2 \right] \\ \text{subject to} \quad & |z_i - z_{i-1}| \leq c_r \ \forall i, \end{aligned} \tag{5.15}$$

where $[v]_+ \equiv \max\{v, 0\}$. Assuming (as we will in our model), that y_i is a Gaussian random variable with mean μ_i and variance σ_i^2, then this expectation has a closed form that can be

computed via analytically integrating the Gaussian PDF.[6] Specifically, this closed form is

$$\begin{aligned}&\mathbf{E}_{y\sim p(y|x;\theta)}\left[\gamma_s[y_i - z_i]_+ + \gamma_e[z_i - y_i]_+ + \frac{1}{2}(z_i - y_i)^2\right] \\ &\quad = \underbrace{(\gamma_s + \gamma_e)(\sigma^2 p(z_i;\mu,\sigma^2) + (z_i - \mu)F(z_i;\mu,\sigma^2)) - \gamma_s(z_i - \mu)}_{\alpha(z_i)} + \frac{1}{2}((z_i - \mu_i)^2 + \sigma_i^2),\end{aligned} \tag{5.16}$$

where $p(z;\mu,\sigma^2)$ and $F(z;\mu,\sigma^2)$ denote the Gaussian PDF and CDF, respectively with the given mean and variance. This is a convex function of z (not apparent in this form, but readily established because it is an expectation of a convex function), and we can thus optimize it efficiently and compute the necessary Jacobians.

We use sequential quadratic programming (SQP) to iteratively approximate the resultant convex objective as a quadratic objective, and iterate until convergence. Specifically, we repeatedly solve

$$\begin{aligned}z^{(k+1)} = \underset{z}{\text{argmin}} \;\; &\frac{1}{2}z^T \text{diag}\left(\frac{\mathrm{d}^2\alpha(z_i^{(k)})}{\mathrm{d}z^2} + 1\right) z + \left(\frac{\mathrm{d}\alpha(z^{(k)})}{\mathrm{d}z} - \mu\right)^T z \\ \text{subject to} \;\; &|z_i - z_{i-1}| \leq c_r \;\forall i\end{aligned} \tag{5.17}$$

until $||z^{(k+1)} - z^{(k)}||_2 < \delta$ for a small δ, where

$$\begin{aligned}\frac{\mathrm{d}\alpha}{\mathrm{d}z} &= (\gamma_s + \gamma_e)F(z;\mu,\sigma) - \gamma_s, \\ \frac{\mathrm{d}^2\alpha}{\mathrm{d}z^2} &= (\gamma_s + \gamma_e)p(z;\mu,\sigma).\end{aligned} \tag{5.18}$$

We then compute the necessary Jacobians using the quadratic approximation (5.17) at the solution, which gives the correct Hessian and gradient terms. We can furthermore differentiate the gradient and Hessian with respect to the underlying model parameters μ and σ^2, again using a recently-developed batch QP solver [AK17].

To develop a predictive model, we make use of a highly-tuned load forecasting methodology. Specifically, we input the past day's electrical load and temperature, the next day's temperature forecast, and additional features such as nonlinear functions of the temperatures, binary indicators of weekends or holidays, and yearly sinusoidal features. We then predict the electrical load over all 24 hours of the next day. We employ a 2-hidden-layer neural network for this purpose, with an additional residual connection from the inputs to the outputs initialized to the linear regression solution. An illustration of the architecture is shown in Figure 5.3.

[6] Part of the philosophy behind using this approach is that we *know* the Gaussian assumption is incorrect: the true underlying load is neither Gaussian distributed nor homoskedastic. However, these assumptions are exceedingly common in practice, as they enable easy model learning and exact analytical solutions. Thus, training the (still Gaussian) system with a task-based loss retains computational tractability while still allowing us to modify the distribution parameters to improve actual performance on the task at hand.

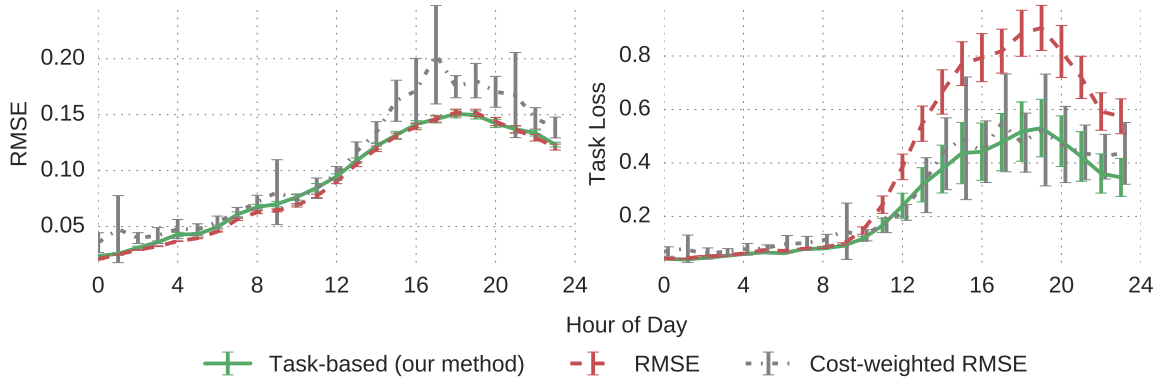


Figure 5.4: Results for 10 runs of the generation-scheduling problem for representative decision parameters $\gamma_e = 0.5, \gamma_s = 50$, and $c_r = 0.4$. (Lower loss is better.) As expected, the RMSE net achieves the lowest RMSE for its predictions. However, the task net outperforms the RMSE net on task loss by 38.6%, and the cost-weighted RMSE on task loss by 8.6%.

In our approach, we first pre-train the model to minimize the root mean squared error (RMSE) between its predictions and the actual load (giving the mean prediction μ_i), and compute σ_i^2 as the (constant) empirical variance between the predicted and actual values. We then fine-tune this model using the task loss; that is, using the (mean and variance) predictions of this base model, we obtain $z^\star(x;\theta)$ by solving the generator scheduling problem (5.15), and then adjusting network parameters to minimize the resultant task loss. We use 7 years of data to train the model, and 1.75 subsequent years for testing.

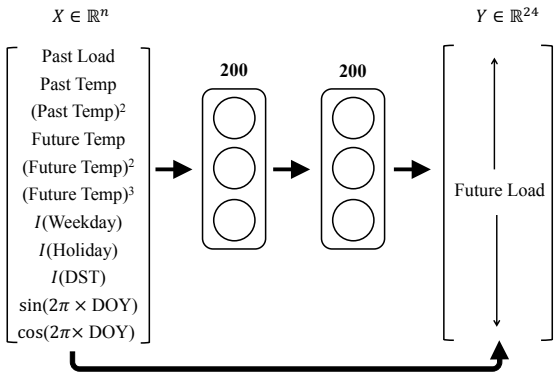


Figure 5.3: 2-hidden-layer neural network to predict hourly electric load for the next day.

We compare against a traditional stochastic programming model that minimizes just the RMSE, as well as a cost-weighted RMSE that periodically reweights training samples given their task loss.[7] A pure policy-optimizing network is not shown, as it could not sufficiently learn the ramp constraints and thus generated infeasible schedules; we could not obtain good performance for the policy optimizer even ignoring this infeasibility.

Figure 5.4 shows the performance of the three models on the testing dataset. As expected, the RMSE model performs best with respect to the RMSE of its predictions (its objective). However, the task-based model substantially outperforms the RMSE model when evaluated on task loss, the actual objective that the system operator cares about:

[7]It is worth noting that a cost-weighted RMSE approach is only possible when direct costs can be assigned independently to each decision point, i.e. when costs do not depend on multiple decision points (as in this experiment). Our task-based method, however, accommodates the (typical) more general setting.

specifically, we improve upon the performance of the traditional stochastic programming method by 38.6%. The cost-weighted RMSE's performance is extremely variable, and overall, the task net improves upon this method by 8.6%.

5.4.3 Price forecasting and battery storage

Finally, we consider a battery arbitrage task, based upon 6 years of real electrical grid data. Here, a grid-scale battery must operate over a 24 hour period based on some (unknown) distribution over future electricity prices (Figure 5.1c). For each hour, the operator must decide how much to charge ($z_{\text{in}} \in \mathbb{R}^{24}$) or discharge ($z_{\text{out}} \in \mathbb{R}^{24}$) the battery, thus inducing a particular state of charge in the battery ($z_{\text{state}} \in \mathbb{R}^{24}$). Given a particular realization y of prices, the operator optimizes over: 1) profits, 2) flexibility to participate in other markets, by keeping the battery near half its capacity B (with weight λ), and 3) battery health, by discouraging rapid charging/discharging (with weight ϵ, $\epsilon < \lambda$). The battery also has a charging efficiency (γ_{eff}), limits on speed of charge (c_{in}) and discharge (c_{out}), and begins at half charge. This can be written as the stochastic programming problem

$$\begin{array}{ll} \underset{z_{\text{in}}, z_{\text{out}}, z_{\text{state}} \in \mathbb{R}^{24}}{\text{minimize}} & \mathbf{E}_{y \sim p(y|x;\theta)} \left[\displaystyle\sum_{i=1}^{24} y_i (z_{\text{in}} - z_{\text{out}})_i + \lambda \left\| z_{\text{state}} - \frac{B}{2} \right\|^2 + \epsilon \|z_{\text{in}}\|^2 + \epsilon \|z_{\text{out}}\|^2 \right] \\ \text{subject to} & z_{\text{state},i+1} = z_{\text{state},i} - z_{\text{out},i} + \gamma_{\text{eff}} z_{\text{in},i} \; \forall i, \quad z_{\text{state},1} = B/2, \\ & 0 \le z_{\text{in}} \le c_{\text{in}}, \quad 0 \le z_{\text{out}} \le c_{\text{out}}, \quad 0 \le z_{\text{state}} \le B. \end{array} \tag{5.19}$$

Assuming (as we will in our model) that y_i is a random variable with mean μ_i, then this expectation has a closed form that depends only on the mean, as follows:

$$\begin{aligned} &\mathbf{E}_{y \sim p(y|x;\theta)} \left[\sum_{i=1}^{24} y_i (z_{\text{in}} - z_{\text{out}})_i + \lambda \left\| z_{\text{state}} - \frac{B}{2} \right\|^2 + \epsilon \|z_{\text{in}}\|^2 + \epsilon \|z_{\text{out}}\|^2 \right] \\ &= \sum_{i=1}^{24} \mu_i (z_{\text{in}} - z_{\text{out}})_i + \lambda \left\| z_{\text{state}} - \frac{B}{2} \right\|^2 + \epsilon \|z_{\text{in}}\|^2 + \epsilon \|z_{\text{out}}\|^2. \end{aligned} \tag{5.20}$$

We can then write this expression in QP form as $\text{minimize}_{\{\boldsymbol{z}: G\boldsymbol{z} \le h,\, A\boldsymbol{z} = b\}} \; \frac{1}{2} \boldsymbol{z}^T Q \boldsymbol{z} + c^T \boldsymbol{z}$ with

$$\begin{gathered} \boldsymbol{z} = \begin{bmatrix} z_{\text{in}} \\ z_{\text{out}} \\ z_{\text{state}} \end{bmatrix}, \; Q = \begin{bmatrix} \epsilon I & 0 & 0 \\ 0 & \epsilon I & 0 \\ 0 & 0 & \lambda I \end{bmatrix}, \; c = \begin{bmatrix} \mu \\ -\mu \\ -\lambda B \mathbf{1} \end{bmatrix}, \\ G = \begin{bmatrix} I & 0 & 0 \\ -I & 0 & 0 \\ 0 & I & 0 \\ 0 & -I & 0 \\ 0 & 0 & I \\ 0 & 0 & -I \end{bmatrix}, \; h = \begin{bmatrix} c_{\text{in}} \\ 0 \\ c_{\text{out}} \\ 0 \\ B \\ 0 \end{bmatrix}, \; A = \begin{bmatrix} 0 & 0 & 0, \ldots, 0, 1 \\ \gamma_{\text{eff}} D_1^T & -D_1^T & D_1^T - D_2^T \end{bmatrix} \; b = \begin{bmatrix} B/2 \\ 0 \end{bmatrix}, \end{gathered} \tag{5.21}$$

Table 5.1: Task loss results for 10 runs each of the battery storage problem, given a lithium-ion battery with attributes $B = 1$, $\gamma_{\text{eff}} = 0.9$, $c_{\text{in}} = 0.5$, and $c_{\text{out}} = 0.2$. (Lower loss is better.) Our task-based net on average somewhat improves upon the RMSE net, and demonstrates more reliable performance.

Hyperparameters		**RMSE net**	**Task-based net (our method)**	**% Improvement**
λ	ϵ			
0.1	0.05	-1.45 ± 4.67	-2.92 ± 0.3	102
1	0.5	4.96 ± 4.85	2.28 ± 2.99	54
10	5	131 ± 145	95.9 ± 29.8	27
35	15	173 ± 7.38	170 ± 2.16	2

where $D_1 = \begin{bmatrix} I & 0 \end{bmatrix}^T \in \mathbb{R}^{24\times 23}$ and $D_2 = \begin{bmatrix} 0 & I \end{bmatrix}^T \in \mathbb{R}^{24\times 23}$.

For this experiment, we assume that y_i is a lognormal random variable (with mean μ_i); thus, to obtain our predictions, we predict the mean of $\log(y)$ (i.e., we predict $\log(\mu)$). To develop a predictive model for the mean, we use an architecture similar to that described in Section 5.4.2. In this case, we input the past day's prices and temperature, the next day's load forecasts and temperature forecasts, and additional features such as nonlinear functions of the temperatures and temporal features similar to those in Section 5.4.2. We again pre-train the model to minimize the mean squared error between the model's predictions and the actual prices (giving the mean prediction μ_i), using about 5 years of data to train the model and 1 subsequent year for testing. We then again fine-tune the neural network parameters by minimizing the task loss; that is, using the mean predictions of this base model, we solve the storage scheduling problem by solving the optimization problem (5.19), compute the necessary Jacobians at the solution, and update the underlying model parameter μ via backpropagation, again using Amos and Kolter [AK17]. We compare against a traditional stochastic programming model that minimizes just the RMSE.

Table 5.1 shows the performance of the two models on the testing dataset. As energy prices are difficult to predict due to numerous outliers and price spikes, the models in this case are not as well-tuned as in our load forecasting experiment; thus, their performance is relatively variable. Even then, in all cases, our task-based model demonstrates better average performance than the RMSE model when evaluated on task loss, the objective most important to the battery operator (although the improvements are not statistically significant). More interestingly, our task-based method shows less (and in some cases, far less) variability in performance than the RMSE-minimizing method. Qualitatively, our task-based method hedges against perverse events such as price spikes that could substantially affect the performance of a battery charging schedule. The task-based method thus yields more reliable performance than a pure RMSE-minimizing method in the case the models are inaccurate due to a high level of stochasticity in the prediction task.

5.5 Conclusion

This chapter proposes an end-to-end approach for training machine learning models that will be used in the loop of a larger process. Specifically, we consider training probabilistic

models in the context of stochastic programming to directly capture a task-based objective. Preliminary experiments indicate that our task-based learning model substantially outperforms MLE and policy-optimizing approaches in all but the (rare) case that the MLE model "perfectly" characterizes the underlying distribution. Our method also achieves a 38.6% performance improvement over a highly-optimized real-world stochastic programming algorithm for scheduling electricity generation based on predicted load. In the case of energy price prediction, where there is a high degree of inherent stochasticity in the problem, our method demonstrates more reliable task performance than a traditional predictive method. The task-based approach thus demonstrates promise in optimizing in-the-loop predictions.

While not explicitly evaluated above, a key limitation of our method is its computational cost; in particular, our method requires solving and differentiating through an optimization problem during each training iteration where the task loss is used, which can be potentially expensive. Future directions include exploring mechanisms to reduce this computational cost, e.g., via faster solution methods or by employing cheaper proxies to the decision-making process. Our method as currently presented also only accounts for certain kinds of decision-making settings, notably focusing on single-round decision-making processes that can be written in analytical form. Future work includes an extension of our approach to stochastic learning models with multiple rounds, further to model predictive control and full reinforcement learning settings, and to multi-objective optimization settings.

Chapter 6

Approximating Optimization Problems with Hard Constraints

Large optimization problems with hard constraints arise in many settings, yet classical solvers are often prohibitively slow, motivating the use of deep networks as cheap "approximate solvers." Unfortunately, naive deep learning approaches typically cannot enforce the hard constraints of such problems, leading to infeasible solutions. In this chapter, we present Deep Constraint Completion and Correction (DC3), an algorithm to address this challenge. Specifically, this method enforces feasibility via a differentiable procedure, which implicitly completes partial solutions to satisfy equality constraints and unrolls gradient-based corrections to satisfy inequality constraints. We demonstrate the effectiveness of DC3 in both synthetic optimization tasks and the real-world setting of AC optimal power flow, where hard constraints encode the physics of the electrical grid. In both cases, DC3 achieves near-optimal objective values while preserving feasibility.

The work in this chapter has previously been published in:[1]

> Priya L. Donti*, David Rolnick*, and J. Zico Kolter. "DC3: A Learning Method for Optimization with Hard Constraints." *International Conference on Learning Representations*. 2021.

[1] Code for all experiments is available at: https://github.com/locuslab/DC3.

6.1 Introduction

Traditional approaches to constrained optimization are often expensive to run for large problems, necessitating the use of function approximators. Neural networks are highly expressive and fast to run, making them ideal as function approximators. However, while deep learning has proven its power for unconstrained problem settings, it has struggled to perform well in domains where it is necessary to satisfy hard constraints at test time. For example, in power systems, weather and climate models, materials science, and many other areas, data follows well-known physical laws, and violation of these laws can lead to answers that are unhelpful or even nonsensical. There is thus a need for fast neural network approximators that can operate in settings where traditional optimizers are slow (such as non-convex optimization), yet where strict feasibility criteria must be satisfied.

In this work, we introduce Deep Constraint Completion and Correction (DC3), a framework for applying deep learning to optimization problems with hard constraints. Our approach embeds differentiable operations into the training of the neural network to ensure feasibility. Specifically, the network outputs a partial set of variables with codimension equal to the number of equality constraints, and "completes" this partial set into a full solution. This completion process guarantees feasibility with respect to the equality constraints and is differentiable (either explicitly, or via the implicit function theorem). We then fix any violations of the inequality constraints via a differentiable correction procedure based on gradient descent. Together, this process of completion and correction enables feasibility with respect to all constraints. Further, this process is fully differentiable and can be incorporated into standard deep learning methods.

Our key contributions are:

- **Framework for incorporating hard constraints.** We describe a general framework, DC3, for incorporating (potentially non-convex) equality and inequality constraints into deep-learning-based optimization algorithms.
- **Practical demonstration of feasibility.** We implement the DC3 algorithm in both convex and non-convex optimization settings. We demonstrate the success of the algorithm in producing approximate solutions with significantly better feasibility than other deep learning approaches, while maintaining near-optimality of the solution.
- **AC optimal power flow.** We show how the general DC3 framework can be used to optimize power flows on the electrical grid. This difficult non-convex optimization task must be solved at scale and is especially critical for renewable energy adoption. Our results greatly improve upon the performance of general-purpose deep learning methods on this task.

6.2 Related work

Fast optimization methods. Many classical optimization methods have been proposed to improve the practical efficiency of solving optimization problems. These include general techniques such as constraint and variable elimination (i.e., the removal of non-active constraints or redundant variables, respectively), as well as problem-specific techniques

(e.g., KKT factorization techniques in the case of convex quadratic programs) [NW06]. Our present work builds upon aspects of this literature, applying concepts from variable elimination to reduce the number of degrees of freedom associated with the optimization problems we wish to solve.

In addition to the classical optimization literature, there has been a large body of literature in deep learning that has sought to approximate or speed up optimization models. As described in reviews on topics such as combinatorial optimization [BLP21] and optimal power flow [HKM20], ML methods to speed up optimization models have thus far taken two main approaches. The first class of approaches has focused on employing machine learning alongside or in the loop of optimization solvers – for instance, to learn warm-start points (see, e.g., [Bak19] and [Don+20]) or to enable constraint elimination techniques by predicting active constraints (see, e.g., [MRN22]). The second class of approaches, akin to work on surrogate modeling [KL13], involves training machine learning models to map directly from optimization inputs to full solutions; however, such approaches often struggle to produce solutions that are feasible with respect to the optimization constraints. Several recent works within this second class of approaches have therefore aimed to make post-hoc feasibility adjustments to the solutions output by the ML models [ZB20; Pan+20], but as these adjustments are not visible to the ML model during training, this can impact the quality of the solutions that are obtained. Our work is situated within this latter line of approaches, aiming to directly learn a machine learning-based surrogate to optimization problems, but in a way that *internalizes* knowledge of any feasibility adjustments into the training process in order to improve learning outcomes.

Constraints in neural networks. While deep learning is often thought of as wholly unconstrained, in reality, it is quite common to incorporate (simple) constraints within deep learning procedures. For instance, softmax layers encode simplex constraints, sigmoids instantiate upper and lower bounds, ReLUs encode projections onto the positive orthant, and convolutional layers enforce translational equivariance (an idea taken further in general group-equivariant networks [CW16]). Recent work has also focused on embedding specialized kinds of constraints into neural networks, such as conservation of energy (see, e.g., [GDY19] and [Beu+19]), and homogeneous linear inequality constraints [FNC20]. However, while these represent common "special cases," there has to date been little work on building more general hard constraints into deep learning models.

6.3 DC3: Deep constraint completion and correction

In this work, we consider solving families of optimization problems for which the objectives and/or constraints vary across instances. Formally, let $x \in \mathbb{R}^d$ denote the problem data, and $y \in \mathbb{R}^n$ denote the solution of the corresponding optimization problem (where y depends on x). For any given x, our aim is then to find y solving:

$$\underset{y \in \mathbb{R}^n}{\text{minimize}} \;\; f_x(y), \;\; \text{s.t. } g_x(y) \leq 0, \; h_x(y) = 0, \tag{6.1}$$

(where f, g, and h are potentially nonlinear and non-convex). Solving such a family of optimization problems can be framed as a learning problem, where an algorithm must predict an optimal y from the problem data x. We consider deep learning approaches to this task – that is, training a neural network N_θ, parameterized by θ, to approximate y given x.

A naive deep learning approach to approximating such a problem involves viewing the constraints as a form of regularization. That is, for training examples $x^{(i)}$, the algorithm learns to minimize a composite loss containing both the objective and two "soft loss" terms representing violations of the equality and inequality constraints (for some $\lambda_g, \lambda_h > 0$):

$$\ell_{\text{soft}}(\hat{y}) = f_x(\hat{y}) + \lambda_g \| \operatorname{ReLU}(g_x(\hat{y}))\|_2^2 + \lambda_h \|h_x(\hat{y})\|_2^2. \tag{6.2}$$

An alternative framework (see, e.g., Zamzam and Baker [ZB20] and Pan, Chen, Zhao, and Low [Pan+20]) is to use supervised learning on examples $(x^{(i)}, y^{(i)})$ for which an optimum $y^{(i)}$ is known. In this case, the loss is simply $||\hat{y} - y^{(i)}||_2^2$. However, both these procedures for training a neural network can lead in practice to highly infeasible outputs (as we demonstrate in our experiments), because they do not strictly enforce constraints. Supervised methods also require constructing a training set (e.g., via an exact solver), a sometimes difficult or expensive step.

To address these challenges, we introduce the method of Deep Constraint Completion and Correction (DC3), which allows hard constraints to be integrated into the training of neural networks. This method is able to train directly from the problem specification (instead of a supervised dataset), via the following two innovations:

Equality completion. We provide a mechanism to enforce equality constraints during training and testing, inspired by the literature on variable elimination. Specifically, rather than outputting the full-dimensional optimization solution directly, we first output a subset of the variables, and then infer the remaining variables via the equality constraints – either explicitly, or by solving an implicit set of equations (through which we can then backpropagate via the implicit function theorem).

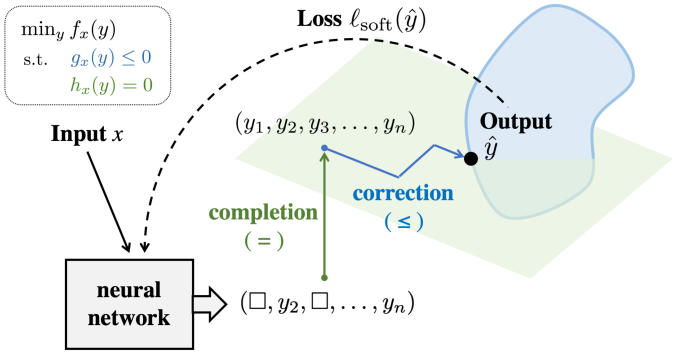


Figure 6.1: A schematic of the DC3 framework.

Inequality correction. We correct for violation of the inequality constraints by mapping infeasible points to feasible points using an internal gradient descent procedure during training. This allows us to fix inequality violations while taking steps along the manifold of points satisfying the equalities, which yields an output that is feasible with respect to all constraints.

Overall, our algorithm involves training a neural network $N_\theta(x)$ to output a partial set of variables z. These variables are then completed to a full set of variables $\tilde{y}$ satisfying the equality constraints. In turn, $\tilde{y}$ is corrected to $\hat{y}$ to satisfy the inequality constraints

Algorithm 2 Deep Constraint Completion and Correction (DC3)

assume equality completion procedure $\varphi_x : \mathbb{R}^m \to \mathbb{R}^{n-m}$ *// to solve eq. constraints*

procedure TRAIN(X)
  init neural network $N_\theta : \mathbb{R}^d \to \mathbb{R}^m$
  while not converged **do** for $x \in X$
    compute partial set of variables $z = N_\theta(x)$
    complete to full set of variables $\tilde{y} = \begin{bmatrix} z^T & \varphi_x(z)^T \end{bmatrix}^T \in \mathbb{R}^n$
    correct to feasible (or approx. feasible) solution $\hat{y} = \rho_x^{(t_{\text{train}})}(\tilde{y})$
    compute constraint-regularized loss $\ell_{\text{soft}}(\hat{y})$
    update θ using $\nabla_\theta \ell_{\text{soft}}(\hat{y})$
  end while
end procedure

procedure TEST(x, N_θ)
  compute partial set of variables $z = N_\theta(x)$
  complete to full set of variables $\tilde{y} = \begin{bmatrix} z^T & \varphi_x(z)^T \end{bmatrix}^T$
  correct to feasible solution $\hat{y} = \rho^{(t_{\text{test}})}(\tilde{y})$
  return $\hat{y}$
end procedure

while continuing to satisfy the equality constraints. The overall network is trained using backpropagation on the soft loss described in Equation (6.2) (which is necessary for correction, as noted below).

Importantly, both the completion and correction procedures are differentiable either implicitly or explicitly (allowing network training to take them into account), and the overall framework is agnostic to the choice of neural network architecture. A schematic of the DC3 framework is given in Figure 6.1, and corresponding pseudocode is given in Algorithm 2.

We note that as this procedure is somewhat general, in cases where constraints have a specialized structure, specialized techniques may be more appropriate to use. For instance, while we examine linearly-constrained settings in our experiments for the purposes of illustration, in practice, techniques such as Minkowski-Weyl decomposition or Cholesky factorization (see [FNC20], [AK17]) may be more efficient in these settings. However, for more general settings without this kind of structure – e.g., non-convex problems such as AC optimal power flow, which we examine in our experiments – the DC3 framework can provide a (differentiable) mechanism for satisfying hard constraints. We now detail the completion and correction procedures used in DC3.

6.3.1 Equality completion

Assuming that the problem (6.1) is not overdetermined, the number of equality constraints $h_x(y) = 0$ must not exceed the dimension of the decision variable $y \in \mathbb{R}^n$: that is, the number of equality constraints equals $n - m$ for some $m \geq 0$. Then, given m of the

entries of a feasible point y, the other $(n-m)$ entries are, in general, determined by the $(n-m)$ equality constraints. We exploit this observation in our algorithm, noting that it is considerably more efficient to output a point in $\mathbb{R}^m$ and complete it to a point $y \in \mathbb{R}^n$ such that $h_x(y) = 0$, as compared with outputting full-dimensional points $y \in \mathbb{R}^n$ and attempting to adjust all coordinates to satisfy the equality constraints.

In particular, we assume that given m entries of y, we either can solve for the remaining entries explicitly (e.g. in a linear system) or that we have access to a process (e.g. Newton's Method) allowing us to solve any implicit equations. Formally, we assume access to a function $\varphi_x : \mathbb{R}^m \to \mathbb{R}^{n-m}$ such that $h_x(\begin{bmatrix} z^T & \varphi_x(z)^T \end{bmatrix}^T) = 0$, where $\begin{bmatrix} z^T & \varphi_x(z)^T \end{bmatrix}^T$ is the concatenation of z and $\varphi_x(z)$.

In our algorithm, we then train our neural network N_θ to output points $z \in \mathbb{R}^m$, which are completed to $\begin{bmatrix} z^T & \varphi_x(z)^T \end{bmatrix}^T \in \mathbb{R}^n$. A challenge then arises as to how to backpropagate the loss during the training of N_θ if $\varphi_x(z)$ is not a readily differentiable explicit function – for example, if the completion procedure uses Newton's Method. We solve this challenge by leveraging the implicit function theorem, as e.g. in OptNet [AK17] and SATNet [Wan+19]. This approach allows us to express, for any training loss ℓ, the derivatives $\mathrm{d}\ell/\mathrm{d}z$ using $\mathrm{d}\ell/\mathrm{d}\varphi_x(z)$.

Namely, let $J^h \in \mathbb{R}^{(n-m)\times n}$ denote the Jacobian of $h_x(y)$ with respect to y. By the chain rule:

$$0 = \frac{\mathrm{d}}{\mathrm{d}z} h_x\left(\begin{bmatrix} z \\ \varphi_x(z) \end{bmatrix}\right) = \frac{\partial h_x}{\partial z} + \frac{\partial h_x}{\partial \varphi_x(z)} \frac{\mathrm{d}\varphi_x(z)}{\mathrm{d}z} = J^h_{:,0:m} + J^h_{:,m:n} \frac{\mathrm{d}\varphi_x(z)}{\mathrm{d}z},$$

$$\Rightarrow \quad \mathrm{d}\varphi_x(z)/\mathrm{d}z = -\left(J^h_{:,m:n}\right)^{-1} J^h_{:,0:m}. \tag{6.3}$$

We can then backpropagate losses through the network by noting that

$$\frac{\mathrm{d}\ell}{\mathrm{d}z} = \frac{\partial \ell}{\partial z} + \frac{\partial \ell}{\partial \varphi_x(z)} \frac{\mathrm{d}\varphi_x(z)}{\mathrm{d}z} = \frac{\partial \ell}{\partial z} - \frac{\partial \ell}{\partial \varphi_x(z)} \left(J^h_{:,m:n}\right)^{-1} J^h_{:,0:m}. \tag{6.4}$$

We note that in practice, the Jacobian $\mathrm{d}\varphi_x(z)/\mathrm{d}z$ should generally not be computed explicitly due to space complexity considerations, and that it is often desirable to form the result of the left matrix-vector product $(\partial\ell/\partial\varphi_x(z))(\mathrm{d}\varphi_x(z)/\mathrm{d}z)$ directly. This general process is discussed in Section 2.2.3, and also shown in detail for the problem of AC optimal power flow in Appendix B.

6.3.2 Inequality correction

While the completion procedure described above guarantees feasibility with respect to the equality constraints, it does not ensure that the inequality constraints will be satisfied. To additionally ensure feasibility with respect to the inequality constraints, our algorithm incorporates a correction procedure that maps the outputs from the previous step into the feasible region. In particular, we employ a gradient-based correction procedure that takes gradient steps in z towards the feasible region *along the manifold of points satisfying the equality constraints*.

Let $\rho_x(y)$ be the operation that takes as input a point $y = \begin{bmatrix} z^T & \varphi_x(z)^T \end{bmatrix}^T$, and moves it closer to satisfying the inequality constraints by taking a step along the gradient of the inequality violation with respect to the partial variables z. Formally, for a learning rate $\gamma > 0$, we define:

$$\rho_x\left(\begin{bmatrix} z \\ \varphi_x(z) \end{bmatrix}\right) = \begin{bmatrix} z - \gamma \Delta z \\ \varphi_x(z) - \gamma \Delta \varphi_x(z) \end{bmatrix},$$

$$\text{for} \quad \Delta z = \nabla_z \left\| \text{ReLU}\left(g_x\left(\begin{bmatrix} z \\ \varphi_x(z) \end{bmatrix}\right)\right)\right\|_2^2, \quad \Delta \varphi_x(z) = \frac{\mathrm{d}\varphi_x(z)}{\mathrm{d}z} \Delta z.$$

While gradient descent methods do not always converge to global (or local) optima for general optimization problems, if initialized *close to an optimum*, gradient descent is highly effective in practice for non-pathological settings (see e.g. Busseti, Moursi, and Boyd [BMB19] and Panageas, Piliouras, and Wang [PPW19]). At test time, the input to the DC3 correction procedure should already be close to feasible with respect to the inequality constraints, as it is the output of a differentiable completion process that is trained using the soft loss ℓ_{soft}. Therefore, we may expect that in practice, the limit $\lim_{t\to\infty} \rho_x^{(t)}(y)$ will converge to a point satisfying both inequality and equality constraints (while for problems with linear constraints as in Sections 6.4.1–6.4.2, the correction process is mathematically guaranteed to converge).

As the exact limit $\lim_{t\to\infty} \rho_x^{(t)}(y)$ is difficult to calculate in practice, we make approximations at both training and test time. Namely, we apply $\rho_x^{(t)}(y)$ to the output of the completion procedure, with $t = t_{\text{train}}$ relatively small at train time to allow backpropagation through the correction. Depending on time constraints, this same value of t may be used at test time, or a larger value $t = t_{\text{test}} > t_{\text{train}}$ may be used to ensure convergence to a feasible point.

6.4 Experiments

We evaluate DC3 for convex quadratic programs (QPs), a simple family of non-convex optimization problems, and the real-world task of AC optimal power flow (ACOPF). In particular, we assess our method on the following criteria:

- **Optimality**: How good is the objective value $f_x(y)$ achieved by the final solution?
- **Feasibility**: How much, if at all, does the solution violate the constraints? Specifically, what are the maximum and mean feasibility violations of the inequality and equality constraints: $\max(\text{ReLU}(g_x(y)))$, $\text{mean}(\text{ReLU}(g_x(y)))$, $\max(h_x(y))$, and $\text{mean}(h_x(y))$?
- **Speed**: How fast is the method?

We compare DC3 against the following methods (referred to by abbreviations in our tables below):

- **Optimizer**: A traditional optimization solver. For convex QP settings, we use `OSQP` [Ste+20], as well as the batched, differentiable solver `qpth` developed as part of OptNet [AK17]. For the generic non-convex setting, we use `IPOPT` [WB06]. For ACOPF, we use the solver provided by `PYPOWER`, a Python port of `MATPOWER` [ZMSG97].

- **NN**: A simple deep learning approach trained to minimize a soft loss (Equation (6.2)).
- **NN, $\leq$ test**: The NN approach, with a gradient-based correction procedure[2] applied to the output at test time in an effort to mitigate violations of equality and inequality constraints. Unlike in DC3, correction is not used during training, and completion is not used at all.
- **Eq. NN**: A more sophisticated approach inspired by[3] that in Zamzam and Baker [ZB20], where (i) the neural network outputs a partial set of variables $\hat{z}$, which is completed to the full set using the equality constraints, (ii) training is performed by supervised learning on optimal pairs $(x^{(i)}, z^{(i)})$ with loss function $||\hat{z} - z^{(i)}||_2^2$, not using the objective value at all.
- **Eq. NN, $\leq$ test**: The approach in Eq. NN, augmented with gradient-based correction at test time to mitigate violations of equality and inequality constraints.

We also attempted to use the output of the **NN** method as a "warm start" for traditional optimizers, but found that the **NN** output was sufficiently far from feasibility that it did not help.

In addition, we consider weaker versions of DC3 in which components of the algorithm are ablated:

- **DC3, $\neq$**. The DC3 algorithm with completion ablated. All variables are output by the network directly and correction is performed by taking gradient steps for both equality and inequality constraints.
- **DC3, $\not\leq$ train**. The DC3 algorithm with correction ablated at train time. Correction is still performed at test time.
- **DC3, $\not\leq$ train/test**. The DC3 algorithm with correction ablated at both train and test time.
- **DC3, no soft loss**. The DC3 algorithm with training performed to minimize the objective value only, without auxiliary terms capturing equality and inequality violation.

As our overall framework is agnostic to the choice of neural network architecture, to facilitate comparison, we use a fixed neural network architecture across all experiments: fully connected with two hidden layers of size 200, including ReLU activation, batch normalization [IS15], and dropout (with rate 0.2) [Sri+14] at each hidden layer. The following parameters were kept fixed for all neural network-based methods across all experiments (with values given

[2]Note that this correction procedure is not exactly the same as that described in Section 6.3.2, as the outputs of the NN baseline do not necessarily meet the prerequisite of satisfying the equality constraints. Instead, we adjust the full set of output variables directly with respect to gradients of the inequality and equality violations.

[3]In Zamzam and Baker [ZB20], the authors employ one step of an ACOPF-specific heuristic called PV/PQ switching to correct inequality constraint violations at test time. We do not apply this heuristic here in the spirit of presenting a more general framework. As PV/PQ switching is not necessarily guaranteed to correct all inequality violations (although it can work well in practice), in principle, one could consider employing a combination of PV/PQ switching *and* gradient-based corrections in the context of ACOPF. We note that Pan, Chen, Zhao, and Low [Pan+20] propose a more general post-hoc inequality correction scheme, but we do not explore this empirically here.

in parentheses), based on a small amount of informal experimentation to ensure training was stable and properly converged: number of epochs (1000), batch size (200), hidden layer size (200, for both hidden layers), correction procedure stopping tolerance (10^{-4}), and correction procedure momentum (0.5). For all remaining parameters, we performed hyperparameter tuning via a coordinate search over relevant parameters. Central values for this coordinate search were determined via a small amount of informal experimentation to ensure training was stable on that central set of parameters. Final parameters were chosen so as to prioritize feasibility (as determined via the mean and max violations of equality and inequality constraints), followed by objective value and speed; these parameter values are reported alongside each experiment below. All neural networks are trained using PyTorch [Pas+19].

To generate timing results, all neural nets and the `qpth` optimizer were run with full parallelization on a GeForce GTX 2080 Ti GPU. The `OSQP`, `IPOPT`, and `PYPOWER` optimizers were run sequentially on an Intel Xeon 2.10GHz CPU, and we report the total time divided by the number of test instances to simulate full parallelization. As our implementations are not tightly optimized, we emphasize that all timing comparisons are approximate.

6.4.1 Convex quadratic programs

As a first test of the DC3 method, we consider solving convex quadratic programs with a quadratic objective function and linear constraints. Note that we examine this simple task first for illustration, but the general DC3 method is assuredly overkill for solving convex quadratic programs. It may not even be the most efficient deep learning-based method for constraint enforcement on this task, since more specialized techniques are available in such linearly constrained settings [FNC20].

We consider the following problem:

$$\underset{y\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}y^TQy + p^Ty, \ \text{ s.t. } Ay = x, \ Gy \leq h, \tag{6.5}$$

for constants $Q \in \mathbb{R}^{n\times n} \succeq 0$, $p \in \mathbb{R}^n$, $A \in \mathbb{R}^{n_{\text{eq}}\times n}$, $G \in \mathbb{R}^{n_{\text{ineq}}\times n}$, $h \in \mathbb{R}^{n_{\text{ineq}}}$, and variable $x \in \mathbb{R}^{n_{\text{eq}}}$ which varies between problem instances. We must learn to approximate the optimal y given x.

In our experiments, we take Q to be a diagonal matrix with all diagonal entries drawn i.i.d. from the uniform distribution on $[0, 1]$, ensuring that Q is positive semi-definite. We take matrices A, G with entries drawn i.i.d. from the unit normal distribution. We assume that in each problem instance, all entries of x are in the interval $[-1, 1]$. In order to ensure that the problem is feasible, we take $h = \sum_j |(GA^+)_{ij}|$, where A^+ is the Moore-Penrose pseudoinverse of A; namely, for this choice of h, the point $y = A^+x$ is feasible (but not, in general, optimal), because:

$$AA^+x = x, \qquad GA^+x \leq \sum_j \left|(GA^+)_{ij}\right| \ \text{ since } |x_j| \leq 1. \tag{6.6}$$

During training, we use examples x with entries drawn i.i.d. from the uniform distribution on $[-1, 1]$.

Table 6.1: Results on QP task for 100 variables, 50 equality constraints, and 50 inequality constraints. We compare the performance of DC3 and other algorithms according to the objective value and max/mean values of equality/inequality constraint violations, each averaged across test instances. We also compare the total time required to run on all 833 test instances, assuming full parallelization. (Std. deviations across 5 runs are shown in parentheses for all figures reported.) Lower values are better for all metrics. We find that methods other than DC3 and Optimizer violate feasibility (as shown in red). DC3 gives a feasible output with reasonable objective value 78× faster than `qpth` and only 9× slower than `OSQP`, which is optimized for convex QPs.

	Obj. value	Max eq.	Mean eq.	Max ineq.	Mean ineq.	Time (s)
Optimizer (OSQP)	-15.05 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.002 (0.000)
Optimizer (qpth)	-15.05 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1.335 (0.012)
DC3	-13.46 (0.01)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.017 (0.001)
DC3, $\neq$	-12.58 (0.04)	0.35 (0.00)	0.13 (0.00)	0.00 (0.00)	0.00 (0.00)	0.008 (0.000)
DC3, $\not\leq$ train	-1.39 (0.97)	0.00 (0.00)	0.00 (0.00)	0.02 (0.02)	0.00 (0.00)	0.017 (0.000)
DC3, $\not\leq$ train/test	-1.23 (1.21)	0.00 (0.00)	0.00 (0.00)	0.09 (0.13)	0.01 (0.01)	0.001 (0.000)
DC3, no soft loss	-21.84 (0.00)	0.00 (0.00)	0.00 (0.00)	23.83 (0.11)	4.04 (0.01)	0.017 (0.000)
NN	-12.57 (0.01)	0.35 (0.00)	0.13 (0.00)	0.00 (0.00)	0.00 (0.00)	0.001 (0.000)
NN, $\leq$ test	-12.57 (0.01)	0.35 (0.00)	0.13 (0.00)	0.00 (0.00)	0.00 (0.00)	0.008 (0.000)
Eq. NN	-9.16 (0.75)	0.00 (0.00)	0.00 (0.00)	8.83 (0.72)	0.91 (0.09)	0.001 (0.000)
Eq. NN, $\leq$ test	-14.68 (0.05)	0.00 (0.00)	0.00 (0.00)	0.89 (0.05)	0.07 (0.01)	0.018 (0.001)

Table 6.1 compares the performance of DC3 (and various ablations of DC3) with traditional optimizers and other deep learning-based methods, for the case of $n = 100$ variables and $n_{\text{eq}} = n_{\text{ineq}} = 50$. Tables 6.2 and 6.3 additionally show the performance of these methods as the number of equality and/or inequality constraints vary. Each experiment is run 5 times for 10,000 examples x (with train/test/validation ratio 10:1:1). Hyperparameter values are shown in Table 6.4.

We find that DC3 preserves feasibility with respect to both equality and inequality constraints, while achieving reasonable objective values. (The average per-instance optimality gap for DC3 over the classical optimizer is 10.59%.) For every baseline deep learning algorithm, on the other hand, feasibility is violated significantly for either equality or inequality constraints. As expected, "DC3 $\neq$" (completion ablated) results in violated equality constraints, while "DC3 $\not\leq$" (correction ablated) violates inequality constraints. Ablating the soft loss also results in violated inequality constraints, leading to an objective value significantly lower than would be possible were constraints satisfied.

Even though we have not optimized the code of DC3 to be maximally fast, our implementation of DC3 still runs about 78× faster than the state-of-the-art differentiable QP solver `qpth`, and only about 9× slower than the classical optimizer `OSQP`, which is specifically optimized for convex QPs. Furthermore, this assumes OSQP is fully parallelized – in this case, across 833 CPUs – whereas standard, non-parallel implementations of OSQP would be orders of magnitude slower. By contrast, DC3 is easily parallelized within a single GPU using standard deep learning frameworks.

Table 6.2: Results on QP task for 100 variables and 50 inequality constraints, as the number of equality constraints varies as $10, 30, 50, 70, 90$. We compare the performance of DC3 and other algorithms according to objective value and maximum equality/inequality constraint violations averaged across test instances. (Standard deviations across 5 runs are shown in parentheses.) We find that neural network baselines give significantly inferior performance (shown in red) across all problems.

		10	30	50	70	90
Optimizers (OSQP, qpth)	Obj. val.	-27.26 (0.00)	-23.13 (0.00)	-15.05 (0.00)	-14.80 (0.00)	-4.79 (0.00)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
DC3	Obj. val.	-25.79 (0.02)	-20.29 (0.20)	-13.46 (0.01)	-13.73 (0.02)	-4.76 (0.00)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
DC3, $\neq$	Obj. val.	-22.59 (0.18)	-20.40 (0.03)	-12.58 (0.04)	-13.36 (0.02)	-5.27 (0.00)
	Max eq.	0.12 (0.01)	0.24 (0.00)	0.35 (0.00)	0.47 (0.00)	0.59 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
DC3, $\not\leq$ train	Obj. val.	-25.75 (0.04)	-20.14 (0.15)	-1.39 (0.97)	-13.71 (0.04)	-4.75 (0.00)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.02 (0.02)	0.00 (0.00)	0.00 (0.00)
DC3, $\not\leq$ train/test	Obj. val.	-25.75 (0.04)	-20.14 (0.15)	-1.23 (1.21)	-13.71 (0.04)	-4.75 (0.00)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.09 (0.13)	0.00 (0.00)	0.00 (0.00)
DC3, no soft loss	Obj. val.	-66.79 (0.01)	-40.63 (0.02)	-21.84 (0.00)	-15.56 (0.02)	-4.76 (0.01)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	122.52 (0.22)	50.25 (0.14)	23.83 (0.11)	5.85 (0.18)	0.00 (0.00)
NN	Obj. val.	-22.65 (0.12)	-20.43 (0.03)	-12.57 (0.01)	-13.35 (0.03)	-5.29 (0.02)
	Max eq.	0.11 (0.01)	0.24 (0.00)	0.35 (0.00)	0.47 (0.00)	0.59 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
NN, $\leq$ test	Obj. val.	-22.65 (0.12)	-20.43 (0.03)	-12.57 (0.01)	-13.35 (0.03)	-5.29 (0.02)
	Max eq.	0.11 (0.01)	0.24 (0.00)	0.35 (0.00)	0.47 (0.00)	0.59 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Eq. NN	Obj. val.	-27.20 (0.02)	-22.74 (0.14)	-9.16 (0.75)	-14.64 (0.01)	-4.74 (0.01)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.43 (0.02)	1.37 (0.09)	8.83 (0.72)	0.86 (0.05)	0.00 (0.00)
Eq. NN, $\leq$ test	Obj. val.	-27.20 (0.02)	-22.76 (0.13)	-14.68 (0.05)	-14.65 (0.01)	-4.74 (0.01)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.42 (0.02)	1.27 (0.09)	0.89 (0.05)	0.85 (0.05)	0.00 (0.00)

Table 6.3: Results on QP task for 100 variables and 50 equality constraints. The number of *inequality* constraints varies as $10, 30, 50, 70, 90$. Content and interpretation as in Table 6.2.

		10	30	50	70	90
Optimizers (OSQP, qpth)	Obj. val.	-17.33 (0.00)	-16.33 (0.00)	-15.05 (0.00)	-14.61 (0.00)	-14.26 (0.00)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
DC3	Obj. val.	-15.18 (0.31)	-13.90 (0.20)	-13.46 (0.01)	-10.52 (0.93)	-11.77 (0.07)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
DC3, $\neq$	Obj. val.	-15.66 (0.05)	-14.10 (0.04)	-12.58 (0.04)	-12.10 (0.03)	-11.72 (0.03)
	Max eq.	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
DC3, $\not\leq$ train	Obj. val.	-15.60 (0.30)	-13.83 (0.15)	-1.39 (0.97)	-11.14 (0.05)	-11.76 (0.06)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.02 (0.02)	0.00 (0.00)	0.00 (0.00)
DC3, $\not\leq$ train/test	Obj. val.	-15.60 (0.30)	-13.83 (0.15)	-1.23 (1.21)	-11.14 (0.05)	-11.76 (0.06)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.09 (0.13)	0.00 (0.00)	0.00 (0.00)
DC3, no soft loss	Obj. val.	-21.78 (0.01)	-21.72 (0.03)	-21.84 (0.00)	-21.82 (0.01)	-21.83 (0.00)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	23.85 (0.52)	23.54 (0.23)	23.83 (0.11)	23.73 (0.09)	23.69 (0.06)
NN	Obj. val.	-15.66 (0.03)	-14.10 (0.05)	-12.57 (0.01)	-12.12 (0.03)	-11.70 (0.02)
	Max eq.	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
NN, $\leq$ test	Obj. val.	-15.66 (0.03)	-14.10 (0.05)	-12.57 (0.01)	-12.12 (0.03)	-11.70 (0.02)
	Max eq.	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)	0.35 (0.00)
	Max ineq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Eq. NN	Obj. val.	-17.07 (0.20)	-15.45 (0.09)	-9.16 (0.75)	-14.20 (0.06)	-14.10 (0.10)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	1.65 (0.22)	1.79 (0.10)	8.83 (0.72)	2.26 (0.04)	1.28 (0.05)
Eq. NN, $\leq$ test	Obj. val.	-17.06 (0.20)	-15.65 (0.09)	-14.68 (0.05)	-14.31 (0.06)	-14.10 (0.10)
	Max eq.	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	Max ineq.	1.53 (0.19)	1.58 (0.08)	0.89 (0.05)	1.92 (0.04)	1.25 (0.04)

Table 6.4: Ranges over which different hyperparameters were tuned for each method for the QP and simple nonconvex tasks, with central values for the coordinate search in *italics*, and the final parameter values in **bold**. In order to minimize the amount of tuning, we employ the hyperparameters obtained for DC3 to "DC3, $\not\leq$ train" and "DC3, no soft loss" as well (rather than tuning the latter two methods separately). For this set of experiments, we set the learning rate of the gradient-based correction procedure ρ_x to 10^{-7} for all methods, as most methods went unstable for higher correction learning rates.

	DC3 DC3, $\not\leq$ train DC3, no soft loss	DC3, $\neq$	NN	Eq. NN
Lr	10^{-2}, *10^{-3}*, **10^{-4}**	10^{-2}, *10^{-3}*, **10^{-4}**	10^{-2}, *10^{-3}*, **10^{-4}**	10^{-2}, *10^{-3}*, **10^{-4}**
$\lambda_g + \lambda_h$	1,***10***,100	1,*10*,**100**	1,*10*,**100**	–
$\frac{\lambda_h}{\lambda_g+\lambda_h}$	***0.5***, 0.1, 0.01	***0.5***, 0.9, 0.99	***0.5***, 0.9, 0.99	–
t_{test}	5, **10**, 50, 100, *1000*	5, **10**, 50, 100, *1000*	5, **10**, 50, 100, *1000*	5, **10**, 50, 100, *1000*
t_{train}	1, 2, *5*, **10**, 100	1, 2, *5*, **10**, 100	–	–

Table 6.5: Results on our simple nonconvex task for 100 variables, 50 equality constraints, and 50 inequality constraints, with details as in Table 6.1. Since this problem is nonconvex, we use `IPOPT` as the classical optimizer. DC3 is differentiable and about 9× faster than `IPOPT`, giving a near-optimal objective value and constraint satisfaction, in contrast to baseline deep learning-based methods which result in significant constraint violations.

	Obj. value	Max eq.	Mean eq.	Max ineq.	Mean ineq.	Time (s)
Optimizer	-11.59 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.121 (0.000)
DC3	-10.66 (0.03)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.013 (0.000)
DC3, $\neq$	-10.04 (0.02)	0.35 (0.00)	0.13 (0.00)	0.00 (0.00)	0.00 (0.00)	0.009 (0.000)
DC3, $\not\leq$ train	-0.29 (0.67)	0.00 (0.00)	0.00 (0.00)	0.01 (0.01)	0.00 (0.00)	0.010 (0.004)
DC3, $\not\leq$ train/test	-0.27 (0.67)	0.00 (0.00)	0.00 (0.00)	0.03 (0.03)	0.00 (0.00)	0.001 (0.000)
DC3, no soft loss	-13.81 (0.00)	0.00 (0.00)	0.00 (0.00)	15.21 (0.04)	2.33 (0.01)	0.013 (0.000)
NN	-10.02 (0.01)	0.35 (0.00)	0.13 (0.00)	0.00 (0.00)	0.00 (0.00)	0.001 (0.000)
NN, $\leq$ test	-10.02 (0.01)	0.35 (0.00)	0.13 (0.00)	0.00 (0.00)	0.00 (0.00)	0.009 (0.000)
Eq. NN	-3.88 (0.56)	0.00 (0.00)	0.00 (0.00)	6.87 (0.43)	0.72 (0.05)	0.001 (0.000)
Eq. NN, $\leq$ test	-10.99 (0.03)	0.00 (0.00)	0.00 (0.00)	0.87 (0.04)	0.06 (0.00)	0.013 (0.000)

6.4.2 Simple non-convex optimization

We now consider a simple non-convex adaptation of the quadratic program above:

$$\underset{y\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}y^TQy + p^T\sin(y), \ \text{ s.t. } Ay = x, \ Gy \leq h,$$

where $\sin(y)$ denotes the componentwise application of the sine function to the vector y, and where all constants and variables are defined as in (6.5). We consider instances of this problem where all parameters are drawn randomly as in our preceding experiments in the convex setting.

In Table 6.5, we compare the performance of DC3 and other deep learning-based methods against the classical non-convex optimizer `IPOPT`.[4] All hyperparameters in these experiments were maintained at the settings chosen for the convex QP task in Section 6.4.1,

[4]We initialize the optimizer using the feasible point $y = A^+x$ noted in Equation (6.6).

since these tasks are similar by design. We find that DC3 achieves good objective values (8.02% per-instance optimality gap), while maintaining feasibility. By contrast, all other deep learning-based methods that we consider violate constraints significantly. DC3 also runs about 10× faster than `IPOPT`, even assuming `IPOPT` is fully parallelized. (Even on the CPU, DC3 takes 0.030 ± 0.000 seconds, about 4× faster than `IPOPT`.) Note that the DC3 algorithm is essentially the same between the convex QP and this non-convex task, since only the objective function is altered.

6.4.3 AC optimal power flow

We now show how DC3 can be applied to the problem of AC optimal power flow (ACOPF), as defined in Section 2.3.2. In particular, as the amount of renewable energy on the power grid grows, ACOPF must be solved more and more frequently to account for the variability of these renewable sources, and at larger scale to account for an increasing number of distributed devices [Rol+22]. However, given that ACOPF is a non-convex optimization problem, classical optimizers scale poorly on it. While specialized approaches to this problem have started to emerge, including using machine learning (see, e.g., [ZB20; Pan+20] for a discussion), we here assess the ability of our more general framework to approximate this problem.

Mathematically, we consider the ACOPF formulation previously defined in Equation (6.7), with a quadratic generator cost function $f_c(p_g) = p_g^T \operatorname{diag}(c_q) p_g + c_a^T p_g$, where $c_q, c_a \in \mathbb{R}^b_{\geq 0}$ are the quadratic and linear cost parameters, respectively, for power generation at each node. The resultant ACOPF optimization problem is as follows:

$$\begin{aligned} \underset{p_g \in \mathbb{R}^b,\, q_g \in \mathbb{R}^b,\, v \in \mathbb{C}^b}{\text{minimize}} \quad & p_g^T \operatorname{diag}(c_q) p_g + c_a^T p_g \\ \text{subject to} \quad & p_g^{\min} \leq p_g \leq p_g^{\max}, \quad q_g^{\min} \leq q_g \leq q_g^{\max}, \quad v^{\min} \leq |v| \leq v^{\max}, \\ & (p_g - p_d) + (q_g - q_d) i = \operatorname{diag}(v) \overline{W} \overline{v}. \end{aligned} \tag{6.7}$$

More details about how we apply DC3 to the problem of ACOPF are given in Appendix B.

We assess our method on a 57-bus power system test case available via the `MATPOWER` package. We conduct 5 runs over 1,200 input datapoints (with a train/test/validation ratio of 10:1:1). Optimality, feasibility, and timing results are reported in Table 6.6. Hyperparameter values are shown in Table 6.7.

We find that DC3 achieves comparable objective values to the optimizer, and preserves feasibility with respect to both equality and inequality constraints. Once again, for every baseline deep learning algorithm, feasibility is violated significantly for either equality or inequality constraints. Ablations of DC3 also suffer from constraint violations, though the effect is less pronounced than for the convex QP and simple non-convex settings, especially for ablation of the correction (perhaps because the inequality constraints here are easier to satisfy than the equality constraints). We also see that DC3 runs about 10× faster than the `PYPOWER` optimizer, even when `PYPOWER` is fully parallelized. (Even when running on the CPU, DC3 takes 0.906 ± 0.003 seconds, slightly faster than `PYPOWER`.)

Out of 100 test instances, there were 3 on which DC3 output lower-than-optimal objective values of up to a few percent (-0.30%, -1.85%, -5.51%), reflecting slight constraint

Table 6.6: Results on ACOPF over 100 test instances. We compare the performance of DC3 and other algorithms according to the metrics described in Table 6.1. We find again that baseline methods violate feasibility (as shown in red), while DC3 gives a feasible and near-optimal output about 10× faster than the `PYPOWER` optimizer, even assuming that `PYPOWER` is fully parallelized.

	Obj. value	Max eq.	Mean eq.	Max ineq.	Mean ineq.	Time (s)
Optimizer	3.81 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.949 (0.002)
DC3	3.82 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.089 (0.000)
DC3, $\neq$	3.67 (0.01)	0.14 (0.01)	0.02 (0.00)	0.00 (0.00)	0.00 (0.00)	0.040 (0.000)
DC3, $\not\leq$ train	3.82 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.089 (0.000)
DC3, $\not\leq$ train/test	3.82 (0.00)	0.00 (0.00)	0.00 (0.00)	0.01 (0.00)	0.00 (0.00)	0.039 (0.000)
DC3, no soft loss	3.11 (0.05)	2.60 (0.35)	0.07 (0.00)	2.33 (0.33)	0.03 (0.01)	0.088 (0.000)
NN	3.69 (0.02)	0.19 (0.01)	0.03 (0.00)	0.00 (0.00)	0.00 (0.00)	0.001 (0.000)
NN, $\leq$ test	3.69 (0.02)	0.16 (0.00)	0.02 (0.00)	0.00 (0.00)	0.00 (0.00)	0.040 (0.000)
Eq. NN	3.81 (0.00)	0.00 (0.00)	0.00 (0.00)	0.15 (0.01)	0.00 (0.00)	0.039 (0.000)
Eq. NN, $\leq$ test	3.81 (0.00)	0.00 (0.00)	0.00 (0.00)	0.15 (0.01)	0.00 (0.00)	0.078 (0.000)

Table 6.7: Ranges over which different hyperparameters were tuned for each method for the ACOPF task, with central values for the coordinate search in *italics*, and the final parameter values in **bold**. In order to minimize the amount of tuning, we employ the hyperparameters obtained for DC3 to "DC3, $\not\leq$ train" and "DC3, no soft loss" as well (rather than tuning the latter two methods separately).

	DC3 DC3, $\not\leq$ train DC3, no soft loss	DC3, $\neq$	NN	Eq. NN
Lr	10^{-2}, ***10*$^{-3}$**, 10^{-4}	10^{-2}, ***10*$^{-3}$**, 10^{-4}	10^{-2}, ***10*$^{-3}$**, 10^{-4}	10^{-2}, ***10*$^{-3}$**, 10^{-4}
$\lambda_g + \lambda_h$	1,***10***,100	1,*10*,**100**	1,*10*,**100**	–
$\frac{\lambda_h}{\lambda_g+\lambda_h}$	***0.5***, 0.1, 0.01	***0.5***, 0.9, 0.99	***0.5***, 0.9, 0.99	–
t_{test}	**5**, *10*, 50, 100	**5**, *10*, 50, 100	**5**, *10*, 50, 100	**5**, *10*, 50, 100, 200
t_{train}	1, 2, ***5***, 10, 100	1, 2, ***5***, 10, 100	–	–
Lr, ρ_x	10^{-3}, ***10*$^{-4}$**, 10^{-5}	10^{-4}, ***10*$^{-5}$**, 10^{-6}	**10^{-5}**, *10*$^{-6}$, 10^{-7}	**10^{-5}**, *10*$^{-6}$, 10^{-7}

violations. Over the other 97 instances, the per-instance optimality gap compared to the classical optimizer was 0.22%.

6.5 Conclusion

We have described a method, DC3, for fast approximate solutions to optimization problems with hard constraints. Our approach includes a neural network that outputs a partial set of variables, a differentiable completion procedure that fills in remaining variables according to equality constraints, and a differentiable correction procedure that fixes inequality violations. We find that DC3 yields solutions of significantly better feasibility and objective value than other approximate deep learning-based solvers on convex and non-convex optimization tasks.

We note that, while DC3 provides a general framework for tackling constrained optimization, depending on the setting, the expensiveness of both the completion and correction procedures may vary (e.g., implicit solutions may be more time-consuming, or gradient descent

may converge more or less easily). We believe that, while our method as stated is broadly applicable, it will be possible in future work to design further improvements tailored to specific problem instances, for example by designing problem-dependent correction procedures.

Chapter 7

Enforcing Robust Control Guarantees within Neural Network Policies

When designing controllers for safety-critical systems, practitioners often face a challenging tradeoff between robustness and performance. While robust control methods provide rigorous guarantees on system stability under certain worst-case disturbances, they often yield simple controllers that perform poorly in the average (non-worst) case. In contrast, nonlinear control methods trained using deep learning have achieved state-of-the-art performance on many control tasks, but often lack robustness guarantees. In this chapter, we propose a technique that combines the strengths of these two approaches: constructing a generic nonlinear control policy class, parameterized by neural networks, that nonetheless enforces the same provable robustness criteria as robust control. Specifically, our approach entails integrating custom convex-optimization-based projection layers into a neural network-based policy. We demonstrate the power of this approach on several synthetic domains (including a synthetic microgrid task), improving in average-case performance over existing robust control methods and in worst-case stability over (non-robust) deep reinforcement learning methods.

The work in this chapter has previously been published in:[1]

> Priya L. Donti, Melrose Roderick, Mahyar Fazlyab, and J. Zico Kolter. "Enforcing Robust Control Guarantees within Neural Network Policies." *International Conference on Learning Representations*. 2021.

[1] Code for all experiments is available at: https://github.com/locuslab/robust-nn-control.

7.1 Introduction

The field of robust control, dating back many decades, has been able to provide rigorous guarantees on when controllers will succeed or fail in controlling a system of interest. In particular, if the uncertainties in the underlying dynamics can be bounded in specific ways, these techniques can produce controllers that are provably robust even under worst-case conditions. However, as the resulting policies tend to be simple (i.e., often linear), this can limit their performance in typical (rather than worst-case) scenarios. In contrast, recent high-profile advances in deep reinforcement learning have yielded state-of-the-art performance on many control tasks, due to their ability to capture complex, nonlinear policies. However, due to a lack of robustness guarantees, these techniques have still found limited application in safety-critical domains where an incorrect action (either during training or at runtime) can substantially impact the controlled system.

In this chapter, we propose a method that combines the guarantees of robust control with the flexibility of deep reinforcement learning (RL). Specifically, we consider the setting of nonlinear, time-varying systems with unknown dynamics, but where (as common in robust control) the uncertainty on these dynamics can be bounded in ways amenable to obtaining provable performance guarantees. Building upon specifications provided by traditional robust control methods in these settings, we construct a new class of nonlinear policies that are parameterized by neural networks, but that are nonetheless *provably robust*. In particular, we *project* the outputs of a nominal (deep neural network-based) controller onto a space of stabilizing actions characterized by the robust control specifications. The resulting nonlinear control policies are trainable using standard approaches in deep RL, yet are *guaranteed* to be stable under the same worst-case conditions as the original robust controller.

We describe our proposed deep nonlinear control policy class and derive efficient, differentiable projections for this class under various models of system uncertainty common in robust control. We demonstrate our approach on several different domains, including synthetic linear differential inclusion (LDI) settings, the cart-pole task, a quadrotor domain, and a microgrid domain. Although these domains are simple by modern RL standards, we show that purely RL-based methods often produce unstable policies in the presence of system disturbances, both during and after training. In contrast, we show that our method remains stable even when worst-case disturbances are present, while improving upon the performance of traditional robust control methods.

7.2 Related work

Robust control. Robust control is concerned with the design of feedback controllers for dynamical systems with modeling uncertainties and/or external disturbances [ZD98; BB08], specifically controllers with guaranteed performance under worst-case conditions. Many classes of robust control problems in both the time and frequency domains can be formulated using linear matrix inequalities (LMIs) [Boy+94; KBM96]; for reasonably-sized problems, these LMIs can be solved using off-the-shelf numerical solvers based on interior-point or first-order (gradient-based) methods. However, providing stability guarantees often

requires the use of simple (linear) controllers, which greatly limits average-case performance. Our work seeks to improve performance via *nonlinear* controllers that nonetheless retain the same stability guarantees.

Reinforcement learning (RL). In contrast, RL (and specifically, deep RL) is not restricted to simple controllers or problems with uncertainty bounds on the dynamics. Instead, deep RL seeks to learn an optimal control policy, represented by a neural network, by directly interacting with an unknown environment. These methods have shown impressive results in a variety of complex control tasks (e.g., [Mni+15; Akk+19]); see [Buş+18] for a survey. However, due to its lack of safety guarantees, deep RL has been predominantly applied to simulated environments or highly-controlled real-world problems, where system failures are either not costly or not possible.

Efforts to address the lack of safety and stability in RL fall into several main categories. The first tries to combine control-theoretic ideas, predominantly robust control, with the nonlinear control policy benefits of RL (e.g., [MD05; AKLH06; FAR09; LLW13; WL13; LWH14; FB17; Pin+17; JL20; CRG19; Han+19; ZHB20; CHW22; Rut+22; JMP21]). For example, RL has been used to address stochastic stability in H_∞ control synthesis settings by jointly learning Lyapunov functions and policies in these settings [Han+19]. As another example, RL has been used to address H_∞ control for continuous-time systems via min-max differential games, in which the controller and disturbance are the "minimizer" and "maximizer" [MD05]. We view our approach as thematically aligned with this previous work, though our method is able to capture not only H_∞ settings, but also a much broader class of robust control settings. A more recent direction has also involved mixing between the outputs of a trusted robust control algorithm and an "untrusted" algorithm (e.g., a standard RL algorithm) [CHW22; Rut+22], an approach we similarly view as being thematically aligned with our work.

Another category of methods addressing this challenge is safe RL, which aims to learn control policies while maintaining some notion of safety during or after learning. Typically, these methods attempt to restrict the RL algorithm to a safe region of the state space by making strong assumptions about the smoothness of the underlying dynamics, e.g., that the dynamics can be modeled as a Gaussian process (GP) [TBK16; Aka+14] or are Lipschitz continuous [Ber+17; Wac+18]. This framework is in theory more general than our approach, which requires using stringent uncertainty bounds (e.g., state-control norm bounds) from robust control. However, there are two key benefits to our approach. First, norm bounds or polytopic uncertainty can accommodate sharp discontinuities in the continuous-time dynamics. Second, convex projections (as used in our method) scale polynomially with the state-action size, whereas GPs in particular scale exponentially (and are therefore difficult to extend to high-dimensional problems).

A third category of methods uses Constrained Markov Decision Processes (C-MDPs). These methods seek to maximize a discounted reward while bounding some discounted cost function [Alt99; Ach+17; TD18; Yan+20]. While these methods do not require knowledge of the cost functions a-priori, they only guarantee the cost constraints hold during test time. Additionally, using C-MDPs can yield other complications, such as optimal policies being stochastic and the constraints only holding for a subset of states.

7.3 Background on LQR and robust control specifications

In this work, our aim is to control nonlinear (continuous-time) dynamical systems of the form

$$\dot{x}(t) \in A(t)x(t) + B(t)u(t) + G(t)w(t), \tag{7.1}$$

where $x(t) \in \mathbb{R}^s$ denotes the state at time t; $u(t) \in \mathbb{R}^a$ is the control input; $w(t) \in \mathbb{R}^d$ captures both external (possibly stochastic) disturbances and any modeling discrepancies; $\dot{x}(t)$ denotes the time derivative of the state x at time t; and $A(t) \in \mathbb{R}^{s\times s}, B(t) \in \mathbb{R}^{s\times a}$, $G(t) \in \mathbb{R}^{s\times d}$. This class of models is referred to as linear differential inclusions (LDIs); however, we note that despite the name, this class does indeed characterize *nonlinear* systems, as, e.g., $w(t)$ can depend arbitrarily on $x(t)$ and $u(t)$ (though we omit this dependence in the notation for brevity). Within this class of models, it is often possible to construct robust control specifications certifying system stability. Given such specifications, our proposal is to learn nonlinear (deep neural network-based) policies that *provably* satisfy these specifications while optimizing some objective of interest. We start by giving background on the robust control specifications and objectives considered in this work.

7.3.1 Robust control specifications

In the continuous-time, infinite-horizon settings we consider here, the goal of robust control is often to construct a time-invariant control policy $u(t) = \pi(x(t))$, alongside some certification that guarantees that the controlled system will be stable (i.e., that trajectories of the system will converge to an equilibrium state, usually $x = 0$ by convention; see [HC11] for a more formal definition). For many classes of systems,[2] this certification is typically in the form of a positive definite Lyapunov function $V : \mathbb{R}^s \to \mathbb{R}$, with $V(0) = 0$ and $V(x) > 0$ for all $x \neq 0$, such that the function is decreasing along trajectories – for instance,

$$\dot{V}(x(t)) \leq -\alpha V(x(t)) \tag{7.2}$$

for some design parameter $\alpha > 0$. (This particular condition implies *exponential stability* with a rate of convergence α.[3]) For certain classes of bounded dynamical systems, time-invariant linear control policies $u(t) = Kx(t)$, and quadratic Lyapunov functions $V(x) = x^T Px$, it is possible to construct such guarantees using semidefinite programming. For instance, consider the class of norm-bounded LDIs (NLDIs)

$$\dot{x} = Ax(t) + Bu(t) + Gw(t), \quad \|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2, \tag{7.3}$$

where $A \in \mathbb{R}^{s\times s}$, $B \in \mathbb{R}^{s\times a}$, $G \in \mathbb{R}^{s\times d}$, $C \in \mathbb{R}^{k\times s}$, and $D \in \mathbb{R}^{k\times a}$ are time-invariant and known, and the disturbance $w(t)$ is arbitrary (and unknown) but obeys the norm bounds

[2] In this work, we consider sub-classes of system (7.1) that may indeed be stochastic (e.g., due to a stochastic external disturbance $w(t)$), but that can be bounded so as to be amenable to deterministic stability analysis. However, other settings may require stochastic stability analysis; please see [Ast71].

[3] See, e.g., [HC11] for a more rigorous definition of (local and global) exponential stability. Condition (7.2) comes from *Lyapunov's Theorem*, which characterizes various notions of stability using Lyapunov functions.

above.[4] For these systems, it is possible to specify a set of stabilizing policies via a set of linear matrix inequalities (LMIs, [Boy+94]):

$$\begin{bmatrix} AS + SA^T + \mu GG^T + BY + Y^T B^T + \alpha S & SC^T + Y^T D^T \\ CS + DY & -\mu I \end{bmatrix} \preceq 0, \quad S \succ 0, \quad \mu > 0, \quad (7.4)$$

where $S \in \mathbb{R}^{s \times s}$ and $Y \in \mathbb{R}^{a \times s}$. For matrices S and Y satisfying (7.4), $K = YS^{-1}$ and $P = S^{-1}$ are then a stabilizing linear controller gain and Lyapunov matrix, respectively. While the LMI above is specific to NLDI systems, this general paradigm of constructing stability specifications using LMIs applies to many settings commonly considered in robust control (e.g., settings with norm-bounded disturbances or polytopic uncertainty, or H_∞ control settings). More details about these types of formulations are given in, e.g., Boyd, El Ghaoui, Feron, and Balakrishnan [Boy+94]; in addition, we provide the relevant LMI constraints for the settings we consider in this work in Appendix C.1.

7.3.2 LQR control objectives

In addition to designing for stability, it is often desirable to optimize some objective characterizing controller performance. While our method can optimize performance with respect to any arbitrary cost or reward function, to make comparisons with existing methods easier, for this work, we consider the well-known infinite-horizon "linear-quadratic regulator" (LQR) cost, defined as

$$\int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt, \quad (7.5)$$

for some $Q \in \mathbb{S}^{s \times s} \succeq 0$ and $R \in \mathbb{S}^{a \times a} \succ 0$. If the control policy is assumed to be time-invariant and linear as described above (i.e., $u(t) = Kx(t)$), minimizing the LQR cost subject to stability constraints can be cast as an SDP (see, e.g., [YZZ01]) and solved using off-the-shelf numerical solvers – a fact that we exploit in our work. For example, to obtain an optimal linear time-invariant controller for the NLDI systems described above, we can solve

$$\underset{S,Y}{\text{minimize}} \quad \text{tr}(QS) + \text{tr}(R^{1/2} Y S^{-1} Y^T R^{1/2}) \quad \text{s. t. Equation (7.4) holds.} \quad (7.6)$$

7.4 Enforcing robust control guarantees within neural networks

We now present the main contribution of this work: A class of *nonlinear* control policies, potentially parameterized by deep neural networks, that is guaranteed to obey the same stability conditions enforced by the robustness specifications described above. The key

[4] A slightly more complex formulation involves an additional term in the norm bound, i.e., $Cx(t) + Du(t) + Hw(t)$, which creates a quadratic inequality in w. The mechanics of obtaining robustness specifications in this setting are largely the same as presented here, though with some additional terms in the equations. As such, as is often done, we assume that $H = 0$ for simplicity.

insight of our approach is as follows: While it is difficult to derive specifications that globally characterize the stability of a generic nonlinear controller, if we are given *known* robustness specifications, we can create a sufficient condition for stability by simply enforcing that our policy satisfies these specifications at all t. For instance, given a known Lyapunov function, we can enforce exponential stability by ensuring that our policy sufficiently decreases this function (e.g., satisfies Equation (7.2)) at any given $x(t)$.

In the following sections, we present our nonlinear policy class, as well as our general framework for learning provably robust policies using this policy class. We then derive the instantiation of this framework for various settings of interest. In particular, this involves constructing (custom) differentiable projections that can be used to adjust the output of a nominal neural network to satisfy desired robustness criteria. For simplicity of notation, we will often suppress the t-dependence of x, u, and w, but we note that these are continuous-time quantities as before.

7.4.1 A provably robust nonlinear policy class

Given a dynamical system of the form (7.1) and a quadratic Lyapunov function $V(x) = x^T P x$, let

$$\mathcal{C}(x) := \{u \in \mathbb{R}^a \mid \dot{V}(x) \leq -\alpha V(x) \quad \forall \dot{x} \in A(t)x + B(t)u + G(t)w\} \tag{7.7}$$

denote a set of actions that, for a *fixed* state $x \in \mathbb{R}^s$, are guaranteed to satisfy the exponential stability condition (7.2) (even under worst-case realizations of the disturbance w). We note that this "safe" set is non-empty if P satisfies the relevant LMI constraints (e.g., system (7.4) for NLDIs) characterizing robust linear time-invariant controllers, as there is then some K corresponding to P such that $Kx \in \mathcal{C}(x)$ for all states x.

Using this set of actions, we then construct a robust nonlinear policy class that *projects* the output of some neural network onto this set. More formally, consider an arbitrary nonlinear (neural network-based) policy class $\hat{\pi}_\theta : \mathbb{R}^s \to \mathbb{R}^a$ parameterized by θ, and let $\mathcal{P}_{(\cdot)}$ denote the projection operator for some set $(\cdot)$. We then define our robust policy class as $\pi_\theta : \mathbb{R}^s \to \mathbb{R}^a$, where

$$\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)). \tag{7.8}$$

We note that this policy class is differentiable if the projections can be implemented in a differentiable manner (e.g., using convex optimization layers [Agr+19], though we construct efficient custom solvers for our purposes). Importantly, as all policies in this class satisfy the stability condition (7.2) for all states x and at all times t, these policies are *certifiably* robust under the same conditions as the original (linear) controller for which the Lyapunov function $V(x)$ was constructed.

Given this policy class and some performance objective ℓ (e.g., LQR cost), our goal is to then find parameters θ such that the corresponding policy optimizes this objective – i.e., to solve

$$\underset{\theta}{\text{minimize}} \quad \int_0^\infty \ell\,(\,x,\,\pi_\theta(x)\,)\,dt \quad \text{s.t. } \dot{x} \in A(t)x + B(t)\pi_\theta(x) + G(t)w. \tag{7.9}$$

Algorithm 3 Learning provably robust controllers with deep RL

input performance objective ℓ // *e.g., LQR cost*
input stability requirement // *e.g.,* $\dot{V}(x) \leq -\alpha V(x)$
input policy optimizer $\mathcal{A}$ // *e.g., a planning or RL algorithm*
compute P, K satisfying LMI constraints // *e.g., by optimizing* (7.6)
construct specifications $\mathcal{C}(x)$ using P // *as defined in Equation* (7.7)
construct robust policy class π_θ using $\mathcal{C}$ // *as defined in Equation* (7.8)
train π_θ via $\mathcal{A}$ to optimize Equation (7.9)
return π_θ

Since π_θ is differentiable, we can solve this problem via a variety of approaches, e.g., a model-based planning algorithm if the true dynamics are known, or virtually any (deep) RL algorithm if the dynamics are unknown.[5]

This general procedure for constructing stabilizing controllers is summarized in Algorithm 3. While seemingly simple, this formulation presents a powerful paradigm: by simply transforming the output of a neural network, we can employ an expressive policy class to optimize an objective of interest while *ensuring* the resultant policy will stabilize the system during both training and testing.

We instantiate our framework by constructing "safe" sets $\mathcal{C}(x)$ and their associated (differentiable) projections $\mathcal{P}_{\mathcal{C}(x)}$ for three settings of interest: NLDIs, polytopic linear differential inclusions (PLDIs), and H_∞ control settings. As an example, we describe this procedure below for NLDIs, and refer readers to Appendix C.2 for corresponding formulations for the additional settings we consider.

7.4.2 Example: NLDIs

In order to apply our framework to the NLDI setting (7.3), we first compute a quadratic Lyapunov function $V(x) = x^T P x$ by solving the optimization problem (7.6) for the given system via semidefinite programming. We then use the resultant Lyapunov function to compute the system-specific "safe" set $\mathcal{C}(x)$, and then create a fast, custom differentiable solver to project onto this set.

7.4.2.1 Computing sets of stabilizing actions

Given P, we compute $\mathcal{C}_{\text{NLDI}}(x)$ as the set of actions $u \in \mathbb{R}^a$ that, for each state $x \in \mathbb{R}^s$, satisfy the stability condition (7.2) at that state *under even a worst-case realization of the dynamics* (i.e., in this case, even under a worst-case disturbance w). The form of the resultant set is given below.

Theorem 1. *Consider the NLDI system* (7.3)*, some stability parameter* $\alpha > 0$*, and a Lyapunov function* $V(x) = x^T P x$ *with* P *satisfying Equation* (7.4)*. Assuming* P *exists,*

[5] While this problem is infinite-horizon and continuous in time, in practice, one would optimize it in discrete time over a large finite time horizon.

define

$$\mathcal{C}_{NLDI}(x) := \left\{ u \in \mathbb{R}^a \mid \|Cx + Du\|_2 \leq \frac{-x^T PB}{\|G^T Px\|_2} u - \frac{x^T (2PA + \alpha P)x}{2\|G^T Px\|_2} \right\}$$

for all states $x \in \mathbb{R}^s$. For all x, $\mathcal{C}_{NLDI}(x)$ is a non-empty set of actions that satisfy the exponential stability condition (7.2). Further, $\mathcal{C}_{NLDI}(x)$ is a convex set in u.

Proof. We seek to find a set of actions such that the condition (7.2) is satisfied along *all* possible trajectories of (7.3). A set of actions satisfying this condition at a given x is given by

$$\mathcal{C}_{\text{NLDI}}(x) := \left\{ u \in \mathbb{R}^a \mid \sup_{w:\|w\|_2 \leq \|Cx+Du\|_2} \dot{V}(x) \leq -\alpha V(x) \right\}.$$

Let $\mathcal{S} := \{w : \|w\|_2 \leq \|Cx + Du\|_2\}$. We can then rewrite the left side of the above inequality as

$$\begin{aligned} \sup_{w \in \mathcal{S}} \dot{V}(x) &= \sup_{w \in \mathcal{S}} \dot{x}^T Px + x^T P\dot{x} \;=\; 2x^T P(Ax + Bu) + \sup_{w \in \mathcal{S}} 2x^T PGw \\ &= 2x^T P(Ax + Bu) + 2\|G^T Px\|_2 \|Cx + Du\|_2, \end{aligned}$$

by the definition of the NLDI dynamics and the closed-form minimization of a linear term over an L_2 ball. Rearranging yields an inequality of the desired form. We note that by definition of the specifications (7.4), there is some K corresponding to P such that the policy $u = Kx$ satisfies the exponential stability condition (7.2); thus, $Kx \in \mathcal{C}_{\text{NLDI}}$, and $\mathcal{C}_{\text{NLDI}}$ is non-empty. Further, as the above inequality represents a second-order cone constraint in u, this set is convex in u. □

We further consider the special case where $D = 0$, i.e., the norm bound on w does not depend on the control action. This form of NLDI arises in many common settings (e.g., where w characterizes linearization error in a nonlinear system but the dynamics depend only linearly on the action), and is one for which we can compute the relevant projection in closed form (as described shortly).

Corollary 1.1. *Consider the NLDI system (7.3) with $D = 0$, some stability parameter $\alpha > 0$, and Lyapunov function $V(x) = x^T Px$ with P satisfying Equation (7.4). Assuming P exists, define*

$$\mathcal{C}_{NLDI\text{-}0}(x) := \left\{ u \in \mathbb{R}^a \mid 2x^T PBu \leq -x^T (2PA + \alpha P)x - 2\|G^T Px\|_2 \|Cx\|_2 \right\}$$

for all states $x \in \mathbb{R}^s$. For all x, $\mathcal{C}_{NLDI\text{-}0}(x)$ is a non-empty set of actions that satisfy the exponential stability condition (7.2). Further, $\mathcal{C}_{NLDI\text{-}0}(x)$ is a convex set in u.

Proof. The result follows by setting $D = 0$ in Theorem 1 and rearranging terms. As the above inequality represents a linear constraint in u, this set is convex in u. □

7.4.2.2 Deriving efficient, differentiable projections

For the general NLDI setting (7.3), we note that the relevant projection $\mathcal{P}_{\mathcal{C}_{\text{NLDI}}(x)}$ (see Theorem 1) represents a projection onto a second-order cone constraint. As this projection does not necessarily have a closed form, we must implement it using a differentiable optimization solver (e.g., [Agr+19]). For computational efficiency purposes, we implement a custom solver that employs an accelerated projected dual gradient method for the forward pass, and employs implicit differentiation through the fixed point equations of this solution method to compute relevant gradients for the backward pass. Derivations and additional details are provided in Appendix C.3.

In the case where $D = 0$ (see Corollary 1.1), we note that the projection operation $\mathcal{P}_{\mathcal{C}_{\text{NLDI-0}}(x)}$ does have a closed form, and can in fact be implemented via a single ReLU operation. Specifically, defining $\eta^T := 2x^T PB$ and $\zeta := -x^T(2PA + \alpha P)x - 2\|G^T Px\|_2\|Cx\|_2$, we see that

$$\mathcal{P}_{\mathcal{C}_{\text{NLDI-0}}(x)}\left(\hat{\pi}(x)\right) = \begin{cases} \hat{\pi}(x) & \text{if } \eta^T\hat{\pi}(x) \le \zeta \\ \hat{\pi}(x) - \frac{\eta^T\hat{\pi}(x)-\zeta}{\eta^T\eta}\eta & \text{otherwise} \end{cases} = \hat{\pi}(x) - \text{ReLU}\left(\frac{\eta^T\hat{\pi}(x)-\zeta}{\eta^T\eta}\right)\eta. \tag{7.10}$$

7.5 Experiments

Having instantiated our general framework, we demonstrate the power of our approach on a variety of simulated control domains. In particular, we evaluate performance on the following metrics:

- **Average-case performance:** How well does the method optimize the performance objective (i.e., LQR cost) under average (non-worst case) dynamics?
- **Worst-case stability:** Does the method remain stable even when subjected to adversarial (worst-case) dynamics?

In all cases, we show that our method is able to ouperform traditional robust controllers under average conditions, while still guaranteeing stability under worst-case conditions.

7.5.1 Description of dynamics settings

We evaluate our approach on five NLDI settings: two synthetic NLDI domains, the cart-pole task, a quadrotor domain, and a microgrid domain. (Additional experiments for PLDI and H_∞ control settings are described in Appendix C.9.) For each setting, we choose a time discretization based on the speed at which the system evolves, and run each episode for 200 steps over this discretization. In all cases except the microgrid setting, we use a randomly generated LQR objective where the matrices $Q^{1/2}$ and $R^{1/2}$ are drawn i.i.d. from a standard normal distribution.

Synthetic NLDI settings. We generate NLDIs of the form (7.3) with $s = 5$, $a = 3$, and $d = k = 2$ by generating matrices A, B, G, C and D i.i.d. from normal distributions, and

producing the disturbance $w(t)$ using a randomly-initialized neural network (with its output scaled to satisfy the norm-bound on the disturbance). We investigate settings both where $D \neq 0$ and where $D = 0$. In both cases, episodes are run for 2 seconds at a discretization of 0.01 seconds.

Cart-pole. We aim to balance an inverted pendulum resting on top of a cart by exerting horizontal forces on the cart. For our experiments, we linearize this system as an NLDI with $D \neq 0$ (see Appendix C.4), and add a small additional randomized disturbance satisfying the NLDI bounds. Episodes are run for 10 seconds at a discretization of 0.05 seconds.

Planar quadrotor. We aim to stabilize a quadcopter in the two-dimensional plane by controlling the amount of force provided by the quadcopter's right and left thrusters. We linearize this system as an NLDI with $D = 0$ (see Appendix C.5), and add a small disturbance as in the cart-pole setting. Episodes are run for 4 seconds at a discretization of 0.02 seconds.

Microgrid. We aim to stabilize a grid-connected microgrid by controlling a storage device and a solar inverter. We augment the system given in [LBR16] with LQR matrices and NLDI bounds (see Appendix C.6). Episodes are run for 2 seconds at a discretization of 0.01 seconds.

7.5.2 Experimental setup

We demonstrate our approach by constructing a robust policy class (7.8) for each of these settings, and optimizing this policy class via different approaches. Specifically, we construct a nominal nonlinear control policy class as $\hat{\pi}_\theta(x) = Kx + \tilde{\pi}_\theta(x)$, where K is obtained via robust LQR optimization (7.6), and where $\tilde{\pi}_\theta(x)$ is a feedforward neural network. To construct the projections $\mathcal{P}_\mathcal{C}$, we employ the value of P obtained when solving for K. For demonstration purposes, we optimize our robust policy class $\pi_\theta(x) = \mathcal{P}_\mathcal{C}(\hat{\pi}_\theta(x))$ using two different methods:

- **Robust MBP** (ours)**:** A model-based planner with access to the true dynamics.
- **Robust PPO** (ours)**:** An RL approach based on PPO [Sch+17b] that does not assume known dynamics (beyond the bounds used to construct the robust policy class).

Robust MBP is optimized using gradient descent for 1,000 updates, where each update samples 20 roll-outs. Robust PPO is trained for 50,000 updates, where each update samples 8 roll-outs; we choose the model that performs best on a holdout set of initial conditions during training. While we use PPO for our demonstration, our approach is agnostic to the particular method of training, and can be deployed with many different (deep) RL paradigms.

We compare our robust neural network-based method against the following baselines:

- **Robust LQR:** Robust (linear) LQR controller obtained via Equation (7.6).
- **Robust MPC:** A robust model-predictive control algorithm [KBM96] based on state-dependent LMIs. (As the relevant LMIs are not always guaranteed to solve, our implementation temporarily reverts to the Robust LQR policy when that occurs.)
- **RARL:** The robust adversarial reinforcement learning algorithm [Pin+17], which trains an RL agent in the presence of an adversary. (We note that unlike the other robust methods considered here, this method is not *provably* robust.)

Table 7.1: Performance of various approaches, both robust (right) and non-robust (left). We report average quadratic loss over 50 episodes under the original dynamics (O) and under an adversarial disturbance (A). For the original dynamics (O), the best performance for both non-robust methods and robust methods is in bold (lower loss is better). Under the adversarial dynamics (A), we seek to observe whether or not methods remain stable; we use "*unstable*" to indicate cases where the relevant method becomes unstable (and † to denote any instabilities due to numerical, rather than theoretical, issues). Our robust methods (denoted by *) improve performance over Robust LQR and Robust MPC in the average case while remaining stable under adversarial dynamics, whereas the non-robust methods and RARL either go unstable or receive much larger losses.

Environment		**LQR**	**MBP**	**PPO**	**Robust LQR**	**Robust MPC**	**RARL**	**Robust MBP***	**Robust PPO***
Generic NLDI ($D = 0$)	O	373	**16**	21	253	253	**27**	69	33
	A	——— *unstable* ———			1009	873	*unstable*	1111	2321
Generic NLDI ($D \neq 0$)	O	278	**15**	82	199	199	147	**69**	80
	A	——— *unstable* ———			1900	1667	*unstable*	1855	1669
Cart-pole	O	36.3	**3.6**	7.2	10.2	10.2	**8.3**	9.7	8.4
	A	— *unstable* —		172.1	42.2	47.8	41.2	50.0	16.3
Quadrotor	O	5.4	**2.5**	7.7	13.8	13.8	12.2	11.0	**8.3**
	A	*unstable*	545.7	137.6	64.8	*unstable*†	63.1	25.7	26.5
Microgrid	O	4.59	**0.60**	0.61	0.73	0.73	0.67	**0.61**	**0.61**
	A	——— *unstable* ———			0.99	0.92	2.17	7.68	8.91

- **LQR:** A standard non-robust (linear) LQR controller.
- **MBP** and **PPO:** The non-robust neural network policy class $\hat{\pi}_\theta(x)$ optimized via a model-based planner and the PPO algorithm, respectively.

In order to evaluate performance, we *train* all methods on the dynamical settings described in Section 7.5.1, and *evaluate* them on two different variations of the dynamics:

- **Original dynamics:** The dynamical settings described above ("average case").
- **Adversarial dynamics:** Modified dynamics with an adversarial test-time disturbance $w(t)$ generated to maximize loss ("worst case"). We generate this disturbance separately for each method described above (see Appendix C.7 for more details).

Initialization states are randomly generated for all experiments. For the synthetic NLDI and microgrid settings, these are generated from a standard normal distribution. For both cart-pole and quadrotor, because our NLDI bounds model linearization error, we must generate initial points within a region where this linearization holds. In particular, the linearization bounds only hold for a specified L_∞ ball, B_{NLDI}, around the equilibrium. We use a simple heuristic to construct this ball and jointly find a smaller L_∞ ball, B_{init}, such that there exists a level set L of the Robust LQR Lyapunov function with $B_{\text{init}} \subseteq L \subseteq B_{\text{NLDI}}$ (details in Appendix C.8). Since Robust LQR (and by extension our methods) are guaranteed to decrease the relevant Lyapunov function, this guarantees that these methods will never leave B_{NLDI} when initialized starting from any point inside B_{init} – i.e., that our NLDI bounds will always hold throughout the trajectories produced by these methods.

7.5.3 Results

Table 7.1 shows the performance of the above methods. We report the integral of the quadratic loss over the prescribed time horizon on a test set of states, or indicate cases

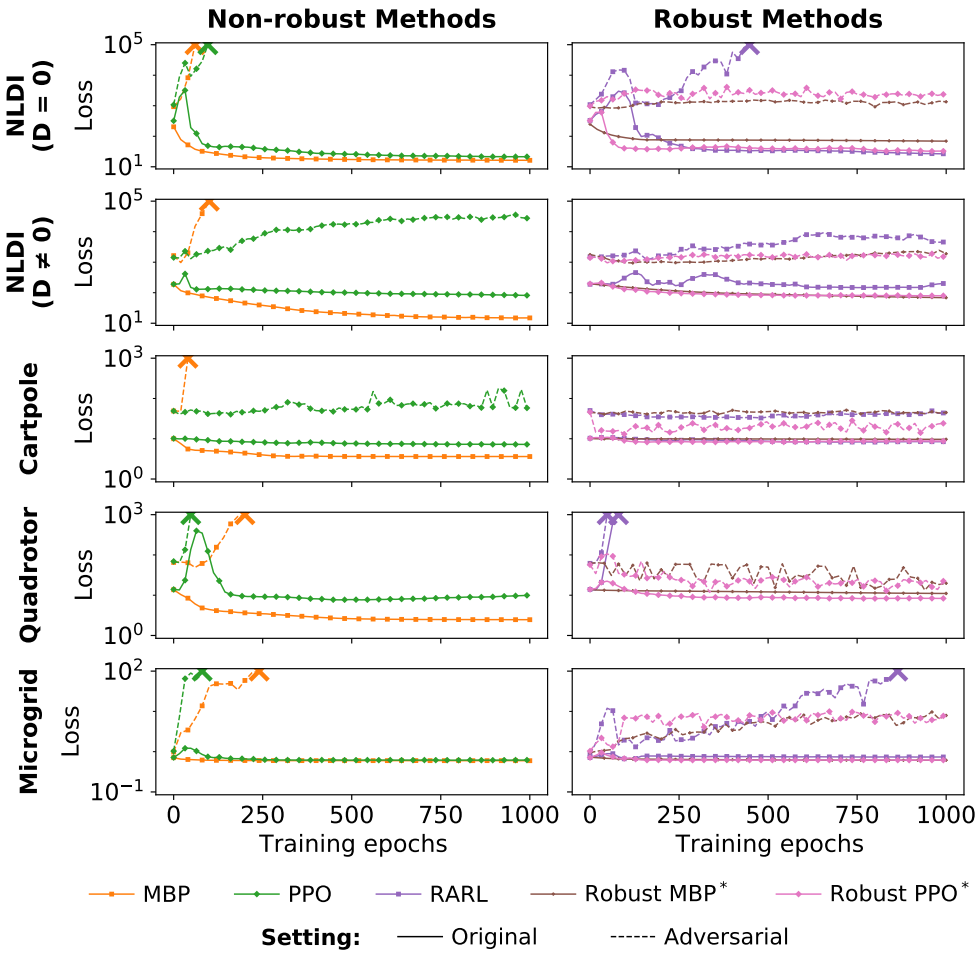


Figure 7.1: Test performance over training epochs for all learning methods employed in our experiments. For each training epoch (10 updates for the MBP model and 18 for PPO), we report average quadratic loss over 50 episodes, and use "X" to indicate cases where the relevant method became unstable. (Lower loss is better.) Our robust methods (denoted by *), unlike the non-robust methods and RARL, remain stable under adversarial dynamics throughout training.

where the relevant method became unstable (i.e., the loss became orders of magnitude larger than for other approaches). (Sample trajectories are also provided in Appendix C.8.)

These results illustrate the basic advantage of our approach. In particular, both our Robust MBP and Robust PPO methods show improved "average-case" performance over the other provably robust methods (namely, Robust LQR and Robust MPC). As expected, however, the non-robust LQR, MBP, and PPO methods often perform better within the original nominal dynamics, as they are optimizing for expected performance but do not need to consider robustness under worst-case scenarios. However, when we apply allowable adversarial perturbations (that still respect our disturbance bounds), the non-robust LQR, MBP, and PPO approaches diverge or perform very poorly. Similarly, the RARL agent performs well under the original dynamics, but diverges under adversarial perturbations in the generic NLDI settings. In contrast, both of our provably robust approaches (as well as Robust LQR) remain stable under even "worst-case" adversarial dynamics. (We note

that the baseline Robust MPC method goes unstable in one instance, though this is due to numerical instability issues, rather than issues with theoretical guarantees.)

Figure 7.1 additionally shows the performance of all neural network-based methods on the test set over training epochs. While the robust and non-robust MBP and PPO approaches both converge quickly to their final performance levels, both non-robust versions become unstable under the adversarial dynamics very early in the process. The RARL method also frequently destabilizes during training. Our Robust MBP and PPO policies, on the other hand, remain stable throughout the *entire* optimization process, i.e., do not destabilize during either training *or* testing. Overall, these results show that our method is able to learn policies that are more expressive than traditional robust methods, while *guaranteeing* these policies will be stable under the same conditions as Robust LQR.

While not depicted in the numerical results above, we note that the chief tradeoff, here, is computational cost. In particular, our approaches require computing a last-layer projection during every step of training and inference, which can be expensive to compute depending on the size and type of the projection. This is in contrast to both of the LQR-based methods we evaluate (which only require one optimization problem to be solved up front) as well as all RL baselines we evaluate (which do not require explicit optimization solves). (Robust MPC, on the other hand, is much more expensive to run, as it requires solving an ho at every time step.) Thus, reducing the computational cost of our approach – e.g., by speeding up the projection layer or using cheaper proxy projection procedures – is an important direction for future work.

7.6 Conclusion

In this chapter, we have presented a class of nonlinear control policies that combines the expressiveness of neural networks with the provable stability guarantees of traditional robust control. This policy class entails projecting the output of a neural network onto a set of stabilizing actions, parameterized via robustness specifications from the robust control literature, and can be optimized using a model-based planning algorithm if the dynamics are known or virtually any RL algorithm if the dynamics are unknown. We instantiate our general framework for dynamical systems characterized by several classes of linear differential inclusions that capture many common robust control settings. In particular, this entails deriving efficient, differentiable projections for each setting, via implicit differentiation techniques. We show over a variety of simulated domains that our method improves upon robust LQR techniques while, unlike non-robust LQR and neural network methods, remaining stable even under worst-case allowable perturbations of the underlying dynamics.

We believe that our approach highlights the possible connections between traditional control methods and (deep) RL methods. Specifically, by enforcing more structure in the classes of deep networks we consider, it is possible to produce networks that *provably* satisfy many of the constraints that have typically been thought of as outside the realm of RL. We hope that this work paves the way for future approaches that can combine more structured uncertainty or robustness guarantees with RL, in order to improve performance in settings traditionally dominated by classical robust control.

Chapter 8

Enforcing Policy Feasibility Constraints through Differentiable Projection for Energy Optimization

While reinforcement learning (RL) is gaining popularity in energy systems control, its real-world applications are limited due to the fact that the actions from learned policies may not satisfy functional requirements or be feasible for the underlying physical system. In this chapter, we propose PROjected Feasibility (PROF), a method to enforce convex operational constraints within neural policies. Specifically, using a similar general paradigm as in Chapter 7, we incorporate a differentiable projection layer within a neural network-based policy to enforce that all learned actions are feasible. We then update the policy end-to-end by propagating gradients through this differentiable projection layer, making the policy cognizant of the operational constraints. We demonstrate our method on two applications: energy-efficient building operation and inverter control. In the building operation setting, we show that PROF maintains thermal comfort requirements while improving energy efficiency by 4% over state-of-the-art methods. In the inverter control setting, PROF perfectly satisfies voltage constraints on the IEEE 37-bus feeder system, as it learns to curtail as little renewable energy as possible within its safety set.

The work in this chapter has previously been published in:.[1]

> Bingqing Chen*, Priya L. Donti*, Kyri Baker, J. Zico Kolter, and Mario Bergés. "Enforcing Policy Feasibility Constraints through Differentiable Projection for Energy Optimization." *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*. 2021, 199–210.

[1] Code for all experiments is available at: https://github.com/INFERLab/PROF.

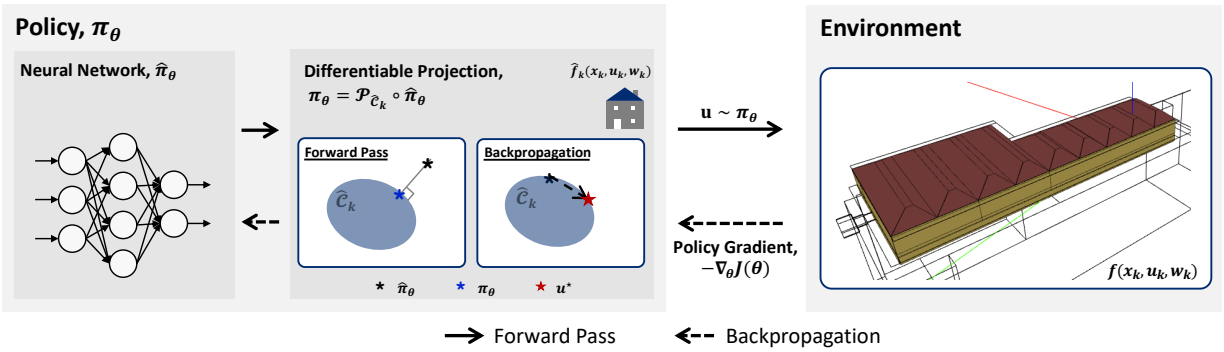


Figure 8.1: The PROF framework. Our policy consists of a neural network followed by a differentiable projection onto a convexified set of operational constraints, $\hat{\mathcal{C}}_k$ (which is constructed via an approximate model, $\hat{f}_k$, of the environment). The differentiable projection layer enforces the constraints in the forward pass, and induces policy gradients that make the neural network cognizant of the constraints in its learning.

8.1 Introduction

There has been increasing interest in using learning-based methods such as reinforcement learning (RL) for applications in energy systems control. However, a fundamental challenge with many of these methods is that they do not respect the physical constraints or functional requirements associated with the systems in which they operate. Therefore, there have been many calls for embedding safety guarantees into learning-based methods in the context of energy systems applications [ZZQ19; Gla19; Dob+20].

One common proposal to address this challenge is to provide machine learning methods with "soft penalties" to encourage them to learn feasible solutions. For instance, Zhang and Lam [ZL18] and Chen, Cai, and Bergés [CCB19] incentivize their RL-based building HVAC controller to satisfy thermal comfort constraints by adding a constraint violation penalty to the reward function. While such approaches often involve tuning some weight on the penalty term, recent work has proposed more theoretically-grounded approaches to choosing these weights; for instance, in the setting of approximating AC optimal power flow, Fioretto, Mak, and Van Hentenryck [FMVH20] and Chatzos, Fioretto, Mak, and Van Hentenryck [Cha+20] interpret the weight on their constraint violation penalty as a dual variable, and learn it via primal-dual updates. Gupta, Kekatos, and Jin [GKJ20] adopt a similar approach in an inverter control problem. However, a challenge with these types of "soft penalty" methods in general is that while they *incentivize* feasibility, they do not strictly *enforce* it, which is potentially untenable in safety-critical energy systems applications.

Given this limitation, a second class of approaches has aimed to strictly enforce operational constraints. For instance, in some cases, the outputs of a machine learning algorithm can be clipped post-hoc in order to make them feasible. However, a challenge is that such post-hoc corrections are not taken into account during the learning process, potentially negatively impacting overall performance. More recent approaches based in deep learning have therefore aimed to enforce simple classes of constraints in a way that *can* be taken into account during learning; for instance, Zamzam and Baker [ZB20] train a neural network to approximate AC optimal power flow (OPF), and enforce box constraints on certain

variables via sigmoid activations in the last layer of the neural network. In general, however, existing approaches have only been able to accommodate *simple* sets of constraints, prompting a need for methods that can incorporate broader classes of constraints.

In this chapter, drawing inspiration from Chapter 7, we propose a method to enforce general convex constraints into RL-based controllers in a way that can be taken into account during the learning process. In particular, we construct a neural network-based policy that culminates in a projection onto a set of constraints characterized by the underlying system. While the "true" constraints associated with the system may be somewhat complex, we observe that simple, approximate physical models are often available for many systems of interest, allowing us to specify convex approximations to the relevant constraints. The projections onto these (approximate) sets can thus be characterized as convex optimization problems, allowing us to leverage recent developments in differentiable convex optimization [AK17; Agr+19] to train our neural network and projection end-to-end using standard RL methods. The result is a powerful neural network-based policy that can flexibly optimize performance on the true underlying dynamics, while still satisfying the specified constraints.

We demonstrate our PROjected Feasibility approach, PROF, on two settings of interest. Specifically, we explore a building operation setting in which the goal is to reduce energy consumption during the heating season, while ensuring the satisfaction of thermal comfort constraints. We additionally explore an inverter control setting where the goal is to mitigate curtailment, while satisfying inverter operational constraints and nodal voltage bounds. In both settings, we find that our controller achieves good performance with respect to the control objective, while ensuring that relevant operational constraints are satisfied. To summarize, our key contributions are as follows:

- **A framework for incorporating convex constraints.** We propose a projection-based method to flexibly enforce convex constraints within neural network policies (as summarized in Figure 8.1). By examining the gradient fields of the differentiable projection layer, we recommend the incorporation of an auxiliary loss for more robust results. We also show in an ablation study (Section 8.5.3) that propagating gradients through the differentiable projection layer is indeed conducive to policy learning.
- **Demonstration on building control.** In the building control setting, we show that PROF further improves energy efficiency by 10% and 4%, respectively, compared to the best-performing RL agents in [ZL18] and [CCB19]. By using a locally-linear assumption to approximate the building thermodynamics and thereby formulating the constraints as a polytope [Zha+17; Che+20a], we largely maintain the temperature within the deadband, except when the control is saturated.
- **Demonstration on inverter control.** In the inverter control setting, PROF satisfies the voltage constraints 100% of the time over more than half a million time steps (1 week at one second per time step), with a randomly initialized neural network, compared to 22% over-voltage violations incurred by a Volt/Var control strategy. In terms of minimizing renewable generation curtailment, PROF performs as well as possible within its conservative safety set after learning safely for a day.

8.2 Related work

Our approach relies on recent developments in implicit neural network layers to enforce constraints via differentiable projections; relevant background and related work is provided in Section 2.2.3. Our work is also thematically related to the area of safe reinforcement learning; relevant background and related work is provided in Section 7.2. At the intersection of these topics, several prior works employ some form of differentiable projection within the loop of deep RL to enforce some notion of "safety" (though the particular notion of safety considered varies considerably between settings). For instance, within the context of constrained Markov decision processes (C-MDPs), Yang, Rosca, Narasimhan, and Ramadge [Yan+20] project neural network-based policies onto a linearly-constrained set of policies with bounded cumulative discounted cost. In the setting of robotic motion planning, Pham, De Magistris, and Tachibana [PDMT18] project actions onto a linear set of robotic operational constraints, and apply separate updates to the neural network based on both pre-projection and post-projection actions. In our own prior work (discussed in Chapter 7), we enforce asymptotic stability guarantees by projecting the actions output by our controller onto a convex set of actions satisfying stability specifications obtained via robust control. Similarly to these prior works, our approach employs differentiable projections within a neural network policy to enforce operational constraints over some planning horizon.

We employ our approach in the context of energy systems optimization, namely for the efficient operation of building heating and cooling systems and for grid inverter control. We refer interested readers to [ZZQ19; Gla19; Dob+20; Drg+20; Rol+22] for comprehensive reviews of relevant work in power and energy systems application domains.

8.3 Preliminaries: Reinforcement learning

The goal of RL is to learn an optimal control policy through direct interaction with the environment. The problem is usually formulated as a Markov decision process (MDP). At each time step k, the agent selects an action u_k given the current state x_k, using its policy π_θ (Equation (8.1)). In many modern RL techniques, the policy is commonly represented by a neural network parameterized by θ. When the agent takes the action u, the state transitions to x' based on the system dynamics f (Equation (8.2)), and the agent receives a reward r_k (or equivalently, incurs a cost $c_k = -r_k$).

$$u \sim \pi_\theta(u_k|x_k), \tag{8.1}$$

$$x' \sim f(x_k, u_k). \tag{8.2}$$

RL algorithms optimize for a policy that maximizes the expected cumulative reward, or equivalently, minimizes the expected cumulative cost, where γ is a temporal discount factor:

$$\theta^\star = \arg\max_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{l=0}^{\infty} \gamma^l r_{k+l}\right] = \arg\min_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{l=0}^{\infty} \gamma^l c_{k+l}\right]. \tag{8.3}$$

To simplify notation, we will denote the expected cumulative cost as $J(\theta)$, i.e.,

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{l=0}^{\infty} \gamma^l c_{k+l}\right]. \tag{8.4}$$

There are three general approaches to RL, namely value-based methods, policy gradient methods, and actor-critic methods. Value-based methods, e.g., Q-learning and its variants, update the value function of state-action pairs using the Bellman equation and take the action that maximizes the value of an action selection policy (the Q function) through exploration. Policy gradient methods, e.g., Proximal Policy Optimization (PPO) [Sch+17b], directly search for an optimal policy $\pi_\theta^\star$ using estimates of policy gradients. Denoting the policy gradient as $g := \nabla_\theta J(\theta)$, the core idea of policy gradient algorithms is that they update θ based on an estimate, $\hat{g}$, of the gradient, i.e.,

$$\theta \leftarrow \theta - \alpha \hat{g} \tag{8.5}$$

for some learning rate α. Different algorithms vary in how they obtain $\hat{g}$. For instance, the learning objective for PPO, which we use in our building control experiment (Section 8.5), is given by the following equation, where $\hat{A}_t$ is the generalized advantage estimate that can be estimated via any of the estimators in [Sch+15]:

$$\begin{aligned} J_{\text{PPO}}(\theta) &= \hat{\mathbb{E}}_k\left[\min(w_k(\theta)\hat{A}_k, \text{clip}(w_k(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_k)\right], \\ w_k(\theta) &= \frac{\pi_\theta(u_k|x_k)}{\pi_{\theta_{old}}(u_k|x_k)}, \end{aligned} \tag{8.6}$$

and the estimate $\hat{g}$ is constructed based on this learning objective. Actor-critic methods, e.g., Advantage Actor-Critic (A2C), are hybrids of the value-based and policy gradient approaches, using a policy network to select actions (the actor) and a value network to evaluate the action (the critic).

8.4 Enforcing feasibility via differentiable projection

We now describe PROF, which incorporates differentiable projections onto convex(ified) sets of operational constraints within a neural policy.

8.4.1 Problem formulation

Consider a discrete-time dynamical system

$$x_{k+1} = f(x_k, u_k, w_k), \tag{8.7}$$

where $x_k \in \mathbb{R}^s$ is the state at time k, $u_k \in \mathbb{R}^a$ is the control input, $w_k \in \mathbb{R}^d$ is an uncontrollable disturbance (which we assume to be observable), and $f : \mathbb{R}^s \times \mathbb{R}^a \times \mathbb{R}^d \to \mathbb{R}^s$ denotes the system dynamics. Letting $\mathcal{X}_k$ and $\mathcal{U}_k$ denote the allowable state and action

space, respectively, we can define the set of all feasible actions over the planning horizon T as $\mathcal{C}_k$, where

$$\mathcal{C}_k = \left\{ u_{k:k+T-1} \;\middle|\; \begin{array}{l} x_{i+1} = f(x_i, u_i, w_i), \\ x_i \in \mathcal{X}_i,\ u_i \in \mathcal{U}_i \end{array} \quad \forall i \in \{k, ..., k+T-1\} \right\}. \tag{8.8}$$

Our goal is then to learn a policy that optimizes the control objective, J, while enforcing the operational constraints. To simplify notation, we denote $\mathbf{u} = u_{k:k+T-1}$. In the case of a deterministic policy, i.e., $\mathbf{u} = \pi_\theta$, the learning problem is simply

$$\min_\theta \quad J(\theta) \quad \text{s.t.} \quad \pi_\theta \in \mathcal{C}_k. \tag{8.9}$$

In the case of a stochastic policy, e.g. $\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, $[\boldsymbol{\mu}, \boldsymbol{\sigma}] = \pi_\theta(x_k)$, we can write the problem as

$$\min_\theta \quad J(\theta) \quad \text{s.t.} \quad \mathbf{u}, \boldsymbol{\mu} \in \mathcal{C}_k. \tag{8.10}$$

In this case, it is necessary to sample actions around $\boldsymbol{\mu}$ in order to estimate policy gradients. At the same time the actions sampled from π_θ might fall outside of $\mathcal{C}_k$. Thus, we enforce that both $\boldsymbol{\mu}$ and the sample action $\mathbf{u}$ satisfy the constraints.

8.4.2 Approximate convex constraints

In practice, there are two key challenges inherent in solving Equations 8.9–8.10 as written. The first is that the disturbances w_i are not known ahead of time, meaning that the optimization problem must be solved under uncertainty. One approach to addressing this, from the field of robust control [ZD98], involves constructing an uncertainty set over the disturbance, and then optimizing for worst-case or expected cost under this uncertainty set. Here, we simply assume a predictive model of the disturbances is available. (By re-planning frequently, we observe that the prediction errors have limited empirical impact on performance in the two applications we study.) We will use the notation $\hat{w}_k$ to denote our forecast of the disturbance if k is a future time step, and the true value of the disturbance if k is the present or a prior time step.

The second challenge pertains to the form of the set $\mathcal{C}_k$, which may be poorly structured or otherwise difficult to optimize over. In particular, our framework relies on obtaining convex approximations to the constraints in order to enable differentiable projections (see Section 2.2.3.1). Fortunately, for many energy systems applications, some approximate model $\hat{f}_k$ is often available based on domain knowledge that allows $\mathcal{C}_k$ to be approximated as a convex set, despite the complex nature of the true dynamical system.

Thus, letting $\hat{f}_i$ denote our approximations of the dynamics and $\hat{w}_i$ denote the (forecast or known) disturbance at each $i = k, \ldots, k+T-1$, we define our approximate convex constraint set as

$$\hat{\mathcal{C}}_k = \left\{ u_{k:k+T-1} \;\middle|\; \begin{array}{l} x_{i+1} = \hat{f}_i(x_i, u_i, \hat{w}_i), \\ x_i \in \mathcal{X}_i,\ u_i \in \mathcal{U}_i \end{array} \quad \forall i \in \{k, ..., k+T-1\} \right\}. \tag{8.11}$$

We note that f and w are approximated *solely* for the purposes of constructing approximate constraint sets, and are not used otherwise during training and inference (i.e., our neural policy interacts with the *true* dynamics and disturbances during training and inference).

8.4.3 Policy optimization

Let $\hat{\pi}_\theta$ be any (e.g., fully-connected or recurrent) neural network parameterized by θ. Our policy entails passing the output from the neural network to the differentiable projection layer $\mathcal{P}_{\hat{\mathcal{C}}_k}$ characterized by the approximate constraints, which enforces that the resultant action is feasible with respect to these constraints. The overall (differentiable) neural policy is then given by

$$\pi_\theta(x_k) = \mathcal{P}_{\hat{\mathcal{C}}_k} \circ \hat{\pi}_\theta(x_k).^{2} \tag{8.12}$$

The key benefit of embedding a differentiable projection into our policy is that it enforces constraints in a way that is visible to the neural network during learning. In this work, we implement the differentiable projection using the `cvxpylayers` library [Agr+19].

We construct the following loss function, which is a weighted sum of the control objective J and an auxiliary loss term to be explained shortly in this section. $\lambda > 0$ is a hyperparameter.

$$\mathcal{L}(\theta, x_k) = J(\theta) + \lambda \|\pi_\theta(x_k) - \hat{\pi}_\theta(x_k)\|_2^2. \tag{8.13}$$

We then train our policy (Equation (8.12)) to minimize this cost using standard approaches in deep reinforcement learning. The full algorithm is presented in Algorithm 4.

8.4.3.1 Visualization of gradient fields.

To provide more intuition on the differentiable projection layer and our cost function, we visualize the gradient fields in a hypothetical example with a deterministic policy and a planning horizon of $T = 1$. Specifically, for the purposes of illustration, let $u^\bullet$ and $u^\star$ denote unique optimal actions minimizing some convex control cost J in the unconstrained and constrained settings, respectively:

$$\begin{aligned} u^\bullet \sim \pi_{\theta^\bullet}; \quad \theta^\bullet &= \arg\min_\theta \quad J(\theta) \\ u^\star \sim \pi_{\theta^\star}; \quad \theta^\star &= \arg\min_\theta \quad J(\theta) \quad \text{s.t.} \quad u \in \mathcal{C}_k. \end{aligned}$$

In Figure 8.2, we then plot the gradient fields in two cases: (a) $u^\bullet \notin \mathcal{C}_k$, and (b) $u^\bullet \in \mathcal{C}_k$. Note that $u^\bullet$ and $u^\star$ are assumed to be known here for illustrative purposes only, and are not known during training.

In particular, we plot the gradients (black arrows) of $\|u^\bullet - \mathcal{P}_{\mathcal{C}_k} \circ \hat{\pi}\|_2^2$ with respect to the output of the neural network $\hat{\pi}$. These indicate the direction in which the neural network would be incentivized to update in order to minimize the system cost. If no differentiable projection were embedded within the policy, all the gradients would point towards $u^\bullet$ without regard for the constraints. Instead, in the case of $u^\bullet \notin \mathcal{C}_k$ (Figure 8.2a), the gradients

[2] We use the notation $f \circ g(x) \coloneqq f(g(x))$ to denote function composition.

Algorithm 4 PROF

procedure MAIN(env, J) *// input: environment, control objective*
  init neural network $\hat{\pi}_\theta$, replay memory $\mathcal{M}$
  specify RL algorithm $\mathcal{A}$, batch size M, update interval K
  specify planning horizon T
  // online execution
  for $k = 1, \ldots$ **do**
    observe state x_k
    predict future disturbances $\hat{w}_{k:k+T-1}$
    construct constraint set $\hat{\mathcal{C}}_k$, policy $\pi_\theta = \mathcal{P}_{\hat{\mathcal{C}}_k} \circ \hat{\pi}_\theta$
    compute $u_k =$ INFERENCE(π_θ, x_k, T)
    execute action ENV.STEP(u_k)
    save MEMORY.APPEND(x_k, u_k, $\hat{w}_{k:k+T-1}$)
    // update policy every K time steps
    if $\mathrm{mod}(k, K) = 0$ **then**
      $\hat{\pi}_\theta =$ TRAIN($\hat{\pi}_\theta$, J, $\mathcal{M}$, $\mathcal{A}$)
    end if
  end for
end procedure

procedure INFERENCE(π_θ, x_k, T)
  // input: neural policy, current state, planning horizon
  select action $u_{k:k+T-1} \sim \pi_\theta$
   // only return the current action; replan at each time step
  return u_k
end procedure

procedure TRAIN($\hat{\pi}_\theta$, J, $\mathcal{M}$, $\mathcal{A}$)
  // input: neural policy, objective, replay memory, RL algorithm
  init $\mathcal{L}(\theta) = 0$
  for $i = 1, \ldots, M$ **do**
    sample $x, u, w \sim \mathcal{M}$
    construct constraint set $\hat{\mathcal{C}}_k$, policy $\pi_\theta = \mathcal{P}_{\hat{\mathcal{C}}_k} \circ \hat{\pi}_\theta$
    compute training loss

$$\mathcal{L}(\theta) \mathrel{+}= J(\theta) + \lambda \|\pi_\theta(x) - \hat{\pi}_\theta(x)\|_2^2$$

  end for
  train $\hat{\pi}_\theta$ via $\mathcal{A}$ to minimize $\mathcal{L}$
  return $\hat{\pi}_\theta$
end procedure

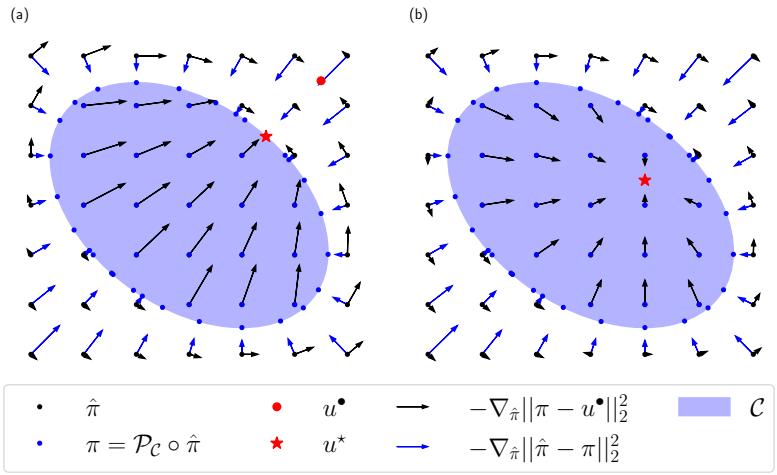


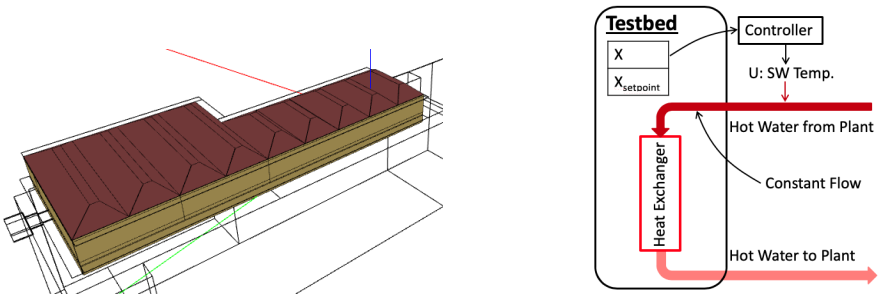
Figure 8.2: Illustrative example of gradients from the differentiable projection layer. $u^\bullet$ and $u^\star$ denote unique optimal actions minimizing some convex control objective J in the unconstrained and constrained settings, respectively; $\nabla_{\hat{\pi}} \|\pi - u^\bullet\|_2^2$ is thus a proxy for $\nabla_{\hat{\pi}} J$. (a) $u^\bullet \notin \mathcal{C}$. The gradients $\nabla_{\hat{\pi}} J$ point towards $u^\star$ as desired, such that $\pi = \mathcal{P}_{\hat{\mathcal{C}}} \circ \hat{\pi}$ will reach this optimal point. (b) $u^\bullet = u^\star$ on the interior of $\mathcal{C}$. The gradients $\nabla_{\hat{\pi}} J$ do not cause $\hat{\pi}$ (or its projection) to update towards the interior. Adding a weighted auxiliary loss term, e.g., $\|\pi - \hat{\pi}\|$, can help direct updates towards the interior.

through the differentiable projection layer point towards $u^\star$ instead of $u^\bullet$. More specifically, if $\hat{\pi}_\theta(x_k) \in \mathcal{C}_k$, then the projection layer is simply the identity, and the gradients point directly towards $u^\star$; otherwise, the gradients point along the boundary of $\mathcal{C}_k$ in the direction of $u^\star$.

This case is of particular interest, as in many practical applications some operational constraint will be binding. As a concrete example, the ultimate energy-saving strategy for building operations is to keep all mechanical systems off (i.e., $u^\bullet = 0$), which obviously violates occupants' comfort requirements and is outside the set of allowable actions (i.e., $u^\bullet \notin \mathcal{C}_k$). Thus, the problem is to find a policy that uses the mechanical system as little as possible without violating comfort requirements. Given the common case where the control objective is convex, this then lies on the boundary of the constraint set (i.e., $u^\star = \mathcal{P}_{\mathcal{C}_k} \circ u^\bullet$).

We also depict the case where the solution of the unconstrained problem already satisfies the constraints, i.e., $u^\bullet = u^\star \in C_k$ (Figure 8.2b). If this is generally the case for a particular application, we note that a constraint enforcement approach (ours or otherwise) is likely not needed, and indeed utilizing gradients through the projection layer may actually degrade performance. Specifically, if $\hat{\pi}_\theta(x_k) \notin \mathcal{C}_k$, the gradients do not point towards the interior of the constraint set, meaning that $\pi_\theta(x_k) = \mathcal{P}_{\mathcal{C}_k} \circ \hat{\pi}_\theta(x_k)$ will lie on the boundary of the constraints despite the optimal solution being in the interior. This can be amended by augmenting the loss function with a (weighted) auxiliary term such as $\|\pi_\theta(x_k) - \hat{\pi}_\theta(x_k)\|_2^2$ whose gradients (blue arrows) point towards the interior.

It may not be known a priori whether or not $u^\bullet$ is in the constraint set in general or at any given time, except when domain experts are fully clear on the structure of the solutions for specific applications. In particular, $\mathcal{C}_k$ is time-varying, making it difficult to know for sure whether or not the constraints will indeed be binding at any given time. For robustness, we therefore recommend incorporating the auxiliary loss $\|\pi_\theta(x_k) - \hat{\pi}_\theta(x_k)\|_2^2$



(a) Geometric view (b) System schematic

Figure 8.3: Building simulation testbed (reproduced from [CCB19]).

within the RL training cost, unless it is known from domain knowledge that the constraints will certainly be active. As such, we formulate the training cost function as previously given in Equation (8.13).

8.5 Experiment 1: Energy-efficient building operation

There is significant potential to save energy through more efficient building operation. Buildings account for about 40% of the total energy consumption in the United States, and it is estimated that up to 30% of that energy usage may be reduced through advanced sensing and control strategies [Fer+17]. However, this potential is largely untapped, as the heterogeneous nature of building environments limits the ability of control strategies developed for one building to scale to others [CCB19]. RL can address this challenge by adapting to individual buildings by directly interacting with the environment.

The most important constraint in building operation is to maintain a satisfactory level of comfort for occupants, while minimizing energy consumption. It is common in the RL-based building control literature to penalize thermal comfort violations [ZL18; CCB19], which incentivizes but does not guarantee the satisfaction of these comfort requirements. In comparison, our proposed neural policy can largely maintain temperature within the specified comfortable range, except when the control is saturated.

We evaluate our policy in the same simulation testbed as [ZL18; CCB19], following the same experimental setup as [CCB19]. Specifically, we first pre-train the neural policy by imitating a proportional-controller (P-controller). We then evaluate and further train our agent in the simulation environment, using a different sequence of weather data.

8.5.1 Problem description

Simulation testbed. We utilize an EnergyPlus (E+) model of a 600m^2 multi-functional space (Figure 8.3a), based on the Intelligent Workplace (IW) on Carnegie Mellon University (CMU) campus, located in Pittsburgh, PA, USA. The system of interest is the water-based radiant heating system, of which a schematic is provided in Figure 8.3b. In this experiment,

we control the *supply water temperature* so as to maintain the state variable, i.e., the *zone temperature*, within a comfortable range during the heating season. In the existing control, the supply water (SW) is maintained at a constant flow rate, and its temperature is managed by a P-controller. For more information on the simulation testbed, refer to [ZL18].

Approximate system model. We approximate the environment as a linear system:

$$x_{k+1} \approx \hat{f}(x_k, u_k, w_k) = Ax_k + B_u u_k + B_d w_k, \tag{8.14}$$

where x_k represents the *zone temperature* and u_k represents the *supply water temperature*. w_k includes distributions from weather and occupancy. While building thermodynamics are fundamentally nonlinear, the locally-linear assumption works well for many control inputs [Pri+13]. We identify the approximate model parameters A, B_u, and B_d with prediction error minimization [Pri+13] on the same data used to pre-train the RL agent (see Section 8.5.2). The root mean squared error (RMSE) of this model on a unseen test set is 0.14°C.

Objective. Since our goal is to minimize energy consumption, we define the control cost at each time step as the agent's control action, i.e. *supply water temperature*, which is linearly proportional to the heating demand, i.e., $c_k = u_k$.

In contrast to the objectives in [ZL18; CCB19], which are defined as weighted sum of energy cost and some penalty on thermal comfort violations, we consider the thermal comfort requirement as hard constraints, in the form of Equation (8.10).

Constraints. To maintain a satisfactory comfort level, we require the zone temperature to be within a deadband $\mathcal{X} = \{x \mid 21.9°C \leq x \leq 25.5°C\}$ when the building is occupied, based on the building code requirement of 10% Predicted Percentage of Dissatisfied (PPD) [Fan86]. We allow for a wider temperature range during unoccupied hours. For the action, the allowable range of supply water temperature is $\mathcal{U} = \{u \mid 20°C \leq u \leq 65°C\}$.

While it may appear from this description that we have only simple box constraints on both the state and action, we highlight the fact that actions are coupled over time through the building thermodynamics [Zha+17]. More concretely, a future state depends on all past actions. Thus, a box constraint on x_{k+l+1} is in fact a constraint on $u_{k:k+l}$. In this case, assuming $\hat{f}$ to be a linear system, $\hat{\mathcal{C}}_k$ is then a set of linear inequalities, which can be geometrically interpreted as a polytope.[3] We refer interested readers to Chen, Francis, Pritoni, Kar, and Berg'es [Che+20a] and Zhao, Zhang, Hao, and Kalsi [Zha+17] for more details on this formulation. In fact, it was experimentally demonstrated in Chen, Francis, Pritoni, Kar, and Berg'es [Che+20a] that projecting actions onto the polytope constructed with an approximate linear model was sufficient to maintain temperature within the deadband in a real-world residential household (though Chen, Francis, Pritoni, Kar, and Berg'es [Che+20a] did not then differentiate through this projection).

Control time step. The EnergyPlus model has a 5-minute simulation time step. Following

[3] A polytope can be characterized as a set $\mathcal{S} = \{x \in \mathbb{R}^n | Ax \leq b\}$.

Zhang and Lam [ZL18] and Chen, Cai, and Bergés [CCB19], we use a 15-min control time step (i.e., each action is repeated 3 times) and a planning horizon of $T = 12$ (i.e., 3 hours).

8.5.2 Implementation details

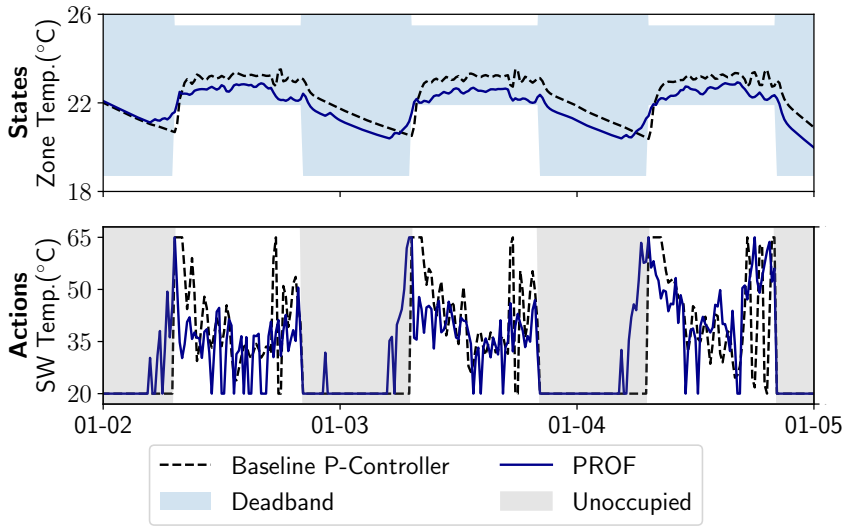
Offline pre-training. We pre-train a long short-term memory (LSTM) recurrent policy (without a subsequent projection) by imitating a P-controller operating under the Typical Meteorological Year 3 (TMY3) [WM08] weather sequence, from Jan. 1 to Mar. 31. We min-max normalize all of the state, action, and disturbance, and use a learning rate of 10^{-3}. Specifically, we use the pre-trained weights after training on the expert demonstrations for 20 epochs following the same procedures as Chen, Jin, Wang, Hong, and Bergés [Che+20b]. We refer readers to Chen, Jin, Wang, Hong, and Bergés [Che+20b] for more details on the neural network architecture, training procedures, loss, and performance evaluation.

Online policy learning. We optimize the policy with PPO [Sch+17b] over the weather sequence in 2017 from Jan. 1 to Mar. 31. We use $\lambda = 10$ (see Equation (8.13)), a learning rate of 5×10^{-4}, and RMSprop [TH12] as the optimizer. We update the policy every four days, by iterating over those samples for 8 epochs with a batch size of 32. For hyperparameters, we use a temporal discount rate of $\gamma = 0.9$, $\epsilon = 0.2$ (see Equation (8.6)), and a Gaussian policy (see Equation (8.10)) with σ linearly decreased from 0.1 to 0.01.

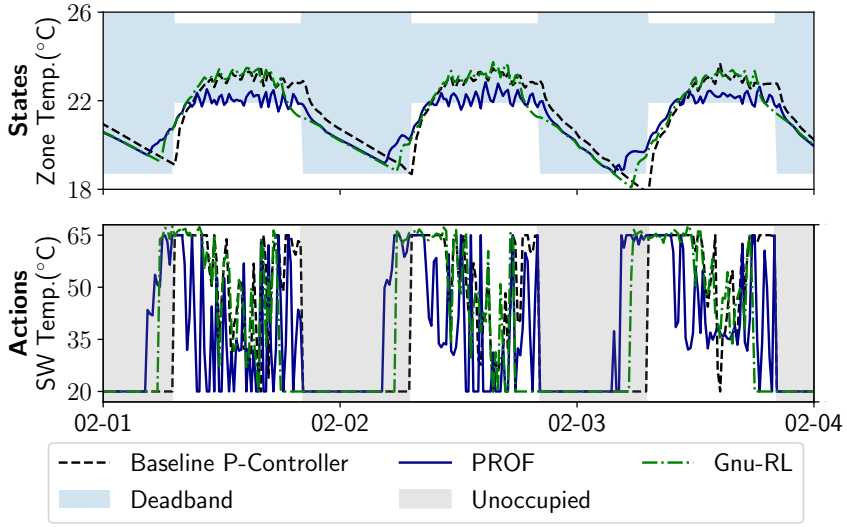
8.5.3 Results

After pre-training on expert demonstrations from the baseline P-controller, our agent directly operated the simulation testbed based on actual weather sequences in Pittsburgh from Jan. 1 to Mar. 31 in 2017. Figure 8.4a shows the behavior of our agent at the onset of deployment over a 3-day period. The baseline P-controller reactively turns on heating when the environment switches from unoccupied to occupied, which results in thermal comfort violations in the mornings. In comparison, PROF preheats the environment such that the environment is already at a comfortable temperature when occupants arrive in the morning. Notably, the differentiable projection layer manages to enforce this preheating behavior despite this behavior not being present in the expert demonstrations.

Figure 8.4b shows the behavior of our agent in comparison with Gnu-RL [CCB19], having interacted with and trained on the environment for a month. Gnu-RL is updated via PPO, similarly to the current work, and incorporates domain knowledge on system dynamics. In comparison to Gnu-RL [CCB19], which ends up trying to maintain temperature at the setpoint, PROF learns an energy-saving behavior by maintaining the temperature at the lower end of the deadband. This explains the further energy savings compared with Gnu-RL [CCB19]. However, we also notice that the temperature requirement may be violated on cold mornings. This happens when the control action is saturated, i.e., full heating over the 3-hour planning horizon is not sufficient to bring temperature back to the comfortable range. (In principle, even these constraint violations could be mitigated by increasing the length of the planning horizon.)



(a) The differentiable projection layer enforces preheating behavior to ensure deadband constraints are never violated, even though this behavior is not present in the expert demonstrations.



(b) The agent has found a more energy-efficient control strategy by maintaining temperature at the lower end of the deadband.

Figure 8.4: Behavior of our proposed agent (a) at the onset of deployment, with pre-trained weights based on expert demonstrations and (b) after a month of interacting with and training on the environment.

Table 8.1: Performance comparison. Our method saves energy while incurring minimal comfort violations.

	Heating Demand (kW)	**PPD**	
		Mean (%)	**SD (%)**
Existing P-Controller [ZL18]	43709	9.45	5.59
Agent #6 [ZL18]	37131	11.71	3.76
Baseline P-Controller [CCB19]	35792	9.71	6.87
Gnu-RL [CCB19]	34687	9.56	6.39
LSTM & Clip + No Update	37938	**8.55**	3.39
LSTM & Clip	36068 ± 2187	9.18 ± 0.67	3.49
PROF (ours)	**33271** ± 1862	9.68 ± 0.48	3.66

Table 8.1 summarizes the performance of our agent with comparison to the RL agents in [ZL18; CCB19]. Our proposed agent (averaged over 5 random seeds) saves 10% and 4% energy compared to the best-performing agents in [ZL18] and [CCB19], respectively.

We also compare our method to two ablations: (1) `LSTM & Clip + No Update`, which uses the same pre-trained weights and the projection layer to enforce feasible actions, but does not update the policy, and (2) `LSTM & Clip`, which uses the same pre-trained weights and the projection layer to enforce feasible actions during inference, but does not propagate gradients through the differentiable projection layer in the policy updates. We find that `LSTM & Clip` slightly improves upon `LSTM & Clip + No Update`, but is less performant compared to PROF. This affirms our hypothesis that the gradients through the differentiable projection layer are cognizant of the constraints and are thus conducive to policy learning.

8.6 Experiment 2: Inverter control

Distributed energy resources (DERs), e.g., solar photovoltaic (PV) panels and energy storage, are becoming increasingly prevalent in an effort to curb carbon dioxide emissions and combat climate change. However, DERs interfacing with the power grid via power electronics, such as inverters, also introduce unintended challenges for grid operators. For instance, over-voltages have become a common occurrence in areas with high renewable penetration [Str+20], and power electronics-interfaced generation has low-inertia and requires active control at much faster timescales compared to traditional synchronous machines [Mil+18].

To alleviate these issues, IEEE standard 1547.8-2018 [Bas14] recommends a Volt/Var control strategy in which the reactive power contribution of an inverter is based on local voltage measurements. As will be clear in our empirical evaluation, this network-agnostic heuristic based on local information alleviates, but does not avoid, over-voltage issues. Given that the optimal solution needs to be obtained at the system-level and that the problem needs to be solved at very short timescales, a common paradigm is to address the problem in a quasi-static fashion (e.g., as in [Bak+17; Jal+19; GKJ20]), where one chooses a policy over the next time period, e.g., 15 minutes-1 hour, and uses the policy without update for fast inference. In this work, we adopt the same paradigm and consider real-time control on

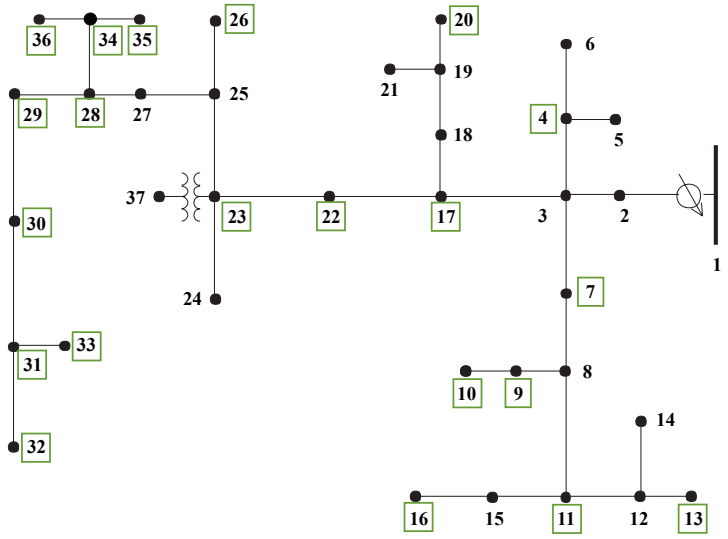


Figure 8.5: IEEE 37-bus feeder system, where the solar PV systems are indicated by green rectangles.

a 1-second timescale of both active (P) and reactive (Q) power setpoints at each inverter.

We envision that a neural policy can learn from its prior experiences, in contrast to the traditional *fit-and-forget* approach [Dob+20], and is capable of making decisions faster compared to solving optimization problems. Our primary contribution compared to existing work is the ability to enforce physical constraints within the neural network. In fact, we successfully enforce voltage constraints 100% of the time with a randomly initialized neural network, over more than half a million time-steps (i.e., 1 week with a one-second time step). The assumed control and communication scheme is consistent with the new definitions for smart inverter capabilities under IEEE standard 1547.1-2020 [IEE20].

8.6.1 Problem description

The problem we are considering here is to control active and reactive power setpoints at each inverter in order to maximize utilization (i.e., minimize curtailment) of renewable generation, while satisfying the maximum and minimum grid voltage requirements. Here, we first define the considered test case and input data, and describe the model of the network. We refer readers to Baker, Bernstein, Dall'Anese, and Zhao [Bak+17] for more details on the problem set-up.

IEEE 37-bus test case. We evaluate our method on the IEEE 37-bus distribution test system [IEE10a], with 21 solar PV systems indicated by green rectangles in Figure 8.5. We utilize a balanced, single-phase equivalent of the system, and simulate the nonlinear AC power flows using PYPOWER [ZMSG97]. For the simulation, the solar generation and loads are based on 1-second solar irradiance and load data collected from a neighborhood in Rancho Cordova, CA [BH13] over a period of one week (604800 samples).

Approximate system model. Denote the number of buses, excluding the slack bus (e.g.,

the distribution substation), as N; the net active and the reactive power as $\mathbf{p} \in \mathbb{R}^N$ and $\mathbf{q} \in \mathbb{R}^N$; and the voltage at all buses as $\mathbf{v} \in \mathbb{R}^N$. We linearize the AC power flow equations around the flat voltage solution, i.e. $\bar{\mathbf{v}} = \mathbb{1}$, using the method in [BD15]. The reference active and reactive power corresponding to $\bar{\mathbf{v}} = \mathbb{1}$ is denoted as $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$. The linearized grid model, $\hat{f}$, is given by Equation (8.15), where $\mathbf{R}$, $\mathbf{B} \in \mathbb{R}^{N \times N}$ represent system-dependent network parameters that can be either estimated from linearization (e.g., [BD15]) or data-driven methods:

$$\begin{aligned} \mathbf{v} \approx \hat{f}(\mathbf{p}, \mathbf{q}) &= \bar{\mathbf{v}} + \mathbf{R}(\mathbf{p} - \bar{\mathbf{p}}) + \mathbf{B}(\mathbf{q} - \bar{\mathbf{q}}) \\ &= \bar{\mathbf{v}} + \underbrace{[\mathbf{R}, \mathbf{B}]}_{\mathbf{H}} \underbrace{\begin{bmatrix} \mathbf{p} - \bar{\mathbf{p}}, \\ \mathbf{q} - \bar{\mathbf{q}} \end{bmatrix}}_{\mathbf{u}}. \end{aligned} \tag{8.15}$$

A notable advantage of the method in [BD15] is that the resulting model has bounded error with respect to the true dynamics. By incorporating the error bound when constructing the safety set, the safety set is guaranteed to be a conservative under-approximation of the true safety set, and thus allow us to satisfy voltage constraints 100% of the time.

Policy. Our policy takes as input the voltage from the previous time-step, load, and generation at all the buses, and outputs active and reactive power setpoints at each inverter. (This is a deterministic policy; see Equation (8.9).) Note that while the grid model (Equation (8.15)) contains all N buses, only those with inverters are controllable.

Our neural architecture is similar to the one used in [GKJ20], which consists of a utility-level network, and inverter-level networks for individual inverters. The utility-level network collects information from all nodes, and broadcasts an intermediate representation to all inverter-level networks. Using this information alongside its local observations, each inverter makes its local control decisions, which are then projected onto the constraints.

Objective. The objective is to minimize the curtailment of solar generation, or equivalently to maximize the utilization of the available solar power, p_{av}. Specifically, letting $\mathcal{I}$ denote the set of buses with inverters, the objective is

$$J(\theta) = \min_{\mathbf{p}_\mathcal{I}, \mathbf{q}_\mathcal{I}} \sum_{i \in \mathcal{I}} [p_{av,i} - p_i]_+, \quad \text{where } \begin{bmatrix} \mathbf{p}_\mathcal{I} & \mathbf{q}_\mathcal{I} \end{bmatrix} = \pi_\theta \tag{8.16}$$

Constraints. For an individual inverter, i, with rated power s_i and an available power (from available solar generation) $p_{av,i}$, the feasible action space is

$$\mathcal{U}_i(k) = \{(p_i, q_i) : 0 \le p_i \le p_{av,i}(k), p_i^2 + q_i^2 \le s_i^2\}$$

$$\mathcal{U}(k) := \mathcal{U}_1(k) \times \cdots \times \mathcal{U}_{|\mathcal{I}|}(k).$$

At the same time, the voltage at each bus should remain between 0.95-1.05 $p.u.$ The primary challenge of satisfying voltage constraints is that the voltage at each bus depends on actions of neighboring nodes, i.e.

$$\mathcal{X} = \{v \mid 0.95 \times \mathbb{1} \le \mathbf{v} \approx \bar{\mathbf{v}} + \mathbf{H}\mathbf{u} \le 1.05 \times \mathbb{1}\},$$

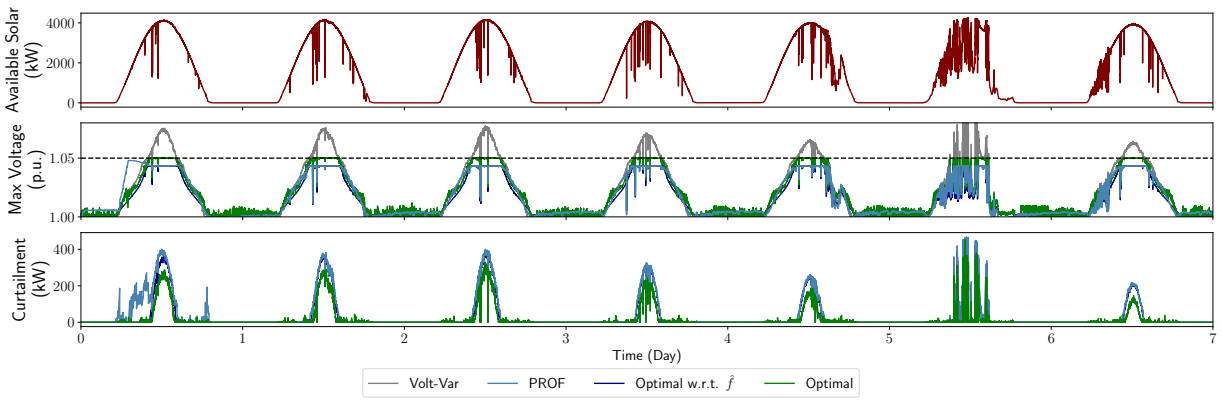


Figure 8.6: PROF satisfies voltage constraints throughout the experiment, and learns to minimize curtailment as well as possible within its conservative safety set, $\hat{C}_k$, after learning safely for a day.

where the sparsity pattern of H is characterized by the admittance matrix. We jointly project actions from all inverters at each time step k onto the constraints $\mathcal{U}(k) \cap \mathcal{X}$.

8.6.2 Implementation details

We evaluate PROF by executing it once over the 1-week dataset (at 1-second intervals). Similarly to other quasi-static approaches, we update the policy every 15-minutes. Similarly to [GKJ20], we optimize the neural policy with stochastic samples by directly differentiating through the objective (Equation (8.16)) and the linearized grid model (Equation (8.15)). However our method differs in that Gupta, Kekatos, and Jin [GKJ20] characterized the constraints as a regularization term, and learned the policy via primal-dual updates. We incorporate the constraints directly via the differentiable projection layer and thus guarantee constraint satisfaction.

We use λ=10 (see Equation (8.13)), a learning rate of 10^{-3}, and RMSprop [TH12] as the optimizer. At every 15 minutes, we sample 16 batches of data with size of 64 from the replay memory. We keep a replay memory size of 86400, i.e., samples from the previous day. For the both the utility-level network and the inverter-level network, we use fully-connected layers with ReLU activations. The utility-level network has hidden layer sizes (256, 128, 64), and each inverter-level network has hidden layer sizes (16, 4) and outputs active and reactive power. On top of the neural network, we implement the differentiable projection layer, following the constraints described in Section 8.6.1.

We compare our methods to three baselines, (1) a Volt/Var strategy following IEEE 1547.8 [Bas14], (2) the optimal solution with respect to the linearized grid model, and (3) the optimal solution with respect to the true AC power flow equations.

8.6.3 Results

The performance of PROF in comparison to the three baselines is summarized in Figure 8.6. For clarity, we only show the maximum voltage over all buses; under-voltage is not a concern for this particular test case.

We see that the Volt/Var strategy violates voltage constraints 22.3% of time, mostly around noon when the solar generation is high and there is a surplus of energy. Since the Volt/Var baseline does not adjust active power, there is no curtailment.

In comparison, PROF satisfies the voltage constraints throughout the experiment, even with a randomly initialized neural policy. While PROF performs poorly on the first morning, it quickly improves its policy. In fact, the behavior of PROF is barely distinguishable from the optimal solution with respect to the linearized grid model, after learning safely for a day. This implies that PROF learned to control inverters as well as possible given its approximate model, which constructs a conservative under-approximation of the true safety set.

The optimal baseline with respect to the true AC power flow equations unsurprisingly achieves the best performance with respect to minimizing curtailment, as it can push the maximum voltage to the allowable limit in order to maximally reduce the amount of curtailed energy. However, inverter control is a task that requires near real-time inputs, and we find that running this baseline can be prohibitively slow. Specifically, we evaluate the computation time of different operations by averaging over 1000 randomly sampled problems from our dataset on a personal laptop. For PROF, on average, a forward pass in the neural network (excluding the projection layer) took 4.5 ms and the differentiable projection operation took 8.6 ms. The computation cost of the differentiable projection could be further reduced by using customized projection solvers such as the ones in [AK17; Don+21b] that avoid the "canonicalization" costs introduced by general-purpose solvers such as the one we use [Agr+19]. In comparison, solving the optimization baseline with respect to the true AC power flow equations took 1.02s on the same machine, which is even longer than the 1s control time-step.

8.7 Conclusion

In this chapter, we have presented a method, PROF, for integrating convex operational constraints into neural network policies for energy systems applications. In particular, we propose a policy that entails passing the output of a neural network to a differentiable projection layer, which enforces a convex approximation of the operational constraints. These convex constraint sets are obtained using approximate models of the system dynamics, which can be fit using system data and/or constructed using domain knowledge. We can then train the resultant neural policy via standard RL algorithms, using an augmented cost function designed to effect desirable policy gradients. The result is that our neural policy is cognizant of relevant operational constraints during learning, enhancing overall performance.

We find in both the building energy optimization and inverter control settings that PROF successfully enforces relevant constraints while improving performance on the control objective. In particular, in the building thermal control setting, we find that our approach achieves a 4% energy savings over the state of the art while largely maintaining the temperature within the deadband. In the inverter control setting, our method perfectly satisfies the voltage constraints over more than half a million time steps, while learning to minimize curtailment as much as possible within the safety set.

While these results demonstrate the promise of our method, a key limitation is in

its computational cost. In particular, computing a projection during every forward pass of training and inference is decidedly more expensive than running a "standard" neural network. A fruitful area for future work – both in the context of our method, and in the context of research in differentiable optimization layers as a whole – may be to improve the speed of such differentiable projection layers. For instance, this might entail developing special-purpose differentiable solvers [AK17; Don+21b] for optimization problems commonly encountered in energy systems applications, developing approximate solvers that do not rely on obtaining optimal solutions in order to compute reasonable gradients, or employing cheaper projection schemes such as α-projection [Sha+20] where possible.

Additionally, the success of our method (and many other constraint enforcement methods) depends fundamentally on the quality of the approximate model used to characterize the constraint sets. In particular, this determines the extent to which the resultant approximate constraint sets are a good representation of the true operational constraints. While we were able to employ reasonably high-quality approximation schemes in the context of this work, future work on safely updating the models or the constraint sets directly [Fis+19] may greatly improve the quality of the solutions.

More generally, while our work highlights one approach to enforcing physical constraints within learning-based methods, we believe this is only the start of a broader conversation on closely integrating domain knowledge and control constraints into learning-based methods. In particular, strictly enforcing physical constraints will be paramount to the real-world success of these methods in energy systems contexts, and we hope that this chapter will serve to spark further inquiry into this important line of work.

Part III

Implicit Differentiation in Power Systems

Chapter 9

Inverse OPF: Assessing the Vulnerability of Power Grid Data

Exposure of critical power grid information could threaten power market efficiency and cybersecurity. It is thus in the best interests of power grid operators to assess what information may be exposed. We formulate an algorithm called inverse optimal power flow to assess the extent to which private power grid data is exposed by publicly-available data. Our algorithm exploits the fact that private and public information are related via the AC optimal power flow optimization problem, and employs implicit differentiation through this problem to explore the private parameter space. We find that we are able to learn private information such as electricity generation costs and (to some extent) grid structural parameters on a 14-bus test case. We seek to share this information with grid operators to aid in their vulnerability assessments.

The work in this chapter has previously been published as a workshop paper:

> Priya L. Donti, Inês Lima Azevedo, and J. Zico Kolter. "Inverse Optimal Power Flow: Assessing the Vulnerability of Power Grid Data." *NeurIPS Workshop on AI for Social Good* (2018).

9.1 Introduction

In the electricity sector, there is a great need to protect critical market and structural information that could compromise efficient electricity market operation or power grid cybersecurity. For instance, an electricity generator that gains information about other generators' costs could bid strategically to increase profits, potentially increasing electricity prices for consumers [Wol03; MW09]. As another example, an adversary who gains information about grid structure could intentionally cause a power outage [Wat03]. It is thus in grid operators' best interests to assess whether critical information is exposed, and then act to prevent this exposure from affecting efficient and safe power system operation.

At the same time, grid operators such as PJM and governmental entities such as the Environmental Protection Agency regularly publish quantities such as five-minute electricity prices [PJM18] and hourly power outputs of electricity generators [Uni18] for the purposes of market transparency and emissions monitoring. While this published information is not sensitive in and of itself, it is possible that individuals could use it to "reverse-engineer" critical market information. We investigate the question of whether and to what extent critical power grid information is exposed by published information, given our knowledge that these private and public quantities are related via AC optimal power flow (ACOPF; see Section 2.3.2). To do this, we formulate an algorithm called *inverse optimal power flow* (inverse OPF) that uses a neural network to learn private quantities from public quantities.

We first describe related work, including the formulation of ACOPF. We then describe our inverse OPF approach, which involves computing gradients through the ACOPF optimization problem. We show that our method can learn cost parameters on a 14-bus test case and shows promise in learning some grid structural parameters. We seek to share our results with grid operators so they may better protect against system vulnerabilities effected by the exposure of critical information.

9.2 Related work

Power system vulnerability analysis. Prior work has assessed the power system's vulnerability to electricity market gaming and cybersecurity attacks. For instance, [Wol03; Mon17] retroactively analyze the efficiency of power market operation in specific United States power markets, and work in the area of mechanism design [SWZ01] attempts to *proactively* design markets that will operate efficiently. Other work has attempted to assess cybersecurity threats to grid stability and reliability [Wat03; SHG+12; Yan+12], especially with the increasing use of smart devices on the grid. Our work is complementary to this body of research, as our analysis of critical data exposure can serve as an input to such vulnerability analyses.

Inverse problems. Inverse problems seek to predict model inputs or decision parameters from model outputs, with examples in machine learning including inverse reinforcement learning [NR+00], inverse imaging problems [MJU17], and deep network applications [Li+18]. Within power systems, prior work has used techniques from game theory, graph

theory, and bi-level optimization to identify power grid structure [Yua+16] and energy demands [AZL18]. We seek to bridge techniques from these two communities by proposing a method to solve inverse power flow problems within a neural network.

9.3 AC optimal power flow formulation

We now present the relevant formulation of AC optimal power flow (ACOPF) used in this work, which plays a crucial role in our formulation of inverse optimal power flow. As described in Section 2.3.2, ACOPF is solved by power system operators to pick power system quantities that minimize the overall cost of delivering power. Here, we use the same formulation of ACOPF previously given in Equation (2.13), but modify the notation for simplicity. In particular, we define the decision variable $z \equiv \begin{bmatrix} p_g^T & q_g^T & |v|^T & \text{angle}(v)^T \end{bmatrix}^T$ over the power injections and voltages at each node. As in Section 6.4.3, we define our generator cost function as $f_c(p_g) = p_g^T \operatorname{diag}(c_q) p_g + c_a^T p_g$, where $c_q, c_a \in \mathbb{R}^b_{\geq 0}$ are quadratic and linear cost parameters, respectively, for power generation at each node; we use $c = \begin{bmatrix} c_q^T & c_a^T \end{bmatrix}^T$ to refer collectively to these cost parameters. We collect all device limits (2.13b) into the linear inequality constraints $Gz \leq h$, and all assignments of known quantities into the linear equality constraints $Az = b$ (see also Section B.1). The ACOPF problem can then be written as:

$$\begin{aligned} \underset{z \equiv \begin{bmatrix} p_g^T & q_g^T & |v|^T & \text{angle}(v)^T \end{bmatrix}^T}{\text{minimize}} \quad & p_g^T \operatorname{diag}(c_q) p_g + c_a^T p_g && \text{(9.1a)} \\ \text{subject to} \quad & Az = b && \text{(9.1b)} \\ & Gz \leq h && \text{(9.1c)} \\ & (p_g - p_d) + (q_g - q_d) i = \operatorname{diag}(v) \bar{W} \bar{v}. && \text{(9.1d)} \end{aligned}$$

We recall also that the dual variable $\lambda \in \mathbb{R}^n$ on the power flow constraint (9.1d) corresponds to the electricity prices at each bus.

For the purposes of taking gradients through the ACOPF problem (as we will later need to do), we linearize the power flow constraint using its Jacobian J at some guess z_0 [WWS14], and then solve the resulting problem iteratively via sequential quadratic programming [BT95]. Specifically, we write the linearized quadratic program corresponding to (9.1) as

$$\begin{aligned} \underset{z \equiv \begin{bmatrix} p_g^T & q_g^T & |v|^T & \text{angle}(v)^T \end{bmatrix}^T}{\text{minimize}} \quad & p_g^T \operatorname{diag}(c_q) p_g + c_a^T p_g \\ \text{subject to} \quad & \tilde{A} z = \tilde{b} \\ & Gz \leq h, \end{aligned} \tag{9.2}$$

where $\tilde{A} = \begin{bmatrix} A^T & J(z_0)^T \end{bmatrix}^T$ and $\tilde{b} = \begin{bmatrix} b^T & k(z_0)^T \end{bmatrix}^T$ collect both the original linear constraints and the linearized power flow constraint $J(z_0) z = k(z_0)$.

Algorithm 5 Inverse OPF Optimization

input: $\{(p_g^{(i)}, \lambda^{(i)}) \mid i = 1, \ldots, m\}$ *// public data*
initialize $\hat{c}, \hat{W}$ *// some initial guess*
for $t = 1, \ldots, T$ **do**
  compute $\ell_{\text{pub}} := \sum_{i=1}^{m} \ell\left((p_g^{(i)}, \lambda^{(i)}), (\hat{p_g}^{(i)}, \hat{\lambda}^{(i)})\right)$
  // update guesses if loss has not converged
  if $\ell_{\text{pub}} \neq 0$ **then**
    update $\hat{c}$ with $\nabla_{\hat{c}} \ell_{\text{pub}}$
    update $\hat{W}$ with $\nabla_{\hat{W}} \ell_{\text{pub}}$
  else
    return $\hat{c}, \hat{W}$
  end if
end for

9.4 Inverse optimal power flow

We now describe our inverse optimal power flow algorithm, which attempts to learn private electricity grid information from public information via ACOPF. Specifically, given public information on real powers p_g, p_d and electricity prices λ, we seek to estimate private generator cost parameters c and the nodal admittance matrix W, where all variables are as described in Section 9.3. We do so by constructing estimates $\hat{c}^\star$ and $\hat{W}^\star$ of c and W, respectively, whose corresponding ACOPF outputs are close to the true values of the publicly-available quantities p_g and λ. Mathematically, this problem can be formulated under some loss function ℓ on publicly-available quantities as

$$\begin{aligned} \hat{c}^\star, \hat{W}^\star = \underset{\hat{c}, \hat{W}}{\text{argmin}} \;\; & \ell\left((p_g, \lambda), (\hat{p_g}, \hat{\lambda})\right) \\ \text{subject to} \;\; & \hat{p_g}, \hat{\lambda} = \text{ACOPF}(\hat{c}, \hat{W}, p_d), \end{aligned} \tag{9.3}$$

where the constraint denotes that $\hat{p_g}$ and $\hat{\lambda}$ are the values of generator power injections and power prices produced by solving the ACOPF problem (9.1) with cost parameters $\hat{c}$, admittance matrix $\hat{W}$, and nodal power demands p_d. We solve this problem iteratively via Algorithm 5, using backpropagation within a neural network to compute the needed gradients. We note that while Equation (9.3) maximizes the agreement between true and estimated *public* quantities, the objective of actual interest is the agreement between the true and estimated *private* quantities. However, there are potentially multiple distinct sets of inputs to ACOPF that would produce identical public outputs. Thus, we must use enough data when executing Algorithm 5 to ensure that there is a unique set of private parameters that can produce the correct public outputs across *all* input datapoints.

Optimizing the inverse OPF problem. The main technical challenge of this approach is in computing the gradients $\nabla_\theta \ell$ for each $\theta \in \{\hat{c}, \hat{W}\}$, as this involves taking the gradient

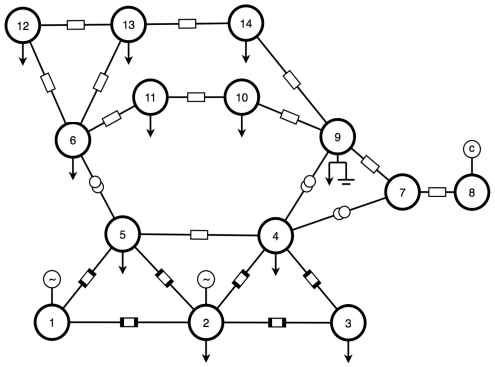


Figure 9.1: The modified version of the IEEE 14-bus test case with three generators located at nodes 1, 2, and 8, respectively, on which we run our experiments. The power generation costs for each generator are $f_1(p_{g_1}) = 2p_{g_1}^2 + 5p_{g_1}$, $f_2(p_{g_2}) = 4p_{g_2}^2 + 2p_{g_2}$, and $f_8(p_{g_8}) = 5p_{g_8}^2 + 1p_{g_8}$. Admittance matrix parameters can be found via Kolter [Kol13].

through the solutions to ACOPF. Specifically, we must compute

$$\frac{\mathrm{d}\ell}{\mathrm{d}\theta} = \frac{\partial \ell}{\partial \hat{p}_g(\theta)} \frac{\mathrm{d}\hat{p}_g(\theta)}{\mathrm{d}\theta} + \frac{\partial \ell}{\partial \hat{\lambda}(\theta)} \frac{\mathrm{d}\hat{\lambda}(\theta)}{\mathrm{d}\theta}, \tag{9.4}$$

where $\frac{\mathrm{d}\hat{p}_g(\theta)}{\mathrm{d}\theta}$ and $\frac{\mathrm{d}\hat{\lambda}(\theta)}{\mathrm{d}\theta}$ are the Jacobians of optimal primal and dual variables, respectively, in problem (9.1), with respect to our parameter estimate θ (and where we denote the dependence of $\hat{p}_g$ and $\hat{\lambda}$ on each θ here explicitly). To compute these Jacobians, we use the method presented in Amos and Kolter [AK17] to take gradients through the optimal quadratic program (9.2) solved during the last iteration of sequential quadratic programming. As also described in Section 2.2.3, at a high level, this involves differentiating through the KKT optimality conditions of (9.2) and using the implicit function theorem to get a set of linear equations we can solve to get the necessary gradients.

9.5 Experiments

We test our algorithm on a modified version of the IEEE 14-bus test case [Kol13] with three generators located at nodes 1, 2, and 8. More details about this system are shown in Figure 9.1. We construct our neural network using PyTorch [Pas+19], and make custom modifications to the qpth quadratic programming library [AK17] to account for the fact that the backward pass vector $\frac{\mathrm{d}\hat{\lambda}(\theta)}{\mathrm{d}\theta}$ associated with the dual variable $\hat{\lambda}(\theta)$ is nonzero. We train this network on up to 201 public outputs generated from the Grid Optimization (GO) Competition simulations [ARP18], using the Adam optimizer [KB15]. Our loss is $\ell((p_g, \lambda), (\hat{p}_g, \hat{\lambda})) = 100\|p_g - \hat{p}_g\|_2^2 + \|\lambda - \hat{\lambda}\|_2^2$ for (9.3), where the weighting term adjusts for differences in orders of magnitude between p_g and λ.

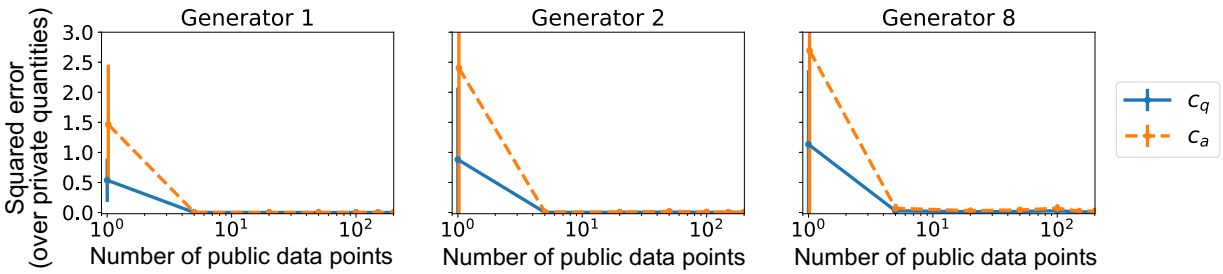


Figure 9.2: Squared error of guesses for quadratic (c_q) and linear (c_a) generator costs when all generators' costs are unknown (lower is better). Each plotted point represents five runs over a given amount of public data. We find that all cost parameters are identifiable with as little as 5 datapoints.

Cost parameters. We test the scenario in which all electricity generation costs are unknown (but the admittance matrix is known). Results for runs over different amounts of training data are in Figure 9.2, with initial guesses for each cost parameter sampled from a Gaussian distribution to encode market participants' prior knowledge of cost distributions. We find that we are able to completely learn the cost parameters for this system with as little as 5 public datapoints. Even though our test system is small, given that real power grid data is published with hourly granularity (i.e. 8760 datapoints per year), there is cause to believe that publicly-available data may expose generator cost parameters on the actual power system as well.

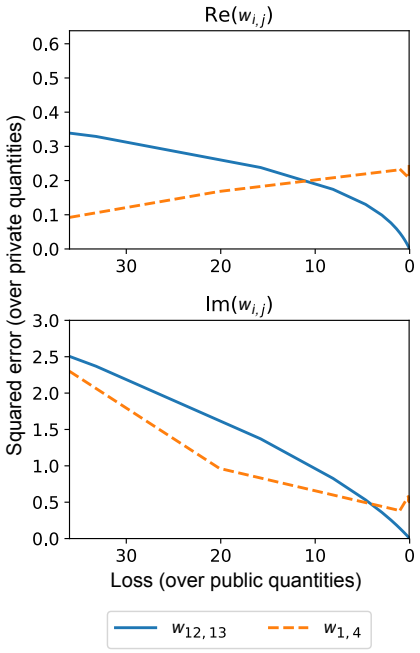


Figure 9.3: Error of sample admittances (real and imaginary parts plotted separately) as training loss on public data goes to zero. Our estimate for $W_{12,13}$ converges, but for $W_{1,4}$ diverges.

Admittance matrix parameters. Admittance matrix parameters (*admittances*) are potentially harder to learn than costs, as the choice of admittances can potentially render problem (9.1) infeasible before or during training. In our experiments, we test whether we can learn one admittance parameter at a time, where our initial guess involves perturbing this parameter with Gaussian noise reflecting the variability across all admittances. (We assume all other parameters are known.) As shown via illustrative results in Figure 9.3, our preliminary tests suggest that some admittances are readily identifiable while others may be harder to identify.

9.6 Conclusion

We find that public power grid data may expose private data. Future work includes a more thorough investigation of admittances on the 14-bus system, as well as assessments

on larger systems. These assessments can aid policymakers as they explore options for data publication, market design, and cybersecurity. While we address the case of power systems here, our method could be applied to *any* setting in which private and public information are related via a known optimization problem; extension of our method to other such settings also remains as important future work.

Chapter 10

Adversarial Robustness for Security-Constrained and Stochastic OPF

In this chapter, we combine innovations from the areas of adversarially robust learning and implicit layers to tackle the problems of N-k security-constrained optimal power flow (N-k SCOPF) and stochastic optimal power flow (stochastic OPF), two core problems for the operation of power grids. N-k SCOPF aims to schedule power generation in a manner that is robust to potentially k simultaneous equipment outages, and is an important part of strategies to address correlated failures in the face of climate extremes. Stochastic OPF aims to schedule power generation with consideration of the stochastic nature of loads and variable renewable energy resources, and is critical for the integration of variable renewable energy into power grids. However, these problems have often been viewed as prohibitively expensive to solve at realistic scale, limiting their practical use. To address this, inspired by methods in adversarially robust training, we frame N-k SCOPF and stochastic OPF as minimax optimization problems – viewing power generation settings as adjustable parameters, and equipment outages or instantiations of stochastic variables as (adversarial) attacks – and solve these problems via gradient-based techniques. The loss functions of these minimax problems involve resolving implicit equations representing grid physics and operational decisions, which we differentiate through via the implicit function theorem. We demonstrate the efficacy of our framework on realistic-scale (5,000- and 10,000-bus) systems under the true (nonlinear) AC network constraints.

The work in this chapter has previously been published in:

> Priya L. Donti*, Aayushya Agarwal*, Neeraj Vijay Bedmutha, Larry Pileggi, and J. Zico Kolter. "Adversarially Robust Learning for Security-Constrained Optimal Power Flow." *Advances in Neural Information Processing Systems* 34 (2021), 28677–28689.

> Aayushya Agarwal, Priya L. Donti, J. Zico Kolter, and Larry Pileggi. "Employing Adversarial Robustness Techniques for Large-Scale Stochastic Optimal Power Flow." *Power Systems Computation Conference* (2022).

10.1 Introduction

Robust optimization problems are pervasive across many applications and domains – such as electric power systems, supply chain management, and civil engineering – where the goal is to construct some solution that is robust under any allowable instantiation of uncertainty [BTEGN09; BBC11; GYH15]. While the aim is generally that these solutions be *provably* robust, there unfortunately remain many settings where it is either not easy or not possible to construct such solutions. This has often motivated the use of heuristic approaches. For instance, many approaches in adversarially robust deep learning formulate neural network training as a minimax game over neural network parameters and input perturbations, optimizing this problem via gradient-based techniques that do not yield provable robustness guarantees, but are nonetheless effective in practice [KM18].

In this chapter, we draw inspiration from adversarially robust training to address the problem of N-k security-constrained optimal power flow (N-k SCOPF). N-k SCOPF is a fundamental problem to schedule power generation in a way that is robust to k potential equipment failures (e.g., generator or line outages). Unfortunately, N-k SCOPF is prohibitively expensive to solve at scale under the true AC network equations, leading grid operators to use rough approximations in practice. To address this challenge, we frame N-k SCOPF as a minimax attacker-defender problem, where the "defender" aims to schedule power generation, and the "attacker" aims to pick adversarial equipment failures. The loss function of this problem requires solving implicit equations representing the physics of the electric grid as well as additional operational decisions that are made after an attack has occurred. As such, we optimize this problem using gradient-based techniques, and employ insights from the literature on implicit differentiation and implicit layers to cheaply compute gradients through the loss function.

Building on this work, we further employ these ideas within the arena of stochastic optimization [SP07], specifically for the problem of stochastic optimal power flow (stochastic OPF) under AC network constraints. Stochastic OPF aims to schedule power generation to minimize expected costs under stochastic loads (e.g., power consumption and variable renewable energy production), and is critical for the integration of renewable energy into power grids. However, this problem (like N-k SCOPF) is often expensive to solve at scale, which means that in practice, the grid analysis methods used today are largely deterministic. To address this, we re-frame stochastic OPF as a minimax attacker-defender problem, and solve it using the same techniques that we developed for N-k SCOPF.

Our key contributions are:

- **Formulation for minimax optimization with implicit variables.** To streamline the presentation of concepts, we provide a generic formulation for gradient-based optimization of minimax problems with implicitly-defined variables. While our main focus in this chapter is on N-k SCOPF and stochastic OPF, we believe this generic formulation may also be of broader interest for minimax settings with physics in the loop, as well as for tri-level optimization settings.

- **Formulation for gradient-based optimization of N-k SCOPF.** We rewrite N-k SCOPF as a continuous minimax optimization problem, and demonstrate how to efficiently compute gradients through relevant implicit components. We also utilize the underlying structure of our optimization solvers to further streamline the outer minimization procedure. Importantly, the per-iteration cost of this approach is agnostic to the number of simultaneous outages k, despite the combinatorial blowup in the number of associated "contingency scenarios."
- **Realistic-scale demonstration on N-1, N-2, and N-3 SCOPF.** We demonstrate the efficacy of our method in addressing SCOPF settings that allow for one, two, or three simultaneous outages on a realistic 4622-bus AC power system with over 38 billion potential N-3 outage scenarios. We find that our method incurs 3-4× fewer N-3 feasibility violations than a baseline AC optimal power flow approach, and requires only 21 minutes to run on a standard laptop.
- **Formulation for gradient-based optimization of stochastic OPF.** Building on our work for N-k SCOPF, we write stochastic OPF as a minimax optimization problem and solve it using efficient gradient-based techniques.
- **Realistic-scale demonstration on stochastic OPF.** We demonstrate that our method maintains AC feasibility over a wide range of probabilistic scenarios, and demonstrate scalability by determining a dispatch for a synthetic 11,000 bus system.

10.2 Related work

Adversarial robustness in deep learning. There has been a growing body of work that aims to parameterize neural networks in a manner that is robust to particular perturbations of their inputs, usually by casting neural network training as an attacker-defender game [KM18; Xu+20]. While there have been several promising approaches for *certifiably* robust neural network training [Won+18; RSL18; WK18], in general, these approaches do not yet scale to large-sized networks and only address a limited set of threat models. As a result, there has been a lot of research in this area that aims to train robust neural networks using approximate, gradient-based training methods [GSS15; Mad+18], an approach we adopt here. In addition, a key part of this literature has been on constructing strong but cheap-to-compute attacks that can strengthen the outcomes of adversarially robust training, e.g., the fast gradient sign method (FGSM) [GSS15] and projected gradient descent (PGD) attacks [Mad+18]. In our experiments, we similarly show how a gradient-based adversarial robustness approach can be used to identify potential grid vulnerabilities or worst-case instantiations of stochasticity, as an input to secure power system optimization.

Security-constrained optimal power flow. In the electric power systems community, there has been a great deal of emphasis on optimizing power grid operations to be secure to sets of outages (*contingencies*) that may be particularly high risk. For instance, many grid operators in the United States require grids to be operated in a way that is N-1 secure (i.e., secure against any single outage), which has led to a focus in the literature on addressing N-

1 SCOPF [BBM20; CSM18; Yan+21; ARP19]. However, ensuring security against *multiple* simultaneous failures (i.e., solving N-k SCOPF for $k > 1$) is becoming increasingly critical, both as evidenced by recent major blackout events [Mor21; Sto19] and as climate change drives weather extremes [IPC21] that may lead to correlated outages [Mur19]. That said, due to the computational complexity of addressing N-k SCOPF in the general case, there have been few attempts at developing methods geared towards this setting. In particular, the computational complexity of N-k SCOPF grows combinatorially with k and the size of the system. Some previous attempts to solve N-k SCOPF have employed exhaustive methods [MSA14], Bender's cuts to reduce the number of contingencies analyzed [WWG13], and bi-level optimization frameworks [WWG13; HCZ17]. In particular, [HCZ17] used bi-level optimization to develop a systematic attacker-defender approach to address N-3 contingency scenarios, but used a simplified, linear power grid model to attain convergence. We similarly adopt a bi-level framework, but solve a realistic nonlinear AC model of the grid by introducing fast gradient calculation methods inspired by the implicit layers literature, which allows us to scale our approach to a 4622-bus system.

Stochastic optimal power flow. Driven by an accelerated roadmap to integrate renewable energy, power systems must increasingly contend with large sources of stochasticity [Bak+19]. As such, several methods have recently been proposed to improve the computational tractability of stochastic grid optimization problems, such as stochastic OPF. These methods have primarily aimed to address or circumvent two main challenges: a) the challenge of obtaining cheap and representative samples of stochastic variables, and b) the challenge of accommodating nonlinear power grid constraints associated with ACOPF.

On the challenge of sampling, previous work has taken two separate approaches. The first has involved developing sampling methods that can efficiently generate representative samples from the stochastic distribution [MMD20; Vra+13] or identifying "worst-case" stochastic disruptions to reduce the sample set within a minimax formulation [JW16; Jab20; Arr+19; ZC15; Dvo+14]. While these approaches significantly reduce the required number of samples, optimizing over even a reduced sample set is computationally challenging to scale to larger systems. The second approach attempts to avoid sampling altogether by forming analytical approximations to stochastic OPF and using these to, e.g., encode the probability of violating network constraints within the optimization objective [RA17; Müh+19] or minimize risk metrics such as the conditional value at risk (CVar) [MGL17].

On the challenge of nonlinear AC constraints, many methods have relaxed the AC constraints to linear DC constraints [RA17; Guo+18; Vra+13], linearized AC constraints [YN20], or SDP relaxations [MR18]. These relaxations avoid the challenge of resolving the AC constraints and are thus fast to solve (and can be extended for use in, e.g., SCOPF). However, the obtained dispatches may not be feasible under the true AC constraints [Bak21].

10.3 Generic problem formulation

Before diving into the details of our power system optimization formulations, we first provide a more generic formulation for gradient-based minimax optimization over an implicit loss

function, which we will later build upon. In particular, we consider the setting of continuous minimax optimization problems over "defender" (minimizer) variables $x \in \mathcal{X}$ and "attacker" (maximizer) variables $y \in \mathcal{Y}$; these are also referred to as first-stage and second-stage decision variables, respectively, in the bi-level optimization literature. In addition, we allow for "third-stage" decisions $z \in \mathcal{Z}$ that are fully defined via a set of implicit constraints on x, y, and z.

Specifically, we consider problems of the form

$$\begin{aligned} &\underset{x \in \mathcal{X}}{\text{minimize}} \ \max_{y \in \mathcal{Y}} \ \ell(x, y, z) \\ &\text{s.t.} \ \ g(x, y, z) = 0, \ \ z \in \mathcal{Z}, \end{aligned} \tag{10.1}$$

where $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ are compact sets; $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}$ is a standard, continuously differentiable loss function (e.g., softmax or mean squared error loss); and $g : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}^m$ is defined such that $g(x, y, z) = 0$ is an implicit function in z with some solution $z \in \mathcal{Z}$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We further restrict ourselves to those functions g that are continuously differentiable with non-singular Jacobians at their roots, i.e., those functions that are compatible with the implicit function theorem [KP12; KM18]. We note that this formulation covers a wide range of settings, e.g., many minimax problems with nphysical constraints, or many tri-level optimization problems where z is a solution to a continuous optimization problem parameterized by x and y (both of which notions we will use in Section 10.4 for the setting of N-k SCOPF).

Inspired by the literature on adversarial robustness in deep learning, we propose to solve problem (10.1) via gradient-based search on both the inner maximization and outer minimization problems. In particular, this entails (a) obtaining some (approximately) optimal y for the inner maximization problem via gradient-based techniques, given some initial value of x, (b) updating x using the gradient at the optimum of the inner maximization problem, and (c) repeating these steps until convergence. We now describe steps (a) and (b) in additional detail.

10.3.1 Attack: Solving the inner maximization problem

Let $\bar{x}$ denote some fixed value for x. The inner maximization problem is then given by

$$\max_{y \in \mathcal{Y}} \ \ell(\bar{x}, y, z) \ \ \text{s.t.} \ \ g(\bar{x}, y, z) = 0, \ \ z \in \mathcal{Z}. \tag{10.2}$$

We optimize this problem via projected gradient descent. Specifically, let $y = y_0$ denote our initial guess for the optimal attack, and let $\mathcal{P}$ denote the projection operator. Until convergence (or for some fixed number of iterations), we then

(i) Obtain $z^\star$ such that $g(\bar{x}, y, z^\star) = 0, \ z^\star \in \mathcal{Z}$.

(ii) Update $y \leftarrow \mathcal{P}_{\mathcal{Y}}\left(y + \gamma \nabla_y \ell(\bar{x}, y, z^\star)\right)$ for step size γ.

Notably, step (ii) entails obtaining the gradient $\nabla_y \ell(\bar{x}, y, z^\star)$. By the chain rule, this involves the gradient through $z^\star$, which is the solution to a set of implicit equations. Specifically, using the notation d to denote total derivatives (e.g., gradients) and ∂ to denote partial derivatives, we have

$$\frac{\mathrm{d}\ell(\bar{x}, y, z^\star)}{\mathrm{d}y} = \frac{\partial \ell(\bar{x}, y, z^\star)}{\partial y} + \frac{\partial \ell(\bar{x}, y, z^\star)}{\partial z^\star} \frac{\mathrm{d}z^\star}{\mathrm{d}y}. \tag{10.3}$$

By the implicit function theorem, we can then obtain an expression for $\mathrm{d}z^\star/\mathrm{d}y$ by noting that

$$\frac{\mathrm{d}g(\bar{x}, y, z^\star)}{\mathrm{d}y} = \frac{\partial g(\bar{x}, y, z^\star)}{\partial y} + \frac{\partial g(\bar{x}, y, z^\star)}{\partial z^\star}\frac{\mathrm{d}z^\star}{\mathrm{d}y} = 0 \implies \frac{\mathrm{d}z^\star}{\mathrm{d}y} = -\left(\frac{\partial g(\bar{x}, y, z^\star)}{\partial z^\star}\right)^{-1}\frac{\partial g(\bar{x}, y, z^\star)}{\partial y}, \tag{10.4}$$

which we can plug into Equation (10.3) to yield our full update.

As discussed in Section 2.2.3, in practice, we seldom want to compute the Jacobian $\mathrm{d}z^\star/\mathrm{d}y \in \mathbb{R}^{\dim(\mathcal{Z})\times\dim(\mathcal{Y})}$ explicitly due to the potentially large time and space complexity of doing so; instead, it is often desirable to compute the left vector-matrix product $(\partial\ell/\partial z^\star)(\mathrm{d}z^\star/\mathrm{d}y) \in \mathbb{R}^{\dim(\mathcal{Y})}$ directly, using the "Jacobian-vector trick."

10.3.2 Defense: Taking a step in the minimization problem

Given some (approximately) optimal $y^\star$ and associated $z^\star$ from the inner optimization under the current value of $x = \bar{x}$, the outer optimization problem then becomes

$$\min_{x\in\mathcal{X}} \quad \ell(x, y^\star, z^\star) \quad \text{s. t.} \quad g(x, y^\star, z^\star) = 0. \tag{10.5}$$

One option is to then update x via a projected gradient step $x \leftarrow \mathcal{P}_{\mathcal{X}}\left(x - \beta\nabla_x\ell(x, y^\star, z^\star)\right)$ for step size β. To calculate the gradient $\nabla_x\ell(x, y^\star, z^\star)$, we note that by Danskin's theorem, we can disregard the dependence of $y^\star$ on x [KM18] (though we cannot ignore the dependence of $z^\star$). As such, we can employ a similar process as in Equations (10.3) and (10.4), where we treat $y^\star$ as constant when computing gradients with respect to x.[1] We note that while this is one potential process for updating x, we actually employ a more efficient, domain-specific process for our SCOPF procedure (see Section 10.4.4).

10.4 Addressing N-k SCOPF

Having presented this generic formulation, we now introduce our approach, CAN∂Y, for addressing N-k SCOPF.[2] In particular, we consider the problem of N-k SCOPF, where power generation must be scheduled so as to be feasible and low-cost both in the absence of equipment outages ("base case") as well as to be robust to any k simultaneous outages of power generators or lines that may occur ("contingency cases"). We note that the set of *contingencies* – i.e., allowable combinations of outages – is combinatorial in the number of potential outages, making the N-k SCOPF problem extremely computationally expensive. For instance, a realistic 4622-bus system with 6133 potential single outages has approximately 38.5 billion contingency scenarios to consider under the N-3 setting.

[1] Technically, Danskin's theorem only holds when $y^\star$ is a unique optimum of the inner maximization problem. However, in the adversarially robust training literature, the conditions of Danskin's theorem do not necessarily hold – in particular, the inner maximization problem often does not have a unique optimum, and many implementations tend to generate approximate (rather than exact) optima [GSS15; Mad+18] – but this method of computing gradients is used in practice regardless [KM18].

[2] CAN∂Y stands for "CMU Adversarial Networks with Differentiable contingencY." This name is inspired by that of SUGAR [Pan+18], whose power flow solver we differentiate through in this work.

Algorithm 6 CAN∂Y

procedure MAIN(sys) *// input: power system description*
  init dispatch x *// e.g., via base case optimal power flow*
  while not converged **do**
    $y^\star$ = ATTACK(sys, x) *// worst-case attack for current dispatch*
    update x via partial solve of Equation (10.13) using Gauss-Siedel
  end while
end procedure

procedure ATTACK(sys, x)
  init attack y
  while not converged *or* for fixed number of steps **do**
    compute $z^\star$, $s^\star$ via Equation (10.8c) *// third-stage variables*
    compute $\nabla_y \ell(x, y, z^\star, s^\star)$ via Equation (10.12) *// gradient of attack objective*
    update $y \leftarrow \mathcal{P}_{\mathcal{Y}}\left(y + \gamma \nabla_y \ell(x, y, z^\star, s^\star)\right)$ *// for attack set $\mathcal{Y}$, step size γ*
  end while
  return y
end procedure

In the rest of this section, we first more formally define the N-k SCOPF problem. We then show how we rewrite N-k SCOPF as a minimax problem of the form (10.1), in particular by forming a compact outer approximation to the contingency space. Finally, we describe how we solve this problem using a combination of gradient-based techniques and domain-specific enhancements, as summarized in Algorithm 6.

10.4.1 Defining N-k SCOPF

Let x denote the dispatch – i.e., setpoints of real power and voltage magnitude – at all power generators on the electricity system, and let $\mathcal{X}$ represent generator-wise box constraints on the dispatch. Let $\mathcal{C}$ denote the set of potential contingencies, i.e., all sets of exactly k potential outages. Finally, let $z^{(i)} \in \mathcal{Z}_i(x, c^{(i)})$ represent slightly adjusted settings of real power and voltage magnitude that the power system operator can create after scheduling x and then observing some contingency $c^{(i)} \in \mathcal{C}$, where the $\mathcal{Z}_i$ represent box constraints. Then, the N-k SCOPF problem can be expressed as

$$\begin{aligned}
&\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f_{\text{base}}(x) + \sum_{(z^{(i)},\, c^{(i)})} f_{\text{cont}}(z^{(i)}, c^{(i)}) \\
&\text{subject to} \;\; g_{\text{flow,base}}(x, w_{\text{base}}) = 0, \;\; w_{\text{base}} \in \mathcal{W}_{\text{base}} \\
&\qquad z^{(i)} \in \underset{\text{s. t. } g_{\text{flow,cont}}(z^{(i)}, w^{(i)}, x) = 0,\; w^{(i)} \in \mathcal{W}_i(x, c^{(i)})}{\operatorname{argmin}_{z^{(i)} \in \mathcal{Z}_i(x, c^{(i)})} f_{\text{cont}}(z^{(i)}, c^{(i)})} \quad \forall c^{(i)} \in \mathcal{C},
\end{aligned} \tag{10.6}$$

where $f_{\text{base}} : \mathcal{X} \to \mathbb{R}$ represents base case power production costs; $g_{\text{flow,base}} : \mathcal{X} \times \mathcal{W}_{\text{base}} \to \mathbb{R}^{n_{\text{bus}}}$ represents the nonlinear AC power flow equations in the base case, with n_{bus} being the number of power system buses; w_{base} represents electrical quantities that result from solving the base case power flow equations (e.g., reactive powers and voltage angles), with box constraints (device limits) represented by $\mathcal{W}_{\text{base}}$; and $f_{\text{cont}} : \mathcal{Z}_i \times \mathcal{C} \to \mathbb{R}$, $g_{\text{flow,cont}} : \mathcal{Z}_i \times \mathcal{W}_i \times \mathcal{X} \to \mathbb{R}^n$, and $w^{(i)} \in \mathcal{W}_i(x, c^{(i)})$ represent their respective contingency-case counterparts.

10.4.2 Rewriting N-k SCOPF as a minimax problem

We reformulate the SCOPF problem (10.6) as an attacker-defender game, where the defender must choose a dispatch that is robust to potential "worst-case" contingencies chosen by an attacker. In particular, since the contingency set $\mathcal{C}$ is discrete, we create a continuous outer approximation to this set in order to enable the use of gradient-based techniques. Specifically, let n_o be the number of generators or power lines that can potentially experience an outage. Then, for any $y \in [0,1]^{n_o}$, we define the jth entry as follows:

$$y_j = \begin{cases} 1 & \text{iff outage } j \text{ is fully active,} \\ 0 & \text{iff outage } j \text{ is not active,} \\ \alpha_j \in (0,1) & \text{iff outage } j \text{ is partially active with fraction } \alpha_j. \end{cases} \tag{10.7}$$

The first two notions presented in Equation (10.7) are standard in power systems: the generator or line pertinent to outage j is either fully operational or out of service. We newly define the notion of a partial outage with fraction α_j as one in which the power flowing through the outage device during normal operation has been reduced by a factor of α_j. For instance, we model a partial contingency on a transmission line or transformer device as reducing its admittance (i.e., ability to conduct current) by a factor of α_j. Similarly, we restrict the power produced by a generator undergoing a partial contingency by multiplying its power output by α_j.

Given these notions, we define our "threat model" for the N-k SCOPF setting to contain all vectors y with an $\mathcal{L}_1$-norm of at most k, i.e., $\mathcal{Y} := \{y : y \in [0,1]^{n_o}, \|y\|_1 \leq k\}$. Notably, the original contingency set $\mathcal{C}$ is fully represented within $\mathcal{Y}$, and in fact, all scenarios with *up to* k simultaneous potential outages are also represented. As such, $\mathcal{Y}$ represents a much broader set of potential contingencies than specified in the original problem. (Relevantly for projected gradient descent, this is also a convex set.) Using this set, we can then write our reformulation of the SCOPF problem as

$$\underset{x \in \mathcal{X}}{\text{minimize}} \max_{y \in \mathcal{Y}} \; f_{\text{base}}(x) + f_{\text{cont}}(z, y) + \frac{1}{2}\|s\|_2^2 \tag{10.8a}$$

$$\text{subject to} \;\; g_{\text{flow,base}}(x, w_{\text{base}}) = 0, \;\; w_{\text{base}} \in \mathcal{W}_{\text{base}} \tag{10.8b}$$

$$z, s \;\in\; \begin{array}{c} \operatorname{argmin}_{z \in \mathcal{Z}(x,y),\, s \in \mathbb{R}^{n_{\text{bus}}}} \; f_{\text{cont}}(z, y) + \frac{1}{2}\|s\|_2^2 \\ \text{s.t.} \;\; g_{\text{flow,cont}}(z, w_{\text{cont}}, x) + s = 0, \;\; w_{\text{cont}} \in \mathcal{W}_{\text{cont}}(x, y), \end{array} \tag{10.8c}$$

where $s \in \mathcal{S} := \mathbb{R}^{2n_{\text{bus}}}$ are slack variables representing potential infeasibilities in the third-stage optimization problem, as necessitated by the expanded contingency set. In particular,

the goal of the attacker is to now to find a set of partial outages that not only increase the cost of power generation, but also create instabilities in the grid, as captured by s. As we hinted at in Section 10.3, this is a minimax optimization problem with implicit constraints over the third-stage variables z, w_{base}, w_{cont}, and s, incorporating both nonlinear equality constraints (10.8b) as well as an optimization-based constraint (10.8c).

10.4.3 Attack stage

The inner maximization stage is responsible for finding a contingency vector $y \in \mathcal{Y}$ that maximizes the inner objective function in (10.8) for a given dispatch. We note that there are three main steps involved in executing each iteration of this process, as also shown in Algorithm 6:

1. Solving for the third-stage variables (or *network response*) $z^\star, s^\star$ due to the current instantiation of y, which requires resolving nonlinear network constraints,
2. Computing the gradient $\mathrm{d}\ell/\mathrm{d}y := \mathrm{d}\big(f_{\text{base}}(x)+f_{\text{cont}}(z,y)+\frac{1}{2}\|s\|_2^2\big)/\mathrm{d}y$,
3. Taking the projected gradient step, which involves a projection onto $\mathcal{Y}$.

In our case, the projected gradient step is cheap to compute, due to the convexity of $\mathcal{Y}$. The first two steps, however, pose potential computational bottlenecks. We therefore employ efficient methods to execute these steps, as we now describe.

10.4.3.1 Solving the network response

The AC network response due to the current instantiation of the contingency y can be determined using a steady-state power flow solver, SUGAR [PAP20], which is based on Newton-Raphson. Unfortunately, solving a power flow at each attack iteration is costly, and can easily become a bottleneck for scalability. To overcome the complexity of solving AC power flow at each iteration within the attack stage, we utilize the solution from the previous attack iteration and employ a homotopy method, known as Network-Stepping [APP21], to efficiently solve for the network response. This method uses a previously-solved network with similar topology and iteratively modifies the network using a homotopy process. At each iteration of the homotopy process, we slightly deform the network and solve the resulting problem using the previous homotopy solution as an initial condition. The result is a solution trajectory that exploits the Newton-Raphson quadratic basin of attraction to achieve fast convergence.

Specifically, we assume a known solution to a previous network $\mathcal{N}_{\text{prev}}$ with attack variables y_{prev}. For our purposes, this previous network emerges from the previous iteration of the attack procedure. To solve for the current network $\mathcal{N}$ with stochastic variables y, we develop an iterative homotopy method that incrementally changes $\mathcal{N}_{\text{prev}}$ to $\mathcal{N}$. This incremental process is controlled using a scalar homotopy factor, $t : 1 \to 0$, which incrementally changes the contingency value via the following equation:

$$y_{\text{hom}} = t y_{\text{prev}} + (1-t)y. \tag{10.9}$$

At each value of t, we solve for the state variables corresponding to the new network with contingency y_{hom}. By transforming t from 1 to 0, we are effectively transforming the network

from $\mathcal{N}_{\text{prev}}$ to $\mathcal{N}$ by iteratively changing the stochastic value from y_{prev} to y. By using the previous homotopy iteration solution as an initial condition, we remain within the Newton-Raphson quadratic basin of attraction, thereby enabling fast convergence.

10.4.3.2 Obtaining attack gradients

As described in Section 10.3.1, we aim to find the worst-case attack via projected gradient descent. In particular, we must compute the gradient of the minimax loss with respect to y, which is given by

$$\frac{\mathrm{d}\ell}{\mathrm{d}y} = \frac{\partial f_{\text{cont}}(z^\star, y)}{\partial y} + \frac{\partial f_{\text{cont}}(z^\star, y)}{\partial z^\star}\frac{\mathrm{d}z^\star}{\mathrm{d}y} + \frac{\mathrm{d}s^\star}{\mathrm{d}y}. \tag{10.10}$$

(As the base case power production cost and the base case power flow constraint (10.8b) have no dependence on y, we do not need to consider these terms during the inner maximization.)

To calculate the terms $\mathrm{d}z^\star/\mathrm{d}y$ and $\mathrm{d}s^\star/\mathrm{d}y$, we implicitly differentiate through the third-stage optimization problem (10.8c). In order to do so inexpensively, we reuse the results of computations that were executed when originally obtaining $z^\star$ and $s^\star$. More specifically, in order to obtain $z^\star$ and $s^\star$, we solve the nonlinear KKT conditions of optimization problem (10.8c) using a Newton solver that entails linearizing these equations at each iteration. More details of this Newton solver are presented in Appendix D.2 (using the more precise N-k SCOPF notation introduced in Appendix D.1). At convergence, we then implicitly differentiate through the linear fixed-point equation obtained at the last iteration:

$$J\begin{pmatrix} z^\star \\ s^\star \end{pmatrix} = b \implies \frac{\mathrm{d}J}{\mathrm{d}y}\begin{pmatrix} z^\star \\ s^\star \end{pmatrix} + J\begin{pmatrix} \frac{\mathrm{d}z^\star}{\mathrm{d}y} \\ \frac{\mathrm{d}s^\star}{\mathrm{d}y} \end{pmatrix} = \frac{\mathrm{d}b}{\mathrm{d}y} \implies \begin{pmatrix} \frac{\mathrm{d}z^\star}{\mathrm{d}y} \\ \frac{\mathrm{d}s^\star}{\mathrm{d}y} \end{pmatrix} = J^{-1}\left(-\frac{\mathrm{d}J}{\mathrm{d}y}\begin{pmatrix} z^\star \\ s^\star \end{pmatrix} + \frac{\mathrm{d}b}{\mathrm{d}y}\right), \tag{10.11}$$

where $J \in \mathbb{R}^{d\times d}$ is the Jacobian of the nonlinear KKT system and $b \in \mathbb{R}^d$ is the corresponding right hand side vector, for $d = \dim(\mathcal{Z}) + \dim(\mathcal{S})$. We note that since the system Jacobian J is in practice extremely sparse, the inverse term J^{-1} can be computed extremely efficiently using sparse LU factorization; similarly, the higher-order derivative $\mathrm{d}J/\mathrm{d}y$ is extremely sparse and can be computed efficiently. We refer the reader to [Pil98] for more details.

Substituting this result into Equation (10.10), the overall gradient of the loss is then

$$\frac{\mathrm{d}\ell}{\mathrm{d}y} = \frac{\partial f_{\text{cont}}(z^\star, y)}{\partial y} + \begin{pmatrix} \frac{\partial f_{\text{cont}}(z^\star, y)}{\partial z^\star} \\ 1 \end{pmatrix}^T J^{-1}\left(-\frac{\mathrm{d}J}{\mathrm{d}y}\begin{pmatrix} z^\star \\ s^\star \end{pmatrix} + \frac{\mathrm{d}b}{\mathrm{d}y}\right). \tag{10.12}$$

As hinted earlier, we employ the "Jacobian-vector trick" in order to efficiently compute these gradients. In particular, rather than computing the terms $\mathrm{d}z^\star/\mathrm{d}y$ and $\mathrm{d}s^\star/\mathrm{d}y$ explicitly via Equation (10.11), we directly compute their left vector-matrix product with the relevant partial derivatives of the loss – i.e., the blue term in Equation (10.12), with multiplications evaluated from left to right to ensure we are always taking matrix-vector (rather than matrix-matrix) products. Inspired by [AK17], we also reuse the LU factor from the last Newton solve we computed when obtaining $z^\star$ and $s^\star$ in order to avoid explicitly (re-)computing the matrix inverse J^{-1}. Finally, we note that for this specific problem, while the last term

$-\frac{\mathrm{d}J}{\mathrm{d}y}\begin{pmatrix} z^\star \\ s^\star \end{pmatrix} + \frac{\mathrm{d}b}{\mathrm{d}y}$ nominally involves tensor products, the relevant terms are vastly sparse due to the structure of the underlying physics – with no more than 20 nonzero entries per potential outage – making these products relatively cheap to compute in practice.

10.4.4 Defense stage

After obtaining a worst-case attack $y^\star$, our next step is to adjust our dispatch $x \in \mathcal{X}$ in response. As described in Section 10.3.2 for the generic setting, one option to do this involves taking a projected gradient step in x. However, we adopt a different approach for N-k SCOPF, due to the practical requirements of this setting. In particular, a general system requirement is that the base case power flow equations $g_{\text{flow,base}}(\cdot) = 0$ must remain feasible under the dispatch $x \in \mathcal{X}$, as the most likely scenario is that no contingency will occur. However, projecting onto this (nonlinear, non-convex) set of constraints can be expensive.

As a result, we instead note that we can rewrite the N-k SCOPF minimax problem (10.8) as a single minimization problem:

$$\begin{aligned} \underset{x\in\mathcal{X},\, z\in\mathcal{Z}(x,y^\star),\, s\in\mathbb{R}^{n_{\text{bus}}}}{\text{minimize}} \quad & f_{\text{base}}(x) + f_{\text{cont}}(z, y^\star) + \frac{1}{2}\|s\|_2^2 \\ \text{subject to} \quad & g_{\text{flow,base}}(x, w_{\text{base}}) = 0, \quad w_{\text{base}} \in \mathcal{W}_{\text{base}} \\ & g_{\text{flow,cont}}(z, w_{\text{cont}}, x) + s = 0, \; w_{\text{cont}} \in \mathcal{W}_{\text{cont}}(x, y^\star). \end{aligned} \tag{10.13}$$

To determine our next iterate of x, our strategy is then to *partially* solve this optimization problem by running one step of a nonlinear Gauss-Seidel method, and then keep the value of x obtained from that step. This allows us to incrementally update x in a direction that is more robust to the worst-case attack $y^\star$, while still maintaining the feasibility of the base case power flow equations.

Importantly, we are able to run this procedure efficiently, as we can reuse the results of existing computations that were executed when obtaining the optimal attack $y^\star$. In particular, as we describe in more detail in Appendix D.3, we can split the KKT conditions of the problem (10.13) into two groups:

$$\begin{pmatrix} \partial\mathcal{L}/\partial x \\ \partial\mathcal{L}/\partial\lambda_{\text{base}} \end{pmatrix} \equiv F_{\text{base}}(x, w_{\text{base}}, \lambda_{\text{base}}) + \begin{pmatrix} \left(\partial g_{\text{flow,cont}}(z, w_{\text{cont}}, x)/\partial x\right)^T \lambda_{\text{cont}} \\ 0 \end{pmatrix} = 0 \tag{10.14a}$$

$$\begin{pmatrix} \partial\mathcal{L}/\partial z \\ \partial\mathcal{L}/\partial s \\ \partial\mathcal{L}/\partial\lambda_{\text{cont}} \end{pmatrix} \equiv F_{\text{cont}}(z, w_{\text{cont}}, \lambda_{\text{cont}}) + \begin{pmatrix} 0 \\ 0 \\ g_{\text{flow,cont}}(z, w_{\text{cont}}, x) \end{pmatrix} = 0, \tag{10.14b}$$

where $\mathcal{L}$ denotes the Lagrangian of problem (10.13), and λ_{base} and λ_{cont} are the dual variables on the base case and contingency power flow constraints, respectively. We note that the two terms F_{base} and F_{cont} are independent in terms of their inputs, but are weakly coupled via the additional terms (which represent a sparse set of ramping constraints and voltage setpoints that tie together the base and contingency cases). This is an ideal setup for decoupling through nonlinear Gauss-Seidel solution methods. In addition, the

contingency-related KKT conditions (10.14b) are actually identical to the KKT conditions of the problem (10.8c) that we solved during the last iteration of the inner maximization problem; as a result, we can reuse the result of this previous computation when executing our Gauss-Seidel step. Together, this allows us to inexpensively identify an update direction for x that nonetheless remains feasible with respect to the base case power flow constraints.

10.5 Experiments for N-k SCOPF

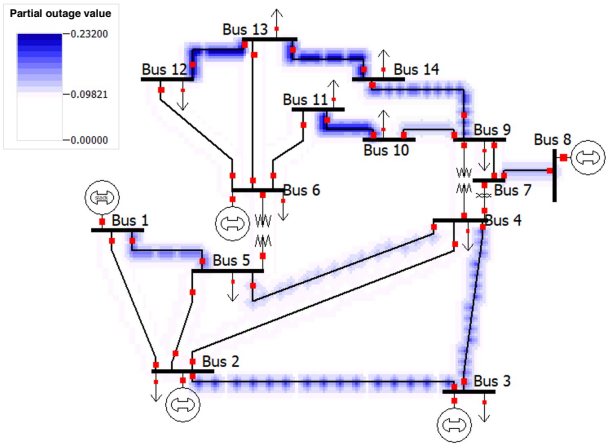
We demonstrate the efficacy of our approach on the settings of N-1, N-2 and N-3 SCOPF. In particular, noting that quality of adversarial attacks is likely to have a large effect on the success of our overall procedure, we first visualize the attacks found by our inner maximization process (Sections 10.3.1, 10.4.4) on a small power system test case. We then demonstrate the performance of our overall approach on a realistic 4622-bus power system with approximately 6 thousand potential N-1 contingencies, almost 19 million N-2 contingencies, and over 38 *billion* N-3 contingencies. We show that CAN∂Y is able to efficiently find solutions that are competitive with other leading approaches in the N-1 case, while reducing violations in the N-2 and N-3 scenarios compared to a base case optimal power flow.

All experiments are run on a single core of a Macbook Pro with a 2.6 GHz Core i7 CPU. We implement our approach in Python, using a custom optimal power flow solver called SUGAR [PAP20] to compute optimization (10.8c), and CVXPY [DB16] to compute convex projections for projected gradient descent. We evaluate all dispatch solutions using PowerWorld, a commercial power flow tool.

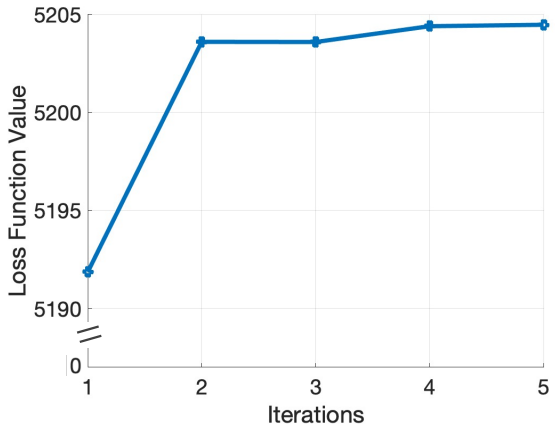
10.5.1 Illustrative adversarial attack

Finding worst-case contingencies is often beneficial to power systems engineers, who try to identify fragile areas of their grid for future development. Traditionally, engineers use linear approximations of the grid physics [KT16; CLL19; Has+17] to identify single outages that pose a significant risk to system stability. However, given that the underlying physics are fundamentally nonlinear, such linear approximations quickly become inaccurate when trying to identify the risks associated with multiple simultaneous outages. Our implicit differentiation approach, on the other hand, employs accurate gradient information from the physics of the network to quickly identify contingencies that maximally increase our loss function (or an alternative loss function of choice that also captures system infeasibilities).

For ease of visualization, we demonstrate this attack-identification approach on the IEEE 14-bus test system. In particular, we identify an adversarial N-2 contingency on this system in just 5 iterations (approximately one minute), increasing the value of the loss function by 3% over the base case scenario, as shown in Figure 10.1b. This worst-case contingency represents a combination of multiple partial outages on different lines, and (perhaps surprisingly) does not include any generator outages, as shown in Figure 10.1a. This is likely to present a stronger attack than those obtained via the "standard" linear approximation approach, serving as a potential benefit to power system planners who are



(a) Visualization of the worst-case contingency found, with the degree of the associated partial outage on each line indicated in blue. (Plot generated via PowerWorld.)



(b) Training curve for finding a worst-case contingency. The process converges within 5 iterations, and increases the loss by 3%.

Figure 10.1: Illustrative example of finding a worst-case N-2 contingency on a 14-bus test system.

Table 10.1: Comparison of the performance of our method against top-performing submissions to the ARPA-E GO Competition, which addresses N-1 SCOPF (lower scores are better). Results are shown for the 4622-bus Challenge 1 test case ("5K network"). While the score comparisons shown are inexact due to subtleties of the evaluation metric (see Appendix D.4), at a high level, we see that CAN∂Y performs competitively with all top-scoring methods.

	gollnl	GO-SNIP	GMI-GO	BAT	gravityx	CAN∂Y*
GO Challenge 1 Rank	1	2	3	4	5	-
Score for 5K network	546,302	553,152	553,328	545,783	550,020	552,032

trying to reinforce their grid, as well as to "adversarially robust training" procedures like ours.

10.5.2 Validating N-1 security

Today, most grid operators in the United States require that their dispatch be N-1 secure, i.e., secure against any single outage, prompting the development of associated methods. In particular, the recent Grid Optimization (GO) Competition [ARP19], hosted by ARPA-E, focused on finding algorithms to solve N-1 SCOPF. Each participating team used a variety of methods to produce a dispatch that was evaluated on the basis of power cost and feasibility in both the base and contingency cases. In order to validate that our method works well in the N-1 setting, we solve a particular case from the competition – namely, the 4622-bus test case with a sub-selection of 3071 N-1 potential contingencies, provided as part of the Challenge 1 stage – by constructing our relaxed contingency set $\mathcal{Y}$ with $k = 1$.

We find that our score is comparable against the top approaches submitted to the GO Competition, as shown in Table 10.1. We note that these score comparisons are not

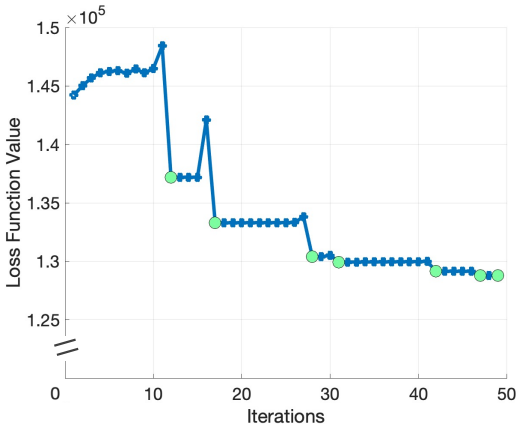


Figure 10.2: Loss vs. iterations in adversarial training. Each attack stage is at most 10 iterations and each defense stage is one iteration. The loss value after the defense step is in green.

Contingency type	N-1	N-2	N-3
Scenarios tested	6,133	359,712	428,730
OPF viol.	59	10,572	4,086
PowerModels viol.	37	4,005	5,391
CAN∂Y viol. (ours)	36	3,580	1,122

Table 10.2: Number of feasibility violations incurred by the N-3 SCOPF version of our method, a baseline optimal power flow (OPF), and the PowerModels N-1 SCOPF solver [Cof+18] on randomly selected N-1, N-2, N-3 contingency scenarios for a 4622-bus test case. We see that CAN∂Y reduces the number of N-2 and N-3 violations by a factor of 3-4× over the OPF baseline, and the number of N-3 violations by a factor of 5× over the PowerModels solution.

exact, as our power flow solver uses a more realistic model for coupling between the base and contingency cases than was posed in Challenge 1, which affects the way that the evaluation metric is computed (see Appendix D.4). Nonetheless, at a high level, these results demonstrate that our method performs competitively with respect to the top-performing methods for solving N-1 SCOPF.

10.5.3 Improving N-3 SCOPF

We now describe the performance of our method on our main setting of interest: N-3 SCOPF. While other competitive methods exist for solving N-1 SCOPF, previous work has struggled to approach settings allowing for larger numbers of simultaneous outages (e.g., N-k SCOPF for $k = 2$ or 3) due to the associated combinatorial explosion in problem size. Our method, however, scales gracefully with respect to the number of allowable simultaneous outages, as we need only tweak the value of k used within our attack set $\mathcal{Y} := \{y : y \in [0,1]^{n_o}, \|y\|_1 \leq k\}$. More specifically, each iteration of the attack maximization and each defense step calculation take approximately the same amount of time *regardless of the value of* k, given that the costs of the gradient computations, projections, and (optimal) power flow solves are independent of k. (The total number of iterations it takes for our method to converge may vary between settings, though we do not notice a substantive difference in this respect between the N-1, N-2, and N-3 versions of our approach during our experiments.)

We use our method to attempt to solve N-3 SCOPF (i.e., set $k = 3$) on a 4622-bus test case over all 6133 potential outages (i.e., over 38 billion N-3 contingency scenarios); the associated training curve is shown in Figure 10.2. In total, our approach takes only 21 minutes to converge.

We evaluate the strength of our obtained dispatch in maintaining feasibility against a combination of N-1, N-2, and N-3 contingency scenarios, which are all technically contained within the threat model represented by our choice of $\mathcal{Y}$. We note that while full security

against all these contingencies is likely impossible with a single dispatch – e.g., we can very often construct an N-3 contingency that isolates, or *islands*, some non-self-sustaining part of the electrical grid – we aim to demonstrate that our method can improve upon existing methods in terms of providing robustness against a wide variety of scenarios. As there remain a lack of available N-2 or N-3 SCOPF methods against which we can readily compare, we compare our performance against that of a base case optimal power flow (OPF) solver, as well as the open-source PowerModels N-1 SCOPF algorithm [Cof+18]. Due to the intractability of evaluating these dispatches on *all* possible contingency scenarios, we randomly sub-select the set of N-1, N-2, and N-3 scenarios on which we evaluate, and run these evaluations in PowerWorld over the course of several days.

The results of our evaluation are shown in Table 10.2. Overall, we see that CAN∂Y significantly reduces the number of total contingency violations as compared to the OPF solution by a factor of 3-4×. While the PowerModels baseline and our approach perform comparably with respect to N-1 and N-2 contingency violations, our approach incurs nearly 5× fewer N-3 violations as compared to PowerModels. Analyzing the N-3 contingencies in more detail, we find that of the 4086 specific violations incurred by OPF, 931 of those were also incurred by CAN∂Y (while the remaining 191 violations incurred by our method were disjoint). Overall, these results indicate that our method is much more effective than OPF and a benchmark N-1 SCOPF solver at guarding against N-2 and N-3 contingencies, though the actual distribution of specific contingencies that are guarded against may differ between these methods.

10.6 Addressing stochastic OPF

We now employ similar ideas as in Section 10.4 to address the problem of stochastic optimal power flow. Here, we define our setting of stochastic OPF setup and show how we rewrite stochastic OPF as a minimax problem. As the methods for solving the attack and defense stages are very similar to those presented in Sections 10.4.3–10.4.4, we do not detail these here, but instead refer the reader to [Aga+22].

10.6.1 Defining stochastic OPF

We consider a two-stage formulation of stochastic optimal power flow, in which 1) a dispatch is determined and enacted based on a point forecast of each stochastic variable, and then 2) generators engage in a real-time recourse strategy (e.g., via automatic generation control or primary frequency response) upon realization of the stochastic variables, in order to correct for deviations from the initial forecast. This kind of formulation is common in the stochastic OPF literature [JW16], and is also similar to pre-contingency and post-contingency logic in security-constrained OPF [Hol+21]. Given this formulation, our goal during the first stage is to find a dispatch that minimizes the cost of dispatched power generation, plus the expected cost of any second-stage actions and infeasibilities under some model of grid stochasticity.

Mathematically, let $x_d \coloneqq \begin{bmatrix} p_g^T & p_g^T & v_{\text{set}}^T & v_r^T & v_i^T \end{bmatrix}^T$ denote the state variables associated with the initial dispatched network. This includes the active power p_g, reactive power p_g,

and voltage magnitude set points v_{set} at all synchronous generators, as well as the real and imaginary voltages v_r and v_i at all buses. Let $\alpha \sim \mathcal{D}$ denote stochastic variables on the grid drawn from a distribution $\mathcal{D}$, with $\text{supp}(\mathcal{D})$ denoting the support of the distribution; in general, α is a vector of multiplicative factors representing deviations from the point forecasts of the active and reactive powers of stochastic loads, as well as of renewable generation; α is embedded into each stochastic device model. Finally, $x_\alpha \coloneqq \begin{bmatrix} p_{g_\alpha}^T & q_{g_\alpha}^T & v_{\text{set}_\alpha}^T & v_{r_\alpha}^T & v_{i_\alpha}^T \end{bmatrix}^T$ denotes the second-stage state variables under any realization of uncertainty $\alpha \sim \mathcal{D}$. The stochastic OPF problem is then:

$$\min_{p_g, v_{\text{set}}} \quad f_d(x_d) + \mathbf{E}_{\alpha \sim \mathcal{D}}(f_a(x_\alpha)) \tag{10.15a}$$

$$\text{s.t.} \quad g_d(x_d) = 0 \tag{10.15b}$$

$$h_d(x_d) \leq 0 \tag{10.15c}$$

$$x_\alpha \in \underset{\text{s.t. } g_a(x_d, x_\alpha, \alpha) = 0 \quad \forall \alpha \in \text{supp}(\mathcal{D}),}{\text{argmin}_{x_\alpha} f_a(x_\alpha)} \tag{10.15d}$$
$$h_a(x_d, x_\alpha, \alpha) \leq 0$$

where f_d and f_a denote first-stage and second-stage cost functions, respectively; the equality constraints $g_d(x_d) = 0$ and $g_a(x_d, x_\alpha, \alpha) = 0$ capture network flow conservation equations during the first and second stages, respectively; $h_d(x_d) \leq 0$ and $h_a(x_d, x_\alpha, \alpha) \leq 0$ represent operating device constraints during the first and second stages, respectively; and (10.15d) thereby represents the second-stage response.

While (10.15) presents generalized notation for simplicity of exposition, in this work, we specifically employ a current-voltage formulation (expressed in detail in [PAP20]) for network flows (i.e., g_d and g_a capture Kirchhoff's Current Law) and employ the same grid model and second-stage response definition as in the ARPA-E GO Competition [Hol+21]. Notably, the second-stage constraints capture automatic responses including automatic generation control, and include higher voltage and current bounds than in the base network. As a result, the constraints, $g_a(x_d, x_\alpha, \alpha) = 0$ and $h_a(x_d, x_\alpha, \alpha) \leq 0$ in (10.15d) couple the first-stage and second-stage networks. While we refer readers to [Hol+21] for the full second-stage response formulation, we highlight that generators are allowed to ramp their active power generation within certain bounds dictated by the dispatch, such that

$$p_{\text{ramp, min}} \leq p_{g_\alpha} - p_g \leq p_{\text{ramp, max}}, \tag{10.16}$$

where p_{g_α} is the active power at all synchronous generators as generated in the second-stage response for a given $\alpha \sim \mathcal{D}$, and $p_{\text{ramp, max}}$ and $p_{\text{ramp, min}}$ are the upper and lower ramping constraints, respectively; this is reflected within the second-stage inequality constraints in (10.15d). Generators are also allowed to adjust local voltage parameters in a way that maintains the voltage magnitude set during dispatch (i.e., $v_{\text{set}} = v_{\text{set}_\alpha}$), such that

$$v_{\text{set}} = \sqrt{v_{g,r_\alpha}^2 + v_{g,i_\alpha}^2}, \tag{10.17}$$

where v_{g,r_α} and v_{g,i_α} are the second-stage real and imaginary bus voltages at the synchronous generators for a given $\alpha \sim \mathcal{D}$; this is reflected within the second-stage equality constraints in (10.15d).

The first-stage cost function reflects the quadratic cost for active power generation, as shown in (10.18). Similarly, the second-stage cost function is the quadratic cost of the second-stage active power generation p_{g_α}, as shown in (10.19). Letting c_1, c_2, and c_3 represent vectors of quadratic, linear, and constant cost coefficients at all synchronous generators, our cost functions are then:

$$f_d(x_d) = p_g^T \operatorname{diag}(c_1) p_g + c_2^T p_g + c_3, \tag{10.18}$$

$$f_a(x_\alpha) = p_{g_\alpha}^T \operatorname{diag}(c_1) p_{g_\alpha} + c_2^T p_{g_\alpha} + c_3. \tag{10.19}$$

The distribution $\mathcal{D}$ from which the stochastic variables α are drawn can be approximated from historical data using various techniques (see, e.g., [MMD20]). As $\mathcal{D}$ is a continuous space, this represents a large set of stochastic scenarios, which presents a potential bottleneck for scalably solving the stochastic ACOPF formulation in its original form. We therefore propose a proxy formulation that can be solved efficiently.

10.6.2 Rewriting stochastic OPF as a minimax problem

To make the stochastic OPF problem amendable to robust optimization techniques, we reframe the problem as an attacker-defender problem, where a "defender" must first formulate a dispatch, after which an "attacker" can pick a worst-case instantiation of the stochastic variables for that dispatch. In particular, our formulation minimizes the effect of a "worst-case" stochastic scenario while considering the probability $P(\alpha)$ of such an event occurring, as:

$$\min_{p_g, v_{\text{set}}} \quad f_d(x_d) + \eta \max_{\alpha \in \operatorname{supp}(\mathcal{D})} P(\alpha) f_a(x_\alpha) \tag{10.20a}$$

$$\text{s.t.} \quad g_d(x_d) = 0, \quad h_d(x_d) \leq 0 \tag{10.20b}$$

$$x_\alpha \in \begin{array}{c} \operatorname{argmin}_{x_\alpha} f_a(x_\alpha) \\ \text{s.t. } g_a(x_d, x_\alpha, \alpha) = 0,\ h_a(x_d, x_\alpha, \alpha) \leq 0, \end{array} \tag{10.20c}$$

where $\eta = \int_{\alpha \in \operatorname{supp}(\mathcal{D})} \mathrm{d}\alpha$. The value of η is chosen such that the objective of the inner maximization in (10.20a) represents an upper bound on the expectation in (10.15b); that is:

$$\begin{aligned} \mathbf{E}_{\alpha \sim \mathcal{D}}(f_a(x_\alpha)) &= \int_{\alpha \in \operatorname{supp}(\mathcal{D})} P(\alpha) f_a(x_\alpha) \mathrm{d}\alpha \\ &\leq \int_{\alpha \in \operatorname{supp}(\mathcal{D})} \left(\max_{\alpha' \in \operatorname{supp}(\mathcal{D})} P(\alpha') f_a(x_{\alpha'}) \right) \mathrm{d}\alpha \\ &= \eta \max_{\alpha \in \operatorname{supp}(\mathcal{D})} P(\alpha) f_a(x_\alpha). \end{aligned} \tag{10.21}$$

In practice, again drawing on the robust optimization literature, we further modify this formulation to incorporate "infeasibility currents" $\iota_\alpha^{\text{slack}}$ that are added to locate any infeasibilities in the second-stage network, as shown in (10.22). (These infeasibility currents are included within the second-stage state vector x_α.) In particular, we add a term to our

second-stage cost to penalize infeasibilities with a weighting factor $w_{\text{slack}} > 0$, and modify the equality constraint in (10.20c) to (10.23). The weighting factor, w_{slack}, adjusts the relative scaling between the economic cost of power generation in (10.19) and the cost of an infeasibility as shown in the multi-objective function (10.22). Generally, a large weighting of $w_{\text{slack}} > 1000$, as used by previous works, ensures that the solution using the modified objective in (10.22) will have $\iota_\alpha^{\text{slack}} = 0$ (or within a small tolerance of zero) when the system is in fact feasible.

$$f_a(x_\alpha) = p_{g_\alpha}^T \operatorname{diag}(c_1) p_{g_\alpha} + c_2^T p_{g_\alpha} + c_3 + w_{\text{slack}} \|\iota_\alpha^{\text{slack}}\|_2^2. \tag{10.22}$$

$$g_a(x_d, x_\alpha, \alpha) + \iota_\alpha^{\text{slack}} = 0. \tag{10.23}$$

While minimizing the upper bound rather than the integral (i.e., expectation) is prone to different dispatch solutions, the goal of formulating this objective function is to prioritize robustness to any realizations of uncertainty with large expected costs (which often represent infeasible networks with large $\iota_\alpha^{\text{slack}}$). This ultimately is most important to operations and planning engineers. We demonstrate the impact of employing our proxy formulation (10.20)–(10.23), as opposed to the original stochastic formulation (10.15), in Section 10.7.1.

10.7 Experiments for stochastic OPF

We now study the performance of our method on various realistic networks ranging in size from 73 buses to 11,615 buses. We begin by validating the proxy minimax reformulation (10.20) of the traditional stochastic OPF (10.15) by comparing our method against an open-source, scenario-based chance constrained stochastic OPF solver [MMD20] on two smaller test-cases. Importantly, our method not only achieves a similar dispatch to the open-source stochastic OPF solver but also exhibits superior convergence time. We then highlight the scalability of our method by optimizing the dispatch of five realistic test cases from the ARPA-E GO competition [Hol+21] to be robust against stochastic noise within appropriate runtime. We further study one of the realistic systems in depth to highlight our method's efficacy in achieving a dispatch that is robust against stochastic noise compared to a deterministic OPF solver. Our experiments are run on a single core on a Macbook Pro with a 2.6Ghz Core i7 CPU. The approach is developed on top of the SUGAR optimal power flow solver [PAP20]; we thus refer to our method as "Stochastic SUGAR." We use CVXPY [DB16] to solve the projection within the projected gradient descent step.

10.7.1 Validating the minimax reformulation

We validate our minimax reformulation (10.20), solved using the techniques presented above, by comparing our results to those of an open-source scenario-driven, chance-constrained OPF method [MMD20] that solves the original problem in (10.15). Specifically, we test these methods on two publicly available test cases that are 73 buses and 1,354 buses [Fli+13] in size (both available at [IEE22]) which have stochastic generation and loads drawn from a uniform distribution allowing for a 10% deviation from the nominal values, as defined in

Table 10.3: Comparison of Stochastic SUGAR (ours) with an open-source stochastic OPF tool [MMD20].

Network	Method runtime (s)		$\max \lvert p_g - \hat{p_g} \rvert$ (p.u.)
	Stochastic SUGAR	Open-Source Stochastic OPF	
73_ieee	48	289	0.0013
1354_pegasse	202	385	0.0082

Table 10.4: Network information for five networks provided by [Hol+21].

	Network 1	Network 2	Network 3	Network 4	Network 5
# buses	500	3,022	4,918	8,733	11,615
# generators	224	637	1340	743	806
# loads	281	1,793	3,070	3,399	19,356

[MMD20]. We highlight the comparison between our tool (Stochastic SUGAR) and the open-source stochastic OPF tool in Table 10.3.

To validate that the reformulation achieves a similar dispatch as solving the original problem for these testcases, we measure the difference in the dispatch between the tools as the maximum difference in the active power generation ($p_g - \hat{p_g}$), where p_g is the vector of active power generated from the Stochastic SUGAR tool and $\hat{p_g}$ is the vector of active power generation from the open-source stochastic OPF tool. We notice a minimal difference in the active power dispatch, indicating that the minimax reformulation is capable to achieving a similar solution as tools solving the original stochastic OPF formulation. Importantly, Table 10.3 also highlights the runtime improvements of our methodology, which achieves a nearly 1.5× speedup over the open-source stochastic OPF tool.

10.7.2 Scaling to realistic networks

After experimentally validating the minimax reformulation, we scale the methodology to solve realistic testcases provided by the ARPA-E GO competition [ARP21] ranging from 500 buses to 11,000 buses. The number of generators, buses and loads for each case are shown in Table 10.4. We treat the loads in these testcases as stochastic sources; specifically, the active and reactive powers (q_l and p_l) of each load are each modeled via independent normal distributions centered at the nominal values provided in the original test case, and with standard deviations that are 10% of the nominal values.

10.7.2.1 Improvement over deterministic solver

We begin by validating our approach on the 500-bus network from the ARPA-E GO Competition [ARP21] with 281 stochastic loads as detailed in Table 10.4.

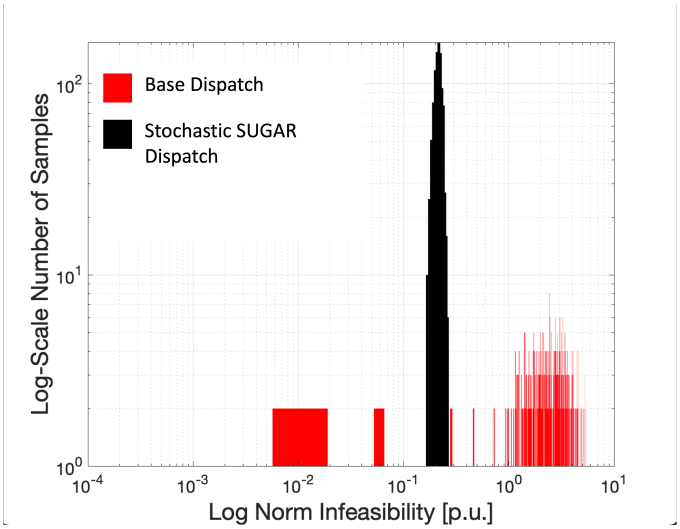


Figure 10.3: Monte-Carlo analysis for infeasibilities over load distribution for base ACOPF dispatch (red) and the Stochastic SUGAR dispatch (black).

Our optimization of the stochastic network converges in 10 defense iterations, taking a total of 73 seconds. In particular, drawing inspiration from the literature on adversarial robustness in deep learning [Mad+18; GSS15], we only take one gradient step in the inner maximization during each attack iteration (rather than converging to a true optimum) to improve runtime.

To validate that our approach indeed improves robustness against stochastic sources, we run a Monte-Carlo analysis for our optimized dispatch obtained via Stochastic SUGAR, as well as for a dispatch obtained via deterministic ACOPF. In this analysis, we obtain different realizations of α, and then solve the second-stage problem (10.20c) to obtain the second-stage state variables; both Monte-Carlo analyses use identical second-stage models and parameters (including slack currents) for solving each scenario. Figure 10.3 illustrates the norm of the infeasibility values over 5000 samples of the load distribution for both Stochastic SUGAR and the base ACOPF dispatch. An infeasible case is defined as a scenario with a norm infeasibility of over a value of 1 (i.e., any norm infeasibility of less than 1 is within the tolerance of the Newton-Raphson solver).

We notice that (as expected) the dispatch obtained by the deterministic ACOPF solver leads to a number of infeasible cases. In contrast, Stochastic SUGAR avoids any larger infeasibilities. This is to be expected, as our formulation explicitly minimizes the upper bound on the expected cost of any stochastic scenario. A side effect of this approach is that there are also fewer samples with small infeasibilities; however, it is important to note that the vast majority of the samples seen by Stochastic SUGAR have sufficiently small infeasibilities such that the network is considered to be convergent.

10.7.2.2 Scalability study

After validating our approach, we now demonstrate its scalability over the various realistic-scale networks. Figure 10.4 highlights the simulation runtime for the five networks. We note

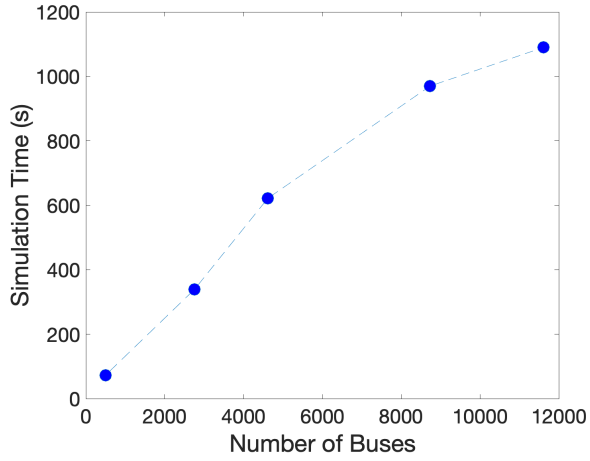


Figure 10.4: Simulation time (s) to solve stochastic ACOPF for various networks using Stochastic SUGAR.

that even on the largest network with over 11,000 buses, our approach takes a total of only 18 minutes to run, which includes time to parse the case data and write the results. We also see that the runtime of our method scales linearly to sub-linearly with the size of the network. This occurs because the larger networks we use tend to have more inertial capacity to account for fluctuations in the grid, which means that less of a change from the original ACOPF dispatch (with which we initialize our approach) is required; that is, our approach often needs fewer iterations to converge for larger networks, even though each iteration is more costly.

10.8 Conclusion

In this chapter, we reformulate both N-k SCOPF and stochastic OPF as attacker-defender minimax optimization problems, and employ algorithms inspired by adversarial robustness in deep learning, implicit layers, and circuit simulation to scalably solve these problems. In particular, for N-k SCOPF, we show for a realistic 4622-bus test case with over 38 billion potential N-3 contingencies, our approach takes only 21 minutes to converge on a standard laptop. Withiin this time, our approach reduces N-3 feasibility violations by a factor of 3-4$\times$ compared to a baseline optimal power flow method and by almost 5$\times$ compared to a baseline N-1 SCOPF solver. For stochastic OPF, we demonstrate the efficacy of our method by improving the robustness of a dispatch on a 500-bus system under load uncertainty within 73 seconds, and showing that our method can scale to larger systems of upwards of 11,000 buses while staying within reasonable time limits. Overall, we believe this demonstrates the promise of our approach in enabling scalable robust and stochastic optimization on realistic-scale power grids.

We note that the success of our minimax optimization approach is likely highly reliant on the strength of the adversarial attacks that we generate, just as in the adversarially robust training literature [KM18; Xu+20]. As such, a fruitful direction for future work may involve developing improved procedures for obtaining adversarial attacks in the context of N-k SCOPF and stochastic OPF. In addition, given the large scale of the power networks

we consider, it is generally impossible to evaluate proposed dispatches against the full suite of potential contingencies in order to check whether they are indeed N-k secure or robust against all potential stochastic deviations. Given that, another fruitful direction may entail developing better evaluation metrics or verification procedures to inexpensively evaluate whether a proposed dispatch is (likely) robust, perhaps again drawing inspiration from the literature on verification methods for adversarially robust deep learning [Car+19].

Part IV

Conclusions and Future Directions

Chapter 11

Conclusions and Future Directions

In this thesis, we have explored several directions for the use of machine learning and related methods to address climate-relevant problems in the electric power sector. These directions illustrate the value of bridging insights from the machine learning and power systems literatures, and present takeaways for machine learning researchers, power systems researchers, power system operators, and other relevant decision-makers.

In Part I, we assessed key assumptions in the calculation of the emissions and damages avoided by power systems interventions (Chapter 3) and presented a new technique for voltage estimation in low-observability distribution systems (Chapter 4). These analyses present important takeaways for power system operators and planners. In particular, Chapter 3 demonstrates that as the power grid rapidly changes, using outdated emissions factor estimates can significantly misrepresent the benefits or damages associated with proposed power system interventions. In addition, as marginal emissions factors are increasingly being used to guide real-time demand flexibility measures, misestimating the values and temporal trends associated with the relevant emissions factors could lead to incorrect or even counterproductive load shifting actions. This underscores the importance of regularly releasing and updating the data underlying emissions factor estimates, of potentially encouraging system operators themselves to release granular emissions factor estimates, and of establishing standards that detail how to properly employ these factors; regular data releases can further inform initiatives to *forecast* emissions factors – rather than simply derive historical estimates – in order to provide important foresight to power system operations and planning strategies. Our analysis in Chapter 4 illustrates the extent to which even present levels of sensing may be sufficient to inform accurate voltage estimates on distribution systems. This implies that it is likely possible to support more active shifts from centralized power system operations paradigms towards more dynamic, distributed paradigms (e.g., to foster the integration of distributed renewable energy resources), even under today's grid monitoring infrastructure. In addition, our work has implications for decisions surrounding how much new sensing needs to be installed on distribution systems (as well as potentially the placement of such sensors), informing potential trade-offs between estimation accuracy and sensor installation costs.

In addition to takeaways for system operators and planners, these analyses further invoke several lines of inquiry for methodological research in machine learning. For instance, Chapter 3 contains multiple examples where a differing assumption drove a difference in emissions factor estimates, but those differences did not ultimately matter when evaluating a particular intervention; conversely, certain small differences in emissions factors can drive disproportionately large differences in estimated intervention effects. This yields an important question: In what scenarios do we care about the raw *accuracy* of a forecast or estimate vs. the *quality of the decision* that it enables us to make, and how can we construct forecasting and estimation methods that are appropriately cognizant of both? As an additional direction, Chapter 4 demonstrates an approach where combining data, physical constraints, and domain-informed inductive biases can provide gains in both data efficiency and performance. This potentially prompts further exploration into how machine learning methods might fruitfully leverage these different kinds of domain knowledge.

In Part II, we presented a paradigm called "optimization-in-the-loop deep learning" to enable the design of deep learning methods that are cognizant of the physics, hard constraints, and domain knowledge associated with the systems in which they operate. We showed how this paradigm can be used to design decision-cognizant forecasts (Chapter 5), feasibility-preserving neural approximators for optimization problems (Chapter 6), and reinforcement learning-based controllers with provable performance guarantees (Chapters 7–8), with the goal of addressing existing challenges in the operation of high-renewables power grids.

A key takeaway for power system operators is that it is becoming possible to leverage dynamic and data-driven methods for scalable power systems operation while retaining the same safety and stability guarantees associated with today's engineering- and physics-based methods. In fact, in certain cases, machine learning-based methods may even *improve* fidelity to system constraints – e.g., by enabling the use of formulations that maintain AC feasibility (as in Chapter 6), instead of the current practice of using physically inaccurate DC power flow models. Unfortunately, there exist several key bottlenecks to the impactful development and deployment of such techniques, notably the relative lack of infrastructure to validate and improve them (e.g., realistic metrics, benchmarks, test beds, or demonstration projects) or of mechanisms to properly integrate them with other techniques (e.g., via open source software workflows). System operators and relevant policymakers can help spur the development, maturation, deployment, and evaluation of such work by creating the needed infrastructure and implementation mechanisms, in close coordination with the research community.

There are also many fruitful methodological directions associated with this line of work. For instance, while we employ differentiable optimization techniques in this thesis for *continuous* optimization problems, many important optimization problems in power systems and beyond involve *discrete* variables. One important direction is therefore to design differentiable optimization workflows that accommodate discrete information, building for example on the literature on discrete neural representations [Duv18] and more broadly on machine learning for combinatorial optimization [BLP21]. Such techniques could be applicable to, e.g., designing fast, feasible neural approximators for unit commitment (building on the work in Chapter 6). Another important direction, relevant to the work on control in Chapters 7 and 8, entails moving from the "single-agent" setting to the "multi-agent" setting – that is,

from considering the impacts of only one controller to considering the physical and strategic interactions between many controllers. This is relevant to, e.g., power system demand response strategies in which multiple loads may dynamically shift how they use power [Ant+20; DK21], and must contend with equipment constraints, service-based constraints, partial or imperfect information exchange, and strategic behavior from different loads on the grid. In addition, while this thesis has largely assumed that the physical and decision-making characteristics of the settings we study are fully known and characterized, this is often not the case – for instance, many decision-making processes involve humans rather than well-specified optimization problems (see Chapter 5), dynamics models or robust control specifications may be incorrect or oversimplified (see Chapters 7–8), or we may only have select information about the underlying physics (as often the case in, e.g., accelerated materials science for clean technologies [But+18; Bai+18]). The challenge of building robust data-driven methods under imperfect physical knowledge is thus, in practice, an extremely important direction of work.

In Part III, we showed how scalable implicit differentiation techniques, inspired by the deep learning literature, can be used to address problems relevant to power systems policy and operations. Specifically, we briefly explored the problem of "inverse optimal power flow" to assess the vulnerability of private power grid data (Chapter 9). We also presented novel, efficient methods for addressing the problems of stochastic optimal power flow and N-k security-constrained optimal power flow, and demonstrated the efficacy of these methods on realistic-scale systems (Chapter 10). The methods in Chapter 10, in particular, can be directly employed today to enable the reliable operation of power systems under high renewables penetration and under correlated failures due to climate extremes. Our framework of combining techniques from adversarial robustness with implicit differentiation techniques may also be more broadly applicable to other power systems problems, as well as in additional domains. Relevant future directions include developing better and more scalable mechanisms for *evaluating* the solutions output by such methods (as, e.g., evaluating the robustness of a dispatch against billions of N-k scenarios can be prohibitively expensive, even if the dispatch itself is inexpensive to obtain). In addition, developing distributed versions of these methods may further facilitate scalable, decentralized power systems operations.

The work above fundamentally spans disciplines and sectors, underscoring the importance of collaborations between (e.g.) machine learning researchers, power systems researchers, system operators, and decision-makers – as well as the importance of mechanisms to foster such collaborations. Overall, we hope this thesis demonstrates how bridging insights from deep learning and electric power systems can help significantly advance methods in both fields, while addressing high-impact problems of relevance to climate action.

References

[AKLH06] Murad Abu-Khalaf, Frank L Lewis, and Jie Huang. "Policy Iterations on the Hamilton–Jacobi–Isaacs Equation for H_∞ State Feedback Control with Input Saturation." *IEEE Transactions on Automatic Control* 51.12 (2006), 1989–1995.

[AE04] Ali Abur and Antonio Gomez Exposito. *Power System State Estimation: Theory and Implementation*. CRC press, 2004.

[Ach+17] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. "Constrained Policy Optimization." *Proceedings of the 34th International Conference on Machine Learning*. 2017.

[Aga+22] Aayushya Agarwal, Priya L. Donti, J. Zico Kolter, and Larry Pileggi. "Employing Adversarial Robustness Techniques for Large-Scale Stochastic Optimal Power Flow." *Power Systems Computation Conference* (2022).

[APP21] Aayushya Agarwal, Amritanshu Pandey, and Larry Pileggi. "Fast AC Steady-State Power Grid Simulation and Optimization Using Prior Knowledge." *2021 IEEE Power & Energy Society General Meeting*. IEEE. 2021, 1–5.

[Agr+19] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. "Differentiable convex optimization layers." *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[Aka+14] Anayo K. Akametalu, Shahab Kaynama, Jaime F. Fisac, Melanie Nicole Zeilinger, Jeremy H. Gillula, and Claire J. Tomlin. "Reachability-based safe learning with Gaussian processes." *53rd IEEE Conference on Decision and Control, CDC 2014*. 2014.

[Akk+19] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. "Solving Rubik's Cube with a Robot Hand." *arXiv preprint arXiv:1910.07113* (2019).

[AES08] Mohamed H Albadi and Ehab F El-Saadany. "A summary of demand response in electricity markets." *Electric Power Systems Research* 78.11 (2008), 1989–1996.

[Alt99] Eitan Altman. *Constrained Markov Decision Processes*. Vol. 7. CRC Press, 1999.

[Amo+15] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. "Deep speech 2: End-to-end speech recognition in English and Mandarin." *arXiv preprint arXiv:1512.02595* (2015).

[AK17] Brandon Amos and J Zico Kolter. "OptNet: Differentiable optimization as a layer in neural networks." *International Conference on Machine Learning*. 2017, 136–145.

[AXK17] Brandon Amos, Lei Xu, and J Zico Kolter. "Input Convex Neural Networks." *International Conference on Machine Learning*. PMLR. 2017, 146–155.

[Ana17] Monitoring Analytics. *2017 State of the Market Report for PJM: Section 9: Interchange Transactions*. Tech. rep. Accessed: 2018-04-30. 2017.

[AZL18] James Anderson, Fengyu Zhou, and Steven H Low. "Disaggregation for Networked Power Systems." *2018 Power Systems Computation Conference (PSCC)*. IEEE. 2018, 1–7.

[Ant+20] Ioannis Antonopoulos, Valentin Robu, Benoit Couraud, Desen Kirli, Sonam Norbu, Aristides Kiprakis, David Flynn, Sergio Elizondo-Gonzalez, and Steve Wattam. "Artificial intelligence and machine learning approaches to energy demand-side response: A systematic review." *Renewable and Sustainable Energy Reviews* 130 (2020), 109899.

[ARP18] ARPA-E. *Grid Optimization Competition: Datasets*. https://gocompetition.energy.gov/content/datasets. 2018.

[ARP19] ARPA-E. *Grid Optimization (GO) Competition*. https://gocompetition.energy.gov/. 2019.

[ARP21] ARPA-E. *Grid Optimization Competition: Datasets*. https://gocompetition.energy.gov/challenges/23/datasets. 2021.

[Arr+19] Adriano Arrigo, Christos Ordoudis, Jalal Kazempour, Zacharie de Grève, Jean-François Toubeau, and François Vallée. "Optimal Power Flow Under Uncertainty: An Extensive Out-of-Sample Analysis." *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*. 2019, 1–5.

[Ast71] Karl J Astrom. *Introduction to Stochastic Control Theory*. Elsevier, 1971.

[ABP+18] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. "End-to-end differentiable physics for learning and control." *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[Aze+21] Inês Lima Azevedo, Priya L. Donti, Nathaniel C. Horner, Greg Schivley, Siler-Evans Kyle, and Parth T. Vaishnav. *Electricity Marginal Factor Estimates*. https://cedm.shinyapps.io/MarginalFactors/. Pittsburgh, PA USA, 2021.

[Bai+18] Junwen Bai, Yexiang Xue, Johan Bjorck, Ronan Le Bras, Brendan Rappazzo, Richard Bernstein, Santosh K Suram, Robert Bruce Van Dover, John M Gregoire, and Carla P Gomes. "Phase Mapper: Accelerating Materials Discovery with AI." *AI Magazine* 39.1 (2018), 15–26.

[BKK19] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "Deep equilibrium models." *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[Bak19] Kyri Baker. "Learning warm-start points for AC optimal power flow." *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2019, 1–6.

[Bak21] Kyri Baker. "Solutions of DC OPF are Never AC Feasible." *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*. 2021, 264–268.

[Bak+17] Kyri Baker, Andrey Bernstein, Emiliano Dall'Anese, and Changhong Zhao. "Network-cognizant voltage droop control for distribution grids." *IEEE Transactions on Power Systems* 33.2 (2017), 2098–2108.

[Bak+19] Jordan Bakke, Maire Boese, A Figueroa-Acevedo, B Heath, Y Li, J Okullo, AJ Prabhakar, and CH Tsai. "Renewable Integration Impact Assessment: The MISO Experience." *IAEE Energy Forum*. 2019.

[BH13] J. Bank and J. Hambrick. "Development of a high resolution, real time, distribution-level metering system and associated visualization modeling, and data analysis functions" (2013). National Renewable Energy Laboratory, Tech. Rep. NREL/TP-5500-56610.

[Ban+17] Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomlin. "Goal-driven dynamics learning via Bayesian optimization." *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, 5168–5173.

[BW89] M. Baran and F. F. Wu. "Optimal sizing of capacitors placed on a radial distribution system." *IEEE Transactions on Power Delivery* 4.1 (1989), 735–743.

[BB08] Tamer Başar and Pierre Bernhard. *H_∞-Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*. Springer Science & Business Media, 2008.

[Bas14] T.S. Basso. *IEEE 1547 and 2030 Standards for Distributed Energy Resources Interconnection and Interoperability with the Electricity Grid*. National Renewable Energy Laboratory, 2014.

[Bau96] Heinz H Bauschke. "Projection algorithms and monotone operators." PhD thesis. Dept. of Mathematics and Statistics, Simon Fraser University, 1996.

[BBM20] Mohammadhafez Bazrafshan, Kyri Baker, and Javad Mohammadi. "Computationally Efficient Solutions for Large-Scale Security-Constrained Optimal Power Flow." *arXiv preprint arXiv:2006.00585* (2020).

[BTEGN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[Ben+21] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" *FAccT*. 2021.

[Ben97] Yoshua Bengio. "Using a financial training criterion rather than a prediction criterion." *International Journal of Neural Systems* 8.04 (1997), 433–443.

[BLP21] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: A methodological tour d'horizon." *European Journal of Operational Research* 290.2 (2021), 405–421.

[Ber+17] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. "Safe Model-based Reinforcement Learning with Stability Guarantees." *Advances in Neural Information Processing Systems*. 2017.

[BD17] Andrey Bernstein and Emiliano Dall'Anese. "Linear power-flow models in multiphase distribution networks." *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE. 2017, 1–6.

[Ber+18] Andrey Bernstein, Cong Wang, Emiliano Dall'Anese, Jean-Yves Le Boudec, and Changhong Zhao. "Load flow in multiphase distribution networks: Existence, uniqueness, non-singularity and linear models." *IEEE Transactions on Power Systems* 33.6 (2018), 5832–5843.

[BBC11] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. "Theory and applications of robust optimization." *SIAM review* 53.3 (2011), 464–501.

[Beu+19] Tom Beucler, Stephan Rasp, Michael Pritchard, and Pierre Gentine. "Achieving conservation of energy in neural network emulators for climate modeling." *ICML 2019 Workshop: "Climate Change: How Can AI Help?"* (2019).

[BKV18] S. Bhela, V. Kekatos, and S. Veeramachaneni. "Enhancing Observability in Distribution Grids using Smart Meter Data." *IEEE Transactions on Smart Grid* 9.6 (2018), 5953–5961. ISSN: 1949-3053.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern Recognition and Machine Learning*. Vol. 4. 4. Springer, 2006.

[BT95] Paul T Boggs and Jon W Tolle. "Sequential Quadratic Programming." *Acta Numerica* 4 (1995), 1–51.

[BD15] Saverio Bolognani and Florian Dörfler. "Fast power system analysis via implicit linearization of the power flow manifold." *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2015, 402–409.

[BCF12] Alex Bowen, Sarah Cochrane, and Samuel Fankhauser. "Climate change, adaptation and economic growth." *Climatic change* 113.2 (2012), 95–106.

[Boy+94] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Vol. 15. Siam, 1994.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[Bre+01] Leo Breiman et al. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)." *Statistical science* 16.3 (2001), 199–231.

[Bur10] United States Census Bureau. *Population Distribution Over Time*. https://www.census.gov/history/www/reference/maps/population_distribution_over_time.html. Accessed: 2018-01-01. 2010.

[Buş+18] Lucian Buşoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko. "Reinforcement learning for control: Performance, stability, and deep approximators." *Annual Reviews in Control* 46 (2018), 8–28.

[BMB19] Enzo Busseti, Walaa M Moursi, and Stephen Boyd. "Solution refinement at regular points of conic problems." *Computational Optimization and Applications* 74.3 (2019), 627–643.

[But+18] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. "Machine learning for molecular and materials science." *Nature* 559.7715 (2018), 547–555.

[BS93] John A Buzacott and J George Shanthikumar. *Stochastic models of manufacturing systems*. Vol. 4. Prentice Hall Englewood Cliffs, NJ, 1993.

[CP10] Emmanuel J Candes and Yaniv Plan. "Matrix completion with noise." *Proceedings of the IEEE* 98.6 (2010), 925–936.

[CR09] Emmanuel J Candès and Benjamin Recht. "Exact matrix completion via convex optimization." *Foundations of Computational mathematics* 9.6 (2009), 717–772.

[Car+11] Sanya Carley, Sara Lawrence, Adrienne Brown, Andrew Nourafshan, and Elinor Benami. "Energy-based economic development." *Renewable and Sustainable Energy Reviews* 15.1 (2011), 282–295.

[Car+19] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. "On evaluating adversarial robustness." *arXiv preprint arXiv:1902.06705* (2019).

[CN13] Richard T Carson and Kevin Novan. "The private and social economics of bulk electricity storage." *Journal of Environmental Economics and Management* 66.3 (2013), 404–423.

[CSM18] Leonel de Magalhães Carvalho, Armando Martins Leite da Silva, and Vladimiro Miranda. "Security-Constrained Optimal Power Flow via Cross-Entropy Method." *IEEE Transactions on Power Systems* 33.6 (2018), 6621–6629.

[Cen17] Center for Climate and Energy Solutions. *Regulating Power Sector Carbon Emissions*. https://www.c2es.org/content/regulating-power-sector-carbon-emissions/. Accessed: 2018-01-01. 2017.

[CRG19] Ya-Chien Chang, Nima Roohi, and Sicun Gao. "Neural Lyapunov Control." *Advances in Neural Information Processing Systems*. 2019, 3245–3254.

[Cha+20] Minas Chatzos, Ferdinando Fioretto, Terrence WK Mak, and Pascal Van Hentenryck. "High-Fidelity Machine Learning Approximations of Large-Scale Optimal Power Flow." *arXiv preprint arXiv:2006.16356* (2020).

[CLL19] Liang Che, Xuan Liu, and Zuyi Li. "Screening Hidden N-k Line Contingencies in Smart Grids Using a Multi-Stage Model." *IEEE Transactions on Smart Grid* 10.2 (2019), 1280–1289.

[CCB19] Bingqing Chen, Zicheng Cai, and Mario Bergés. "Gnu-RL: A precocial reinforcement learning solution for building HVAC control using a Differentiable MPC policy." *Proceedings of the 6th ACM International Conference on Sys-*

tems for Energy-Efficient Buildings, Cities, and Transportation. 2019, 316–325.

[Che+21] Bingqing Chen*, Priya L. Donti*, Kyri Baker, J. Zico Kolter, and Mario Bergés. "Enforcing Policy Feasibility Constraints through Differentiable Projection for Energy Optimization." *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*. 2021, 199–210.

[Che+20a] Bingqing Chen, Jonathan Francis, Marco Pritoni, Soummya Kar, and Mario Berg'es. "COHORT: Coordination of Heterogeneous Thermostatically Controlled Loads for Demand Flexibility." *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 2020, 31–40.

[Che+20b] Bingqing Chen, Ming Jin, Zhe Wang, Tianzhen Hong, and Mario Bergés. "Towards Off-policy Evaluation as a Prerequisite for Real-world Reinforcement Learning in Building Control." *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities*. 2020, 52–56.

[Che+18] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. "Neural ordinary differential equations." *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[CHW22] Nicolas Christianson, Tinashe Handina, and Adam Wierman. "Chasing convex bodies and functions with black-box advice." *Conference on Learning Theory*. PMLR. 2022, 867–908.

[Cle11] Kevin A Clements. "The impact of pseudo-measurements on state estimator accuracy." *2011 IEEE Power and Energy Society General Meeting*. IEEE. 2011, 1–4.

[CB+21] Peter Clutton-Brock*, David Rolnick*, Priya L. Donti*, Lynn H. Kaack*, Tegan Maharaj, Alexandra Sasha Luccioni, Hari Prasanna Das, Cyrus Hodes, Virginia Dignum, Marta Kwiatkowska, Raja Chatila, and Nicolas Miailhe. *Climate Change and AI: Recommendations for Government Action*. Tech. rep. Global Partnership on AI, 2021.

[Cof+18] Carleton Coffrin, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin. "PowerModels. JL: An Open-Source Framework for Exploring Power Flow Formulations." June 2018, 1–8.

[CW16] Taco Cohen and Max Welling. "Group equivariant convolutional networks." *International Conference on Machine Learning (ICML)*. 2016.

[Cre16] Felix Creutzig. "Economic and ecological views on climate change mitigation with bioenergy and negative emissions." *GCB Bioenergy* 8.1 (2016), 4–10.

[DG+14] JA De Gouw, DD Parrish, GJ Frost, and M Trainer. "Reduced emissions of CO_2, NO_x, and SO_2 from US power plants owing to switch from coal to natural gas with combined cycle technology." *Earth's Future* 2.2 (2014), 75–82.

[Deh+18] Kaveh Dehghanpour, Zhaoyu Wang, Jianhui Wang, Yuxuan Yuan, and Fankun Bu. "A survey on state estimation techniques and challenges in smart distribution systems." *IEEE Transactions on Smart Grid* (2018).

[DHZ02] Youman Deng, Ying He, and Boming Zhang. "A branch-estimation-based state estimation method for radial distribution systems." *IEEE Transactions on power delivery* 17.4 (2002), 1057–1062.

[DB16] Steven Diamond and Stephen Boyd. "CVXPY: A Python-embedded modeling language for convex optimization." *The Journal of Machine Learning Research* 17.1 (2016), 2909–2913.

[DK17] Josip Djolonga and Andreas Krause. "Differentiable Learning of Submodular Models." *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[Dob+20] Roel Dobbe, Patricia Hidalgo-Gonzalez, Stavros Karagiannopoulos, Rodrigo Henriquez-Auba, Gabriela Hug, Duncan S Callaway, and Claire J Tomlin. "Learning to control in power systems: Design and analysis guidelines for concrete safety problems." *Electric Power Systems Research* 189 (2020), 106615.

[Don+20] Wenqian Dong, Zhen Xie, Gokcen Kestor, and Dong Li. "Smart-PGSim: Using Neural Network to Accelerate AC-OPF Power Grid Simulation" (2020), 1–15.

[DR09] Asen L Dontchev and R Tyrrell Rockafellar. *Implicit functions and solution mappings*. Vol. 543. Springer, 2009.

[Don+21a] Priya L. Donti*, Aayushya Agarwal*, Neeraj Vijay Bedmutha, Larry Pileggi, and J. Zico Kolter. "Adversarially Robust Learning for Security-Constrained Optimal Power Flow." *Advances in Neural Information Processing Systems* 34 (2021), 28677–28689.

[DAK17] Priya L. Donti, Brandon Amos, and J. Zico Kolter. "Task-based End-to-End Model Learning in Stochastic Optimization." *Advances in Neural Information Processing Systems*. 2017, 5490–5500.

[DAK18] Priya L. Donti, Inês Lima Azevedo, and J. Zico Kolter. "Inverse Optimal Power Flow: Assessing the Vulnerability of Power Grid Data." *NeurIPS Workshop on AI for Social Good* (2018).

[DK21] Priya L. Donti and J. Zico Kolter. "Machine Learning for Sustainable Energy Systems." *Annual Review of Environment and Resources* 46 (2021), 719–747.

[DKA19] Priya L. Donti, J. Zico Kolter, and Inês Lima Azevedo. "How Much Are We Saving After All? Characterizing the Effects of Commonly Varying Assumptions on Emissions and Damage Estimates in PJM." *Environmental Science & Technology* 53.16 (2019), 9905–9914.

[Don+19] Priya L. Donti, Yajing Liu, Andreas J. Schmitt, Andrey Bernstein, Rui Yang, and Yingchen Zhang. "Matrix Completion for Low-Observability Voltage Estimation." *IEEE Transactions on Smart Grid* 11.3 (2019), 2520–2530.

[Don+21b] Priya L. Donti, Melrose Roderick, Mahyar Fazlyab, and J. Zico Kolter. "Enforcing Robust Control Guarantees within Neural Network Policies." *International Conference on Learning Representations*. 2021.

[DRK21] Priya L. Donti*, David Rolnick*, and J. Zico Kolter. "DC3: A Learning Method for Optimization with Hard Constraints." *International Conference on Learning Representations*. 2021.

[Drg+20] Ján Drgoňa, Javier Arroyo, Iago Cupeiro Figueroa, David Blum, Krzysztof Arendt, Donghun Kim, Enric Perarnau Ollé, Juraj Oravec, Michael Wetter, Draguna L Vrabie, et al. "All you need to know about model predictive control for buildings." *Annual Reviews in Control* (2020).

[DK08] Johan Driesen and Farid Katiraei. "Design for distributed energy resources." *IEEE Power and Energy Magazine* 6.3 (2008).

[Duv18] David Duvenaud. *Course: Learning Discrete Latent Structure*. https://duvenaud.github.io/learn-discrete/. 2018.

[Dvo+14] Yury Dvorkin, Hrvoje Pandžić, Miguel A Ortega-Vazquez, and Daniel S Kirschen. "A hybrid stochastic/interval approach to transmission-constrained unit commitment." *IEEE Transactions on Power Systems* 30.2 (2014), 621–631.

[EG22] Adam N Elmachtoub and Paul Grigas. "Smart "predict, then optimize"." *Management Science* 68.1 (2022), 9–26.

[Fan+11] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. "Smart grid—The new and improved power grid: A survey." *IEEE communications surveys & tutorials* 14.4 (2011), 944–980.

[Fan86] PO Fanger. "Thermal environment—Human requirements." *Environmentalist* 6.4 (1986), 275–278.

[Fed15] Federal Energy Regulatory Commission. "Energy Primer: A Handbook of Energy Market Basics." *Federal Energy Regulatory Commission: Washington, DC, USA* (2015).

[FAR09] Yantao Feng, Brian DO Anderson, and Michael Rotkowitz. "A game theoretic algorithm to compute local stabilizing solutions to HJBI equations in nonlinear H_∞ control." *Automatica* 45.4 (2009), 881–888.

[Fer+17] Nicholas EP Fernandez, Srinivas Katipamula, Weimin Wang, YuLong Xie, Mingjie Zhao, and Charles D Corbin. *Impacts of commercial building controls on energy savings and peak load reduction*. Tech. rep. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2017.

[FIN18] FINESCE. *About FINESCE*. http://www.finesce.eu/. Accessed: 2018-01-01. Jan. 2018.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" (2017), 1126–1135.

[FMVH20] Ferdinando Fioretto, Terrence WK Mak, and Pascal Van Hentenryck. "Predicting AC optimal power flows: Combining deep learning and lagrangian dual methods." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 01. 2020, 630–637.

[Fis+19] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. "Bridging Hamilton-Jacobi safety analysis and reinforcement learning." *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, 8550–8556.

[FA17] Michael J Fisher and Jay Apt. "Emissions and Economics of Behind-the-Meter Electricity Storage." *Environmental Science & Technology* 51.3 (2017), 1094–1101.

[Fli+13] Stéphane Fliscounakis, Patrick Panciatici, Florin Capitanescu, and Louis Wehenkel. "Contingency ranking with respect to overloads in very large power systems taking into account uncertainty, preventive, and corrective actions." *IEEE Transactions on Power Systems* 28.4 (2013), 4909–4917.

[FNC20] Thomas Frerix, Matthias Nießner, and Daniel Cremers. "Homogeneous linear inequality constraints for neural network activations." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, 748–749.

[FB17] Stefan R Friedrich and Martin Buss. "A robust stability approach to robot reinforcement learning based on a parameterization of stabilizing controllers." *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, 3365–3372.

[Gao+16] Pengzhi Gao, Meng Wang, Scott G Ghiocel, Joe H Chow, Bruce Fardanesh, and George Stefopoulos. "Missing data recovery by exploiting low-dimensionality in power system synchrophasor measurements." *IEEE Transactions on Power Systems* 31.2 (2016), 1006–1013.

[GD20] Timnit Gebru and Emily Denton. *Tutorial on Fairness Accountability Transparency and Ethics in Computer Vision at CVPR 2020*. https://sites.google.com/view/fatecv-tutorial. 2020.

[Gen+18] C. Genes, I. Esnaola, S. M. Perlaza, L. F. Ochoa, and D. Coca. "Robust Recovery of Missing Data in Electricity Distribution Systems." *IEEE Transactions on Smart Grid* (2018), 1–1. ISSN: 1949-3053.

[GAJ14] Nathaniel Gilbraith, Inês L Azevedo, and Paulina Jaramillo. "Evaluating the Benefits of Commercial Building Energy Codes and Improving Federal Incentives for Code Adoption." *Environmental Science & Technology* 48.24 (2014), 14121–14130.

[Gla19] Mevludin Glavic. "(Deep) Reinforcement learning for electric power system control and related problems: A short review and perspectives." *Annual Reviews in Control* 48 (2019), 22–35.

[Goo+16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. MIT Press Cambridge, 2016.

[GSS15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *International Conference on Learning Representations* (2015).

[GYH15] Bram L Gorissen, İhsan Yanıkoğlu, and Dick den Hertog. "A practical guide to robust optimization." *Omega* 53 (2015), 124–137.

[Gou+16] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. "On differentiating parameterized argmin and argmax problems with application to bi-level optimization." *arXiv preprint arXiv:1607.05447* (2016).

[GHC21] Stephen Gould, Richard Hartley, and Dylan Campbell. "Deep declarative networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.8 (2021), 3988–4004.

[GZKM14] Joshua S Graff Zivin, Matthew J Kotchen, and Erin T Mansur. "Spatial and temporal heterogeneity of marginal emissions: Implications for electric cars and other electricity-shifting policies." *Journal of Economic Behavior & Organization* 107 (2014), 248–268.

[GJ14] Alex Graves and Navdeep Jaitly. "Towards End-To-End Speech Recognition with Recurrent Neural Networks." *International Conference on Machine Learning*. Vol. 14. 2014, 1764–1772.

[GDY19] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. "Hamiltonian neural networks." *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[GW08] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[Gro11] David Gross. "Recovering low-rank matrices from few coefficients in any basis." *IEEE Transactions on Information Theory* 57.3 (2011), 1548–1566.

[Guo+18] Yi Guo, Kyri Baker, Emiliano Dall'Anese, Zechun Hu, and Tyler Holt Summers. "Data-based Distributionally Robust Stochastic Optimal Power Flow - Part I: Methodologies." *IEEE Transactions on Power Systems* 34.2 (2018), 1483–1492.

[GKJ20] Sarthak Gupta, Vassilis Kekatos, and Ming Jin. "Deep Learning for Reactive Power Control of Smart Inverters under Communication Constraints." *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE. 2020, 1–6.

[HC11] Wassim M Haddad and VijaySekhar Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton University Press, 2011.

[Han+19] Minghao Han, Yuan Tian, Lixian Zhang, Jun Wang, and Wei Pan. "H_∞ Model-free Reinforcement Learning with Robust Stability Guarantee." *CoRR* (2019).

[HSK06] Ken Harada, Jun Sakuma, and Shigenobu Kobayashi. "Local search for multiobjective function optimization: pareto descent method." *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. ACM. 2006, 659–666.

[HKM20] Fouad Hasan, Amin Kargarian, and Ali Mohammadi. "A Survey on Applications of Machine Learning for Optimal Power Flow." *2020 IEEE Texas Power and Energy Conference (TPEC)*. IEEE. 2020, 1–6.

[Has+17] Saqib Hasan, Amin Ghafouri, Abhishek Dubey, Gabor Karsai, and Xenofon Koutsoukos. "Heuristics-based approach for identifying critical N-k contingencies in power systems." *2017 Resilience Week (RWS)*. 2017, 191–197.

[HKM10] Tamir Hazan, Joseph Keshet, and David A McAllester. "Direct loss minimization for structured prediction." *Advances in Neural Information Processing Systems*. 2010, 1594–1602.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 770–778.

[HAG16] Jinhyok Heo, Peter J Adams, and H Oliver Gao. "Reduced-form modeling of public health impacts of inorganic $PM_{2.5}$ and precursor emissions." *Atmospheric Environment* 137 (2016), 80–89.

[HA17] Eric Hittinger and Inês ML Azevedo. "Estimating the Quantity of Wind and Solar Required To Displace Storage-Induced Emissions." *Environmental Science & Technology* 51.21 (2017), 12988–12997.

[HA15] Eric S Hittinger and Inês ML Azevedo. "Bulk Energy Storage Increases United States Electricity System Emissions." *Environmental Science & Technology* 49.5 (2015), 3203–3210.

[HL17] J Scott Holladay and Jacob LaRiviere. "The impact of cheap natural gas on marginal emissions from electricity generation and implications for energy policy." *Journal of Environmental Economics and Management* 85 (2017), 205–227.

[Hol+21] Jesse Holzer, Carleton Coffrin, Christopher DeMarco, Ray Duthu, Stephen Elbert, Scott Greene, Olga Kuchar, Bernard Lesieutre, Hanyue Li, Wai Keung Mak, Hans Mittelmann, Richard O'Neill, Thomas Overbye, Ahmad Tbaileh, Pascal Van Hentenryck, Arun Veeramany, and Jessica Wert. *Grid Optimization Competition Challenge 2 Problem Formulation*. Tech. rep. ARPA-E, 2021.

[HCZ17] Shaoyun Hong, Haozhong Cheng, and Pingliang Zeng. "N-K Constrained Composite Generation and Transmission Expansion Planning With Interval Load." *IEEE Access* 5 (2017), 2779–2789.

[Hor16] Nathaniel Charles Horner. "Powering the Information Age: Metrics, Social Cost Optimization Strategies, and Indirect Effects Related to Data Center Energy Use." PhD thesis. Pittsburgh, PA: Carnegie Mellon University, 2016.

[Hu+13] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. "Fast and accurate matrix completion via truncated nuclear norm regularization." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.9 (2013), 2117–2130.

[IEE10a] IEEE. *37 node distribution test feeder*. https://ewh.ieee.org/soc/pes/dsacom/testfeeders/. 2010.

[IEE10b] IEEE. *Resources: 123-bus Feeder*. http://sites.ieee.org/pes-testfeeders/files/2017/08/feeder123.zip. 2010.

[IEE20] IEEE. "IEEE Standard Conformance Test Procedures for Equipment Interconnecting Distributed Energy Resources with Electric Power Systems and Associated Interfaces." *IEEE Std 1547.1-2020* (2020), 1–282.

[IEE22] IEEE PES Task Force on Benchmarks for Validation of Emerging Power System Algorithms. *Power Grid Lib - Optimal Power Flow*. https://github.com/power-grid-lib/pglib-opf. 2022.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International Conference on Machine Learning*. PMLR. 2015, 448–456.

[IPC18] IPCC. *Global warming of 1.5°C. An IPCC special report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty* [V. Masson-Delmotte, P. Zhai, H. O. Pörtner, D. Roberts, J. Skea, P.R. Shukla, A. Pirani, Y. Chen, S. Connors, M. Gomis, E. Lonnoy, J. B. R. Matthews, W. Moufouma-Okia, C. Péan, R. Pidcock, N. Reay, M. Tignor, T. Waterfield, X. Zhou (eds.)] 2018.

[IPC21] IPCC. *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* [Masson-Delmotte, V., P. Zhai, A. Pirani, S. L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M. I. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T. K. Maycock, T. Waterfield, O. Yelekçi, R. Yu and B. Zhou (eds.)]. Cambridge University Press. 2021.

[IPC22a] IPCC. *Climate Change 2022: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* [H.-O. P ortner, D.C. Roberts, E.S. Poloczanska, K. Mintenbeck, M. Tignor, A. Alegría, M. Craig, S. Langsdorf, S. L oschke, V. M oller, A. Okem (eds.)]. Cambridge University Press. 2022.

[IPC22b] IPCC. *Climate Change 2022: Mitigation of Climate Change. Contribution of Working Group III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change* [P.R. Shukla, J. Skea, R. Slade, A. Al Khourdajie, R. van Diemen, D. McCollum, M. Pathak, S. Some, P. Vyas, R. Fradera, M. Belkacemi, A. Hasija, G. Lisboa, S. Luz, J. Malley, (eds.)]. Cambridge University Press. 2022.

[Jab20] Rabih A. Jabr. "Distributionally Robust CVaR Constraints for Power Flow Optimization." *IEEE Transactions on Power Systems* 35.5 (2020), 3764–3773.

[Jad+17] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. "Reinforcement learning with unsupervised auxiliary tasks." *International Conference on Learning Representations* (2017).

[Jal+19] Mana Jalali, Vassilis Kekatos, Nikolaos Gatsis, and Deepjyoti Deka. "Designing reactive power control rules for smart inverters using support vector machines." *IEEE Transactions on Smart Grid* 11.2 (2019), 1759–1770.

[JZ16] Huaiguang Jiang and Yingchen Zhang. "Short-term distribution system state forecast based on optimal synchrophasor sensor placement and extreme learning machine." *2016 IEEE Power and Energy Society General Meeting (PESGM)*. IEEE. 2016, 1–5.

[JL20] Ming Jin and Javad Lavaei. "Stability-certified reinforcement learning: A control-theoretic perspective." *IEEE Access* 8 (2020), 229086–229100.

[JMP21] Wanxin Jin, Shaoshuai Mou, and George J Pappas. "Safe pontryagin differentiable programming." *Advances in Neural Information Processing Systems* 34 (2021), 16034–16050.

[Jin+20] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. "Pontryagin Differentiable Programming: An End-to-End Learning and Control Framework." *Advances in Neural Information Processing Systems* 33 (2020).

[JW16] Chengquan Ju and Peng Wang. "Optimal power flow with worst-case scenarios considering uncertainties of loads and renewables." *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. IEEE. 2016, 1–7.

[Kaa+22] Lynn H. Kaack, Priya L. Donti, Emma Strubell, George Kamiya, Felix Creutzig, and David Rolnick. "Aligning Artificial Intelligence with Climate Change Mitigation." *Nature Climate Change* (2022), 1–10.

[KML13] Daniel T Kaffine, Brannin J McBee, and Jozef Lieskovsky. "Emissions Savings from Wind Power Generation in Texas." *The Energy Journal* 34 (2013), 155–175.

[KT16] P. Kaplunovich and K. Turitsyn. "Fast and Reliable Screening of N-2 Contingencies." *IEEE Transactions on Power Systems* 31.6 (2016), 4243–4252.

[KA16] Jeremy F Keen and Jay Apt. "Are high penetrations of commercial cogeneration good for society?" *Environmental Research Letters* 11.12 (2016), 124014.

[KMO10] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. "Matrix Completion from a Few Entries." *IEEE Transactions on Information Theory* 56.6 (2010), 2980–2998.

[KG02] Hassan K Khalil and Jessy W Grizzle. *Nonlinear Systems*. Vol. 3. Prentice Hall Upper Saddle River, NJ, 2002.

[KB15] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *International Conference on Learning Representations* (2015).

[KS04] Daniel S Kirschen and Goran Strbac. "Fundamentals of Power System Economics." John Wiley & Sons, 2004. Chap. 6.

[KZ15] Cecilia Klauber and Hao Zhu. "Distribution system state estimation using semidefinite programming." *2015 North American Power Symposium (NAPS)*. IEEE. 2015, 1–6.

[Kol13] J. Zico Kolter. *Problem Set 3, Computational Methods for the Smart Grid*. https://www.cs.cmu.edu/~zkolter/course/15-884/assignments.html. Oct. 2013.

[KDJ20] J. Zico Kolter, David Duvenaud, and Matthew Johnson. *Tutorial: Deep Implicit Layers - Neural ODEs, Deep Equilibirum Models, and Beyond*. http://implicit-layers-tutorial.org/. Dec. 2020.

[KM18] J. Zico Kolter and Aleksander Madry. *Adversarial Robustness - Theory and Practice*. https://adversarial-ml-tutorial.org/. Dec. 2018.

[KBM96] Mayuresh V Kothare, Venkataramanan Balakrishnan, and Manfred Morari. "Robust constrained model predictive control using linear matrix inequalities." *Automatica* 32.10 (1996), 1361–1379.

[KL13] Slawomir Koziel and Leifur Leifsson. *Surrogate-based modeling and optimization*. Springer, 2013.

[KP12] Steven G Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.

[KRP13] Roman Kuiava, Rodrigo A Ramos, and Hemanshu R Pota. "A New Method to Design Robust Power Oscillation Dampers for Distributed Synchronous Generation Systems." *Journal of Dynamic Systems, Measurement, and Control* 135.3 (2013).

[LBR16] Quang Linh Lam, Antoneta Iuliana Bratcu, and Delphine Riu. "Frequency Robust Control in Stand-alone Microgrids with PV Sources: Design and Sensitivity Analysis." 2016.

[LeC+05] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. "Off-road obstacle avoidance through end-to-end learning." *Conference on Neural Information Processing Systems*. 2005, 739–746.

[Lev+16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-end training of deep visuomotor policies." *Journal of Machine Learning Research* 17.39 (2016), 1–40.

[Li+18] Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. "NETT: Solving Inverse Problems with Deep Neural Networks." *arXiv preprint arXiv:1803.00092* (2018).

[Li+17] Mo Li, Timothy M Smith, Yi Yang, and Elizabeth J Wilson. "Marginal Emission Factors Considering Renewables: A Case Study of the US Midcontinent Independent System Operator (MISO) System." *Environmental Science & Technology* 51.19 (2017), 11215–11223.

[Lia82] TE Dy Liacco. "The role of state estimation in power system operation." *IFAC Proceedings Volumes* 15.4 (1982), 1531–1533.

[Lia+19] Mang Liao, Di Shi, Zhe Yu, Zhehan Yi, Zhiwei Wang, and Yingmeng Xiang. "An alternating direction method of multipliers based approach for PMU data recovery." *IEEE Transactions on Smart Grid* 10.4 (2019), 4554–4565.

[LSW06] Jeff Linderoth, Alexander Shapiro, and Stephen Wright. "The empirical behavior of sampling methods for stochastic programming." *Annals of Operations Research* 142.1 (2006), 215–241.

[LFK18] Chun Kai Ling, Fei Fang, and J Zico Kolter. "What Game Are We Playing? End-to-end Learning in Normal and Extensive Form Games." *International Joint Conference on Artificial Intelligence*. 2018.

[LLW13] Derong Liu, Hongliang Li, and Ding Wang. "Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm." *Neurocomputing* 110 (2013), 92–100.

[Liu+18] Yuxiao Liu, Ning Zhang, Yi Wang, Jingwei Yang, and Chongqing Kang. "Data-driven power flow linearization: A regression approach." *IEEE Transactions on Smart Grid* 10.3 (2018), 2569–2580.

[Lue+16] Roger Lueken, Kelly Klima, W Michael Griffin, and Jay Apt. "The climate and health effects of a USA switch from coal to gas electricity generation." *Energy* 109 (2016), 1160–1166.

[LWH14] Biao Luo, Huai-Ning Wu, and Tingwen Huang. "Off-Policy Reinforcement Learning for H_∞ Control Design." *IEEE Transactions on Cybernetics* 45.1 (2014), 65–76.

[Mad+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks." *International Conference on Learning Representations*. 2018.

[Man+12] Efthymios Manitsas, Ravindra Singh, Bikash C Pal, and Goran Strbac. "Distribution system state estimation using an artificial neural network approach for pseudo measurement modeling." *IEEE Transactions on power systems* 27.4 (2012), 1888–1896.

[MAT20] MATPOWER. *case33bw*. https://matpower.org/docs/ref/matpower6.0/case33bw.html. 2020.

[McA18] James McAnany. *2017 Demand Response Operations Markets Activity Report: April 2018*. Tech. rep. 2018.

[MJU17] Michael T McCann, Kyong Hwan Jin, and Michael Unser. "A review of convolutional neural networks for inverse problems in imaging." *IEEE Signal Processing Magazine* 34.6 (Nov. 2017), 85–95.

[MW09] Shaun D McRae and Frank A Wolak. "How do firms exercise unilateral market power? Evidence from a bid-based wholesale electricity market." *EUI RSCAS* (2009).

[Meh+21] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. "A survey on bias and fairness in machine learning." *ACM Computing Surveys (CSUR)* 54.6 (2021), 1–35.

[MMD20] Ilyes Mezghani, Sidhant Misra, and Deepjyoti Deka. "Stochastic AC optimal power flow: A data-driven approach." *Electric Power Systems Research* 189 (2020), 106567.

[Mil+18] Federico Milano, Florian Dörfler, Gabriela Hug, David J Hill, and Gregor Verbič. "Foundations and challenges of low-inertia systems." *2018 Power Systems Computation Conference (PSCC)*. IEEE. 2018, 1–25.

[MRN22] Sidhant Misra, Line Roald, and Yeesian Ng. "Learning for constrained optimization: Identifying optimal active constraint sets." *INFORMS Journal on Computing* 34.1 (2022), 463–480.

[Mni+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015), 529–533.

[MGL17] Erfan Mohagheghi, Aouss Gabash, and Pu Li. "A framework for real-time aunder wind energy penetration." *Energies* 10.4 (2017), 535.

[MR18] Daniel K Molzahn and Line A Roald. "Towards an AC optimal power flow algorithm with robust feasibility guarantees." *2018 Power Systems Computation Conference (PSCC)*. IEEE. 2018, 1–7.

[Mon17] Monitoring Analytics, LLC. *State of the Market Report for PJM: Volume 2: Detailed Analysis*. Tech. rep. 2017.

[Mor21] Catherine Morehouse. "ERCOT releases plan to boost reliability after blackouts, as report outlines gas, electric failures." *Utility Dive* (July 14, 2021). URL: https://www.utilitydive.com/news/ercot-releases-plan-to-boost-reliability-after-blackouts-as-report-outline/603263/ (visited on 10/25/2021).

[MSA14] Alexandre Moreira, Alexandre Street, and José M Arroyo. "An adjustable robust optimization approach for contingency-constrained transmission expansion planning." *IEEE Transactions on Power Systems* 30.4 (2014), 2013–2022.

[MD05] Jun Morimoto and Kenji Doya. "Robust Reinforcement Learning." *Neural Computation* 17.2 (2005), 335–359.

[Mos+20] T Moss, M Bazilian, M Blimpo, L Culver, J Kincer, M Mahadavan, V Modi, B Muhwezi, R Mutiso, V Sivaram, et al. "The Modern Energy Minimum: The Case for a New Global Electricity Consumption Threshold." *Energy for Growth Hub* (2020).

[Mos+16] Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. "Multi-Objective Deep Reinforcement Learning." *arXiv preprint arXiv:1610.02707* (2016).

[Müh+19] Tillmann Mühlpfordt, Line Roald, Veit Hagenmeyer, Timm Faulwasser, and Sidhant Misra. "Chance-constrained AC optimal power flow: A polynomial chaos approach." *IEEE Transactions on Power Systems* 34.6 (2019), 4806–4816.

[Mul14] Nicholas Z Muller. *Toward the Measurement of Net Economic Welfare: Air Pollution Damage in the US National Accounts–2002, 2005, 2008*. Ed. by Dale W. Jorgenson, J. Steven Landefeld, and Paul Schreyer. Chicago, IL: University of Chicago Press, 2014, 429–459.

[Mur22] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[Mur19] Sinnott J Murphy. "Correlated Generator Failures and Power System Reliability." PhD thesis. Carnegie Mellon University, 2019.

[Nat17] National Academies of Sciences, Engineering, and Medicine. *Enhancing the resilience of the nation's electricity system*. National Academies Press, 2017.

[Nes13] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Vol. 87. Springer Science & Business Media, 2013.

[NR+00] Andrew Y Ng, Stuart J Russell, et al. "Algorithms for inverse reinforcement learning." *International Conference on Machine Learning*. 2000, 663–670.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[PD11] Peter Palensky and Dietmar Dietrich. "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads." *IEEE Transactions on Industrial Informatics* 7.3 (2011), 381–388.

[Pan+20] Xiang Pan, Minghua Chen, Tianyu Zhao, and Steven H Low. "DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems." *arXiv preprint arXiv:2007.01002* (2020).

[PPW19] Ioannis Panageas, Georgios Piliouras, and Xiao Wang. "First-order methods almost always avoid saddle points: The case of vanishing step-sizes." *Conference on Neural Information Processing Systems* 32 (2019).

[PAP20] Amritanshu Pandey, Aayushya Agarwal, and Larry Pileggi. "Incremental Model Building Homotopy Approach for Solving Exact AC-Constrained Optimal Power Flow." *arXiv preprint arXiv:2011.00587* (2020).

[Pan+18] Amritanshu Pandey, Marko Jereminov, Martin R Wagner, David M Bromberg, Gabriela Hug, and Larry Pileggi. "Robust power flow and three-phase power flow analyses." *IEEE Transactions on Power Systems* 34.1 (2018), 616–626.

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "PyTorch: An imperative style, high-performance deep learning library." *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[PSM04] Jorge Pereira, JT Saraiva, and V Miranda. "An integrated load allocation/state estimation approach for distribution networks." *Probabilistic Methods Applied to Power Systems, 2004 International Conference on*. IEEE. 2004, 180–185.

[PAW14] Kasun S Perera, Zeyar Aung, and Wei Lee Woon. "Machine learning techniques for supporting renewable energy generation and integration: a survey." *International Workshop on Data Analytics for Renewable Energy Integration*. Springer. 2014, 81–96.

[Per+16] Michael Pertl, Kai Heussen, Oliver Gehrke, and Michel Rezkalla. "Voltage estimation in active distribution grids using neural networks." *2016 IEEE Power and Energy Society General Meeting (PESGM)*. IEEE. 2016, 1–5.

[PDMT18] Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. "Optlayer-practical constrained optimization for deep reinforcement learning in the real world." *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, 6236–6243.

[Pil98] Lawrence Pillage. *Electronic Circuit & System Simulation Methods (SRE)*. McGraw-Hill, Inc., 1998.

[Pin+17] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. "Robust Adversarial Reinforcement Learning." *Proceedings of the 34th International Conference on Machine Learning*. JMLR. org. 2017, 2817–2826.

[PJM17a] PJM. *2012-2016 CO2, SO2, and NOx Emission Rates*. https://www.pjm.com/~/media/library/reports-notices/special-reports/20170317-2016-emissions-report.ashx. Accessed: 2018-01-01. 2017.

[PJM17b] PJM. *Demand Response Strategy*. https://www.pjm.com/~/media/library/reports-notices/demand-response/20170628-pjm-demand-response-strategy.ashx. Accessed: 2018-01-01. 2017.

[PJM17c] PJM. *Generation Attribute Tracking System*. https://gats.pjm-eis.com/GATS2/PublicReports/PJMSystemMix. Accessed: 2018-01-01. 2017.

[PJM17d] PJM. *Metered Load Data*. http://www.pjm.com/markets-and-operations/ops-analysis/historical-load-data.aspx. Accessed: 2018-01-01. 2017.

[PJM17e] PJM. *PJM's Evolving Resource Mix and System Reliability*. https://www.pjm.com/~/media/library/reports-notices/special-reports/20170330-pjms-evolving-resource-mix-and-system-reliability.ashx. Accessed: 2018-01-01. 2017.

[PJM18] PJM. *2017 PJM Annual Report*. https://www.pjm.com/-/media/about-pjm/newsroom/annual-reports/2017-annual-report.ashx. Accessed: 2019-03-31. 2018.

[PJM18] PJM. *Data Miner 2: Real-Time Hourly LMPs*. http://dataminer2.pjm.com/feed/rt_hrl_lmps/definition. 2018.

[PL17] Anggoro Primadianto and Chan-Nan Lu. "A review on distribution system state estimation." *IEEE Transactions on Power Systems* 32.5 (2017), 3875–3883.

[Pri+13] Samuel Privara, Jiří Cigler, Zdeněk Váňa, Frauke Oldewurtel, Carina Sagerschnig, and Eva Žáčeková. "Building modeling as a crucial part for building predictive control." *Energy and Buildings* 56 (2013), 8–22.

[RS20] Maithra Raghu and Eric Schmidt. "A survey of deep learning for scientific discovery." *arXiv preprint arXiv:2003.11755* (2020).

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples." *International Conference on Learning Representations* (2018).

[RF+21] Francisco Ralston Fonseca, Michael Craig, Paulina Jaramillo, Mario Bergés, Edson Severnini, Aviva Loew, Haibo Zhai, Yifan Cheng, Bart Nijssen, Nathalie Voisin, et al. "Climate-Induced Tradeoffs in Planning and Operating Costs of a Regional Electricity System." *Environmental Science & Technology* 55.16 (2021), 11204–11215.

[Ram+12] Sarvapali D Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nicholas R Jennings. "Putting the'smarts' into the smart grid: a grand challenge for artificial intelligence." *Communications of the ACM* 55.4 (2012), 86–97.

[RA17] Line Roald and Göran Andersson. "Chance-constrained AC optimal power flow: Reformulations and efficient algorithms." *IEEE Transactions on Power Systems* 33.3 (2017), 2906–2918.

[RW91] R Tyrrell Rockafellar and Roger J-B Wets. "Scenarios and policy aggregation in optimization under uncertainty." *Mathematics of operations research* 16.1 (1991), 119–147.

[Rol+22] David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, Alexandra Sasha Luccioni, Tegan Maharaj, Evan D. Sherwin, S. Karthik Mukkavilli, Konrad P. Kording, Carla P. Gomes, Andrew Y. Ng, Demis Hassabis, John C. Platt, Felix Creutzig, Jennifer Chayes, and Yoshua Bengio. "Tackling Climate Change with Machine Learning." *ACM Computing Surveys* 55.2 (Feb. 2022, preprint 2019).

[Rus10] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[Rut+22] Daan Rutten, Nico Christianson, Debankur Mukherjee, and Adam Wierman. "Online Optimization with Untrusted Predictions." *arXiv preprint arXiv:2202.03519* (2022).

[RJK16] Nicole A Ryan, Jeremiah X Johnson, and Gregory A Keoleian. "Comparative Assessment of Models and Methods To Calculate Grid Electricity Emissions." *Environmental Science & Technology* 50.17 (2016), 8937–8953.

[Rya+18] Nicole A Ryan, Jeremiah X Johnson, Gregory A Keoleian, and Geoffrey M Lewis. "Decision Support Algorithm for Evaluating Carbon Dioxide Emissions from Electricity Generation in the United States." *Journal of Industrial Ecology* 22.6 (2018), 1318–1330.

[Sch+17a] Greg Schivley, Adam Goldstein, Constantine Samaras, Ines L Azevedo, Haibo Zhai, and H Scott Matthews. *Power Sector Carbon Index: Data, Sources, and Methods:* Carnegie Mellon University. 2017.

[Sch+15] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-dimensional continuous control using generalized advantage estimation." *arXiv preprint arXiv:1506.02438* (2015).

[Sch+17b] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal Policy Optimization Algorithms." *arXiv preprint arXiv:1707.06347* (2017).

[Sch+20] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. "Green AI." *Communications of the ACM* 63.12 (2020), 54–63.

[Sha+20] Sanket Shah, Sinha Arunesh, Varakantham Pradeep, Perrault Andrew, and Tambe Milind. "Solving online threat screening games using constrained action space reinforcement learning." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 02. 2020, 2226–2235.

[SP07] Alexander Shapiro and Andy Philpott. "A Tutorial on Stochastic Programming." *Manuscript. Available at www2.isye.gatech.edu/ashapiro/publications.html* 17 (2007).

[SEAM12] Kyle Siler-Evans, Ines Lima Azevedo, and M Granger Morgan. "Marginal Emissions Factors for the U.S. Electricity System." *Environmental Science & Technology* 46.9 (2012), 4742–4748.

[SE+13] Kyle Siler-Evans, Inês Lima Azevedo, M Granger Morgan, and Jay Apt. "Regional variations in the health, environmental, and climate benefits of wind and solar generation." *Proceedings of the National Academy of Sciences* 110.29 (2013), 11768–11773.

[SWZ01] Carlos Silva, Bruce F Wollenberg, and Charles Z Zheng. "Application of mechanism design to electric power markets (republished)." *IEEE Transactions on Power Systems* 16.4 (2001), 862–869.

[Sil+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "liGame of Go with Deep Neural Networks and Tree Search." *Nature* 529.7587 (2016), 484–489.

[SPV09] R. Singh, B. C. Pal, and R. B. Vinter. "Measurement Placement in Distribution System State Estimation." *IEEE Transactions on Power Systems* 24.2 (2009), 668–675.

[Sin+20] Sumeet Singh, Spencer M Richards, Vikas Sindhwani, Jean-Jacques E Slotine, and Marco Pavone. "Learning Stabilizable Nonlinear Dynamics with Contraction-Based Regularization." *The International Journal of Robotics Research* (2020), 0278364920949931.

[Son+16] Yang Song, Alexander G Schwing, Richard S Zemel, and Raquel Urtasun. "Training deep neural networks via direct loss minimization." *Proceedings of The 33rd International Conference on Machine Learning*. 2016, 2169–2177.

[SHG+12] Siddharth Sridhar, Adam Hahn, Manimaran Govindarasu, et al. "Cyber-Physical System Security for the Electric Power Grid." *Proceedings of the IEEE* 100.1 (2012), 210–224.

[Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 15.1 (2014), 1929–1958.

[Ste+20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. "OSQP: an operator splitting solver for quadratic programs." *Mathematical Programming Computation* 12.4 (2020), 637–672.

[Sto19] Liam Stoker. "National Grid ESO probing power cuts following sudden generation collapse." *Current±* (Aug. 10, 2019). URL: https://www.current-news.co.uk/news/national-grid-eso-probing-power-cuts-following-sudden-generation-collapse (visited on 10/25/2021).

[SRE11] Veselin Stoyanov, Alexander Ropson, and Jason Eisner. "Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure." *International Conference on Artificial Intelligence and Statistics* 15 (2011), 725–733. ISSN: 15324435.

[Str+20] N. Stringer, A. Bruce, I. MacGill, N. Haghdadi, P. Kilby, J. Mills, T. Veijalainen, M. Armitage, and N. Wilmot. "Consumer-Led Transition: Australia's World-Leading Distributed Energy Resource Integration Efforts."

IEEE Power and Energy Magazine 18.6 (2020), 20–36. DOI: 10.1109/MPE.2020.3014720.

[SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for deep learning in NLP." *Annual Meeting of the Association for Computational Linguistics* (2019).

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[TD18] Majid Alkaee Taleghan and Thomas G. Dietterich. "Efficient Exploration for Constrained MDPs." *2018 AAAI Spring Symposia*. 2018.

[Tam+16] Aviv Tamar, Sergey Levine, Pieter Abbeel, YI WU, and Garrett Thomas. "Value iteration networks." *Advances in Neural Information Processing Systems*. 2016, 2146–2154.

[Tam+15] Mili-Ann M Tamayao, Jeremy J Michalek, Chris Hendrickson, and Inês M.L. Azevedo. "Regional Variability and Uncertainty of Electric Vehicle Life Cycle CO2 Emissions across the United States." *Environmental Science & Technology* 49.14 (2015), 8844–8855.

[Ted09] Russ Tedrake. "Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for MIT 6.832." *Working draft edition* 3 (2009).

[Thi+17] Maninder PS Thind, Elizabeth J Wilson, Inês L Azevedo, and Julian D Marshall. "Marginal Emissions Factors for Electricity Generation in the Midcontinent ISO." *Environmental Science & Technology* 51.24 (2017), 14445–14452.

[Tho+06] Ryan W Thomas, Daniel H Friend, Luiz A Dasilva, and Allen B Mackenzie. "Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives." *IEEE Communications Magazine* 44.12 (2006), 51–57.

[TH12] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." *COURSERA: Neural networks for machine learning* 4.2 (2012), 26–31.

[TCC11] C-C Tsao, JE Campbell, and Yihsu Chen. "When renewable portfolio standards meet cap-and-trade regulations in the electricity sector: Market interactions, profits implications, and policy redundancy." *Energy Policy* 39.7 (2011), 3966–3974.

[TSK18] Sebastian Tschiatschek, Aytunc Sahin, and Andreas Krause. "Differentiable submodular maximization." *International Joint Conference on Artificial Intelligence*. 2018, 2731–2738.

[TA16] Maria Lorena Tuballa and Michael Lochinvar Abundo. "A review of the development of Smart Grid technologies." *Renewable and Sustainable Energy Reviews* 59 (2016), 710–725.

[TBK16] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. "Safe Exploration in Finite Markov Decision Processes with Gaussian Processes." *Advances in Neural Information Processing Systems*. 2016.

[Uni12a] United States Census Bureau. *Households and Families*. https://www.census.gov/prod/cen2010/briefs/c2010br-14.pdf. Accessed: 2018-01-01. 2012.

[Uni12b] United States Energy Information Administration. *Emissions allowance prices for SO_2 and NO_x remained low in 2011*. https://www.eia.gov/todayinenergy/detail.php?id=4830. Accessed: 2018-01-01. 2012.

[Uni17a] United States Energy Information Administration. *U.S. Energy Mapping System*. https://www.eia.gov/state/maps.php. Accessed: 2018-01-01. 2017.

[Uni17b] United States Energy Information Administration. *What are the greenhouse gas and air pollutant emissions factors for fuels and electricity?* https://www.eia.gov/tools/faqs/faq.php?id=76&t=11. Accessed: 2018-01-01. 2017.

[Uni09] United States Environmental Protection Agency. *Plain English Guide to the Part 75 Rule*. Tech. rep. June 2009.

[Uni17c] United States Environmental Protection Agency. *AVoided Emissions and geneRation Tool (AVERT) User Manual, Version 1.6*. https://www.epa.gov/sites/production/files/2017-07/documents/avert_user_manual_07-31-17_508.pdf. Accessed: 2018-01-01. 2017.

[Uni17d] United States Environmental Protection Agency. *National Emissions Inventory (NEI)*. https://www.epa.gov/air-emissions-inventories/national-emissions-inventory-nei. Nov. 2017.

[Uni18] United States Environmental Protection Agency. *n Air Markets Program Data*. https://campd.epa.gov/. 2018.

[Uni16a] United States Interagency Working Group on Social Cost of Greenhouse Gases. *Technical Update of the Social Cost of Carbon for Regulatory Impact Analysis Under Executive Order 12866*. Tech. rep. 2016.

[Uni16b] United States Supreme Court. *Federal Energy Regulatory Commission v. Electric Power Supply Association et al.* Volume 577, no. 14-840. Jan. 2016.

[VMN14] Kristof Van Moffaert and Ann Nowé. "Multi-objective reinforcement learning using sets of pareto dominating policies." *Journal of Machine Learning Research* 15.1 (2014), 3483–3512.

[Vic19] David G. Victor. *How artificial intelligence will affect the future of energy and climate*. https://www.brookings.edu/research/how-artificial-intelligence-will-affect-the-future-of-energy-and-climate/. 2019.

[VM06] Alexandra Von Meier. *Electric Power Systems: A Conceptual Introduction*. Wiley Online Library, 2006.

[Vra+13] Maria Vrakopoulou, Kostas Margellos, John Lygeros, and Göran Andersson. "A Probabilistic Framework for Reserve Scheduling and $N-1$ Security Assessment of Systems With High Wind Power Penetration." *IEEE Transactions on Power Systems* 28.4 (2013), 3885–3896.

[Wac+18] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. "Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[WB06] Andreas Wächter and Lorenz T Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." *Mathematical programming* 106.1 (2006), 25–57.

[Wal+08] Rahul Walawalkar, Seth Blumsack, Jay Apt, and Stephen Fernands. "An economic welfare analysis of demand response in the PJM electricity market." *Energy Policy* 36.10 (2008), 3692–3702.

[WF03] Stein W Wallace and Stein-Erik Fleten. "Stochastic programming models in energy." *Handbooks in operations research and management science* 10 (2003), 637–677.

[WBB11] Kai Wang, Boris Babenko, and Serge Belongie. "End-to-end scene text recognition." *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, 1457–1464.

[Wan+19] Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. "SATNet: Bridging Deep Learning and Logical Reasoning using a Differentiable Satisfiability Solver." *International Conference on Machine Learning*. 2019.

[WWG13] Qianfan Wang, Jean-Paul Watson, and Yongpei Guan. "Two-stage robust optimization for N-k contingency-constrained unit commitment." *IEEE Transactions on Power Systems* 28.3 (2013), 2366–2375.

[Wan+12] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. "End-to-end text recognition with convolutional neural networks." *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, 3304–3308.

[Wat03] David Watts. "Security and vulnerability in electric power systems." *35th North American power symposium*. Vol. 2. 2003, 559–566.

[Wei+15] Allison Weis, Jeremy J Michalek, Paulina Jaramillo, and Roger Lueken. "Emissions and Cost Implications of Controlled Electric Vehicle Charging in the U.S. PJM Interconnection." *Environmental Science & Technology* 49.9 (2015), 5813–5819.

[WWD14] Marco A Wiering, Maikel Withagen, and Mădălina M Drugan. "Model-based multi-objective reinforcement learning." *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014 IEEE Symposium on*. IEEE. 2014, 1–6.

[WM08] Stephen Wilcox and William Marion. *Users manual for TMY3 data sets*. National Renewable Energy Laboratory Golden, CO, 2008.

[WDT18] Bryan Wilder, Bistra Dilkina, and Milind Tambe. "Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization." *AAAI Conference on Artificial Intelligence*. 2018.

[Wil+20] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. "Integrating physics-based modeling with machine learning: A survey." *arXiv preprint arXiv:2003.04919* (2020).

[Wol03] Frank A Wolak. "Measuring Unilateral Market Power in Wholesale Electricity Markets: The California Market, 1998-2000." *American Economic Review* 93.2 (2003), 425–430.

[WK18] Eric Wong and Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope." *International Conference on Machine Learning*. PMLR. 2018, 5286–5295.

[Won+18] Eric Wong, Frank R Schmidt, Jan Hendrik Metzen, and J Zico Kolter. "Scaling provable adversarial defenses." *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, 8410–8419.

[WWS14] Allen J. Wood, Bruce F. Wollenberg, and Gerald B. Sheblé. "Optimal Power Flow." *Power generation, operation, and control*. John Wiley & Sons, 2014. Chap. 8, 350–402.

[WL13] Huai-Ning Wu and Biao Luo. "Simultaneous policy update algorithms for learning the solution of linear continuous-time H_∞ state feedback control." *Information Sciences* 222 (2013), 472–485.

[WHJ13] Jianzhong Wu, Yan He, and Nick Jenkins. "A robust state estimator for medium voltage distribution networks." *IEEE Transactions on Power Systems* 28.2 (2013), 1008–1016.

[Xu+20] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. "Adversarial attacks and defenses in images, graphs and text: A review." *International Journal of Automation and Computing* 17.2 (2020), 151–178.

[Yan+21] Mingyu Yan, Mohammad Shahidehpour, Aleksi Paaso, Liuxi Zhang, Ahmed Alabdulwahab, and Abdullah Abusorrah. "A Convex Three-Stage SCOPF Approach to Power System Flexibility With Unified Power Flow Controllers." *IEEE Transactions on Power Systems* 36.3 (2021), 1947–1960. DOI: 10.1109/TPWRS.2020.3036653.

[Yan+12] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. "A survey on cyber security for smart grid communications." *IEEE Communications Surveys and tutorials* 14.4 (2012), 998–1010.

[YN20] Haoxiang Yang and Harsha Nagarajan. "Optimal power flow in distribution networks under stochastic N-1 disruptions." *Electric Power Systems Research* 189 (2020), 106689.

[Yan+20] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. "Projection-Based Constrained Policy Optimization." *International Conference on Learning Representations*. 2020.

[YZZ01] David D Yao, Shuzhong Zhang, and Xun Yu Zhou. "A primal-dual semi-definite programming approach to linear quadratic control." *IEEE Transactions on Automatic Control* 46.9 (2001), 1442–1447.

[Yua+16] Ye Yuan, Omid Ardakanian, Steven Low, and Claire Tomlin. "On the Inverse Power Flow Problem." *arXiv preprint arXiv:1610.06631* (2016).

[Yuk+16] Tugce Yuksel, Mili-Ann M Tamayao, Chris Hendrickson, Inês ML Azevedo, and Jeremy J Michalek. "Effect of regional grid mix, driving patterns and

climate on the comparative carbon footprint of gasoline and plug-in electric vehicles in the United States." *Environmental Research Letters* 11.4 (2016), 044007.

[ZB20] Ahmed S Zamzam and Kyri Baker. "Learning optimal solutions for extremely fast AC optimal power flow." *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE. 2020, 1–6.

[ZHB20] Kaiqing Zhang, Bin Hu, and Tamer Basar. "Policy Optimization for $\mathcal{H}_2$ Linear Control with $\mathcal{H}_\infty$ Robustness Guarantee: Implicit Regularization and Global Convergence." *Learning for Dynamics and Control*. PMLR. 2020, 179–190.

[Zha+15] Xiao Zhang, Gabriela Hug, Zico Kolter, and Iiro Harjunkoski. "Industrial demand response by steel plants with spinning reserve provision." *North American Power Symposium (NAPS)* (2015), 1–6.

[ZL18] Zhiang Zhang and Khee Poh Lam. "Practical Implementation and Evaluation of Deep Reinforcement Learning Control for a Radiant Heating System." *Proceedings of the 5th Conference on Systems for Built Environments*. BuildSys '18. ACM, 2018, 148–157. ISBN: 978-1-4503-5951-1.

[ZZQ19] Zidong Zhang, Dongxia Zhang, and Robert C Qiu. "Deep reinforcement learning for power system applications: An overview." *CSEE Journal of Power and Energy Systems* 6.1 (2019), 213–225.

[Zha+17] Lin Zhao, Wei Zhang, He Hao, and Karanjit Kalsi. "A geometric approach to aggregate flexibility modeling of thermostatically controlled loads." *IEEE Transactions on Power Systems* 32.6 (2017), 4721–4731.

[ZD98] Kemin Zhou and John Comstock Doyle. *Essentials of Robust Control*. Vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

[ZV06] William T Ziemba and Raymond G Vickson. *Stochastic optimization models in finance*. Vol. 1. World Scientific, 2006.

[ZMSG97] Ray D Zimmerman, Carlos E Murillo-Sánchez, and Deqiang Gan. "MATPOWER: A MATLAB power system simulation package." *Manual, Power Systems Engineering Research Center, Ithaca NY* 1 (1997).

[ZC15] Marco Zugno and Antonio J Conejo. "A robust optimization approach to energy and reserve dispatch in electricity markets." *European Journal of Operational Research* 247.2 (2015), 659–671.

Appendices

Appendix A

Assessing Emissions and Damage Factors in PJM

Here, we provide additional detail on the data, assumptions, and results presented in Chapter 3: "Assessing Emissions and Damage Factors in PJM."

A.1 Discussion of National Emissions Inventory data

As described in Section 3.2.1, the EPA's National Emissions Inventory (NEI) reports primary annual $PM_{2.5}$ emissions for a subset of PJM fossil generators, but only for the years 2008, 2011, and 2014. We use this NEI data to calculate generator-specific emissions rates for generators in NEI; there were 142, 145, and 43 PJM generators represented in NEI 2008, 2011, and 2014, respectively (out of 1017 generators associated with PJM in eGRID 2011). For the PJM generators *not* represented in NEI, we apply average by-fuel-type emissions rates calculated using all United States generators included in NEI (i.e. using the 789, 786, and 432 US generators included in NEI 2008, 2011, and 2014, respectively).

We use the rates calculated from the 2008, 2011, and 2014 NEIs for the years 2006-2010, 2011-2013, and 2014-2017, respectively. However, it is possible that plant $PM_{2.5}$ rates may change somewhat from year to year, depending on plant operation levels. To assess this effect, we compared $PM_{2.5}$ rates for plants in PJM and the United States as a whole between the three NEI years available. Our analysis shows that *average* plant $PM_{2.5}$ rates do not change statistically significantly over time, where this average is over all NEI plants in PJM or the United States (Tables A.1–A.2). Since we used average rates for most plants in our analysis, this result indicates that our analysis is likely not affected by changes in average rates over time. However, $PM_{2.5}$ rates for specific plants may change over time (Figure A.1). Therefore, the availability of more individual plant-level data would allow us to slightly increase the accuracy of our results.

Table A.1: Average $PM_{2.5}$ rates for plants in PJM that are represented in all of NEI 2008, 2011, and 2014 (39 plants). We see that changes over time in mean $PM_{2.5}$ rates are not statistically significant for this small sample of PJM plants.

Year	Average plant $PM_{2.5}$ rate (kg/MWh)
2008	0.21 ± 0.29
2011	0.1 ± 0.12
2014	0.09 ± 0.1

Table A.2: Average $PM_{2.5}$ rates for plants in the United States in each NEI year (over 789, 786, and 432 plants for NEI 2008, 2011, and 2014, respectively). Standard deviations are too large to plot on the graph, so are shown in the table. We see that changes over time in mean $PM_{2.5}$ rates are not statistically significant, though rates for individual plants may potentially change over time.

Year	Overall	Biomass	Coal	Gas	Oil
2008	0.15 ± 0.55	0.25 ± 0.36	0.19 ± 0.54	0.11 ± 0.57	0.15 ± 0.14
2011	0.13 ± 0.61	0.21 ± 0.36	0.15 ± 0.31	0.09 ± 0.53	0.81 ± 2.86
2014	0.13 ± 0.61		0.1 ± 0.28	0.07 ± 0.19	0.02 ± 0.03

A.2 Information on damage models

We employ data from two models to estimate the damages from electricity generation in PJM, namely AP2 [Mul14] and EASIUR [HAG16].

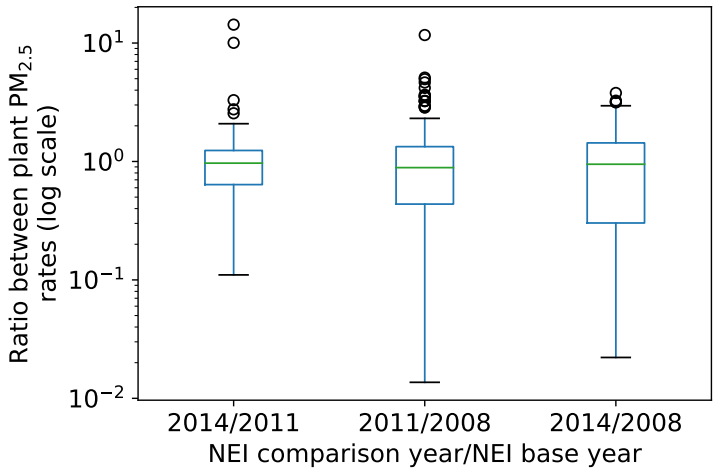
AP2 is an integrated assessment model that estimates social costs to human health, agriculture, forests, man-made structures, and recreation from SO_2, NO_x, $PM_{2.5}$, PM_{10}, NH_3, and VOCs (in 2000$/short ton). AP2 is based on a Gaussian dispersion model called the Climatological Regional Dispersion Model (CRDM).

EASIUR is a regression-based model that estimates human health damages from $PM_{2.5}$, SO_2, NO_x, and NH_3 emissions (in 2010$/metric ton). EASIUR is based on simulations from the chemical transport model CAMx and employs an "average plume" population exposure assumption.

Given marginal emissions produced at a specific location and height, both models predict the induced primary or secondary $PM_{2.5}$ concentrations. These concentrations are translated to monetary damages using a dose-response model and a value of statistical life (VSL). AP2 is available for emissions and population data years of 1999, 2002, 2005, 2008, and 2011, and employs a VSL of $6 million. EASIUR employs data from 2005 and a VSL of $8.8 million.

For model comparability, we use unmodified outputs from EASIUR and modified outputs from AP2. Specifically, AP2 outputs were scaled by 8.8/6 to employ the same VSL as EASIUR, converted to 2010$ using a multiplier of 1.27, and converted from short tons to metric tons.

Further, since AP2 outputs are available for multiple years, we employ linear interpolation between the reported damages for 2005, 2008, and 2011 to calculate damage intensities for the years 2006-2011. For years 2012 and later, we employ 2011 damage intensities



	2014/2011	2011/2008	2014/2008
Number of plants	42	142	39
Mean ratio	1.59	1.19	1.13
Std dev. of ratio	2.52	1.37	1.00
Min ratio	0.11	0.01	0.02
25th percentile ratio	0.64	0.44	0.30
50th percentile (median) ratio	0.97	0.89	0.95
75th percentile ratio	1.24	1.33	1.43
Max ratio	14.30	11.69	3.79

Figure A.1: Summary statistics for ratios of individual plant $PM_{2.5}$ rates between NEI years. Ratios necessarily include only plants represented in both NEI years considered.

(as more recent damage intensity estimates are not publicly available). We suspect that our damage estimates for 2011 and later are therefore underestimates of true damages, theoretically because the marginal damage of any unit of pollution increases as the air gets cleaner (due to the shape of the concentration-response curves used for human health damage assessments) and practically because damage intensities increased between 2008 and 2011 (so we would expect them to continue increasing). For EASIUR, we apply the reported 2005 outputs in all years.

We match each fossil generator in CEMS to its damage intensity in each model using generator locations from eGRID and generator stack height data from EIA Form 860. Stack heights were not available for many fossil generators, and were estimated where needed via a linear regression on generator capacity, age, and fuel type. AP2, which we use for our main analysis in Chapter 3, predicts one damage intensity per year. Since EASIUR predicts different damage intensities by season, for comparability, these were averaged at the generator level to get a generator-specific annual damage intensity for each of SO_2, NO_x, and $PM_{2.5}$. For context regarding the potential effects of this averaging, we report the seasonal variability of EASIUR aggregated damage intensities in Table A.3.

For each hour, we then multiply each generator's CEMS-reported SO_2, NO_x, and

Table A.3: Ratio of EASIUR seasonal damage intensity to annual average damage intensity. The PJM states used are those completely contained within PJM territory: Delaware, Maryland, New Jersey, Ohio, Pennsylvania, Virginia, and West Virginia.

	SO_2	NO_x	$PM_{2.5}$
Fall	0.72	1.29	0.90
Winter	0.76	1.13	1.01
Spring	1.25	1.14	1.01
Summer	1.26	0.44	1.07

(a) National

	SO_2	NO_x	$PM_{2.5}$
Fall	0.65	1.28	0.89
Winter	0.62	1.13	1.04
Spring	1.35	1.16	1.01
Summer	1.38	0.43	1.06

(b) PJM states

$PM_{2.5}$ emissions by the relevant generator-specific damage factors. These hourly damages are then aggregated to the PJM level for use in the calculations described in Sections 3.2.2–3.2.3.

A.3 Results under EASIUR

A.3.1 Annual and monthly emissions factors over time

Annual and monthly total damage factors under EASIUR (including CO_2, SO_2, NO_x, and $PM_{2.5}$) are shown in Figure A.2.

A.3.2 Intra-annual variability in emissions and damage factors

Monthly TOD total damage factors under EASIUR (including CO_2, SO_2, NO_x, and $PM_{2.5}$) are shown in Figure A.3.

A.3.3 Effects of a building-level lighting intervention

Figure A.4a shows annual damages avoided for the simple building-level lighting intervention described in Section 3.3.3 (a 100W reduction between 8pm-midnight every day in 2017) under the EASIUR damage model. We note the following results for damage factors under EASIUR (which were not previously reported). As a reminder, our baseline is marginal monthly TOD fossil+non-emitting damage factors (for PJM in 2017).

- Estimates for the health, environmental, and climate change damages avoided range from \$4-10 depending on the factor used (with a baseline estimate of \$7).
- The *average fossil+non-emitting* counterpart to the baseline underestimates damage reductions by 41%, and the *average fossil-only* counterpart to the baseline overestimates damage reductions by 1%.
- Outdated 2016 monthly TOD fossil+non-emitting MEFs overestimate total damages avoided by 30% compared to their 2017 counterpart (i.e. the baseline).

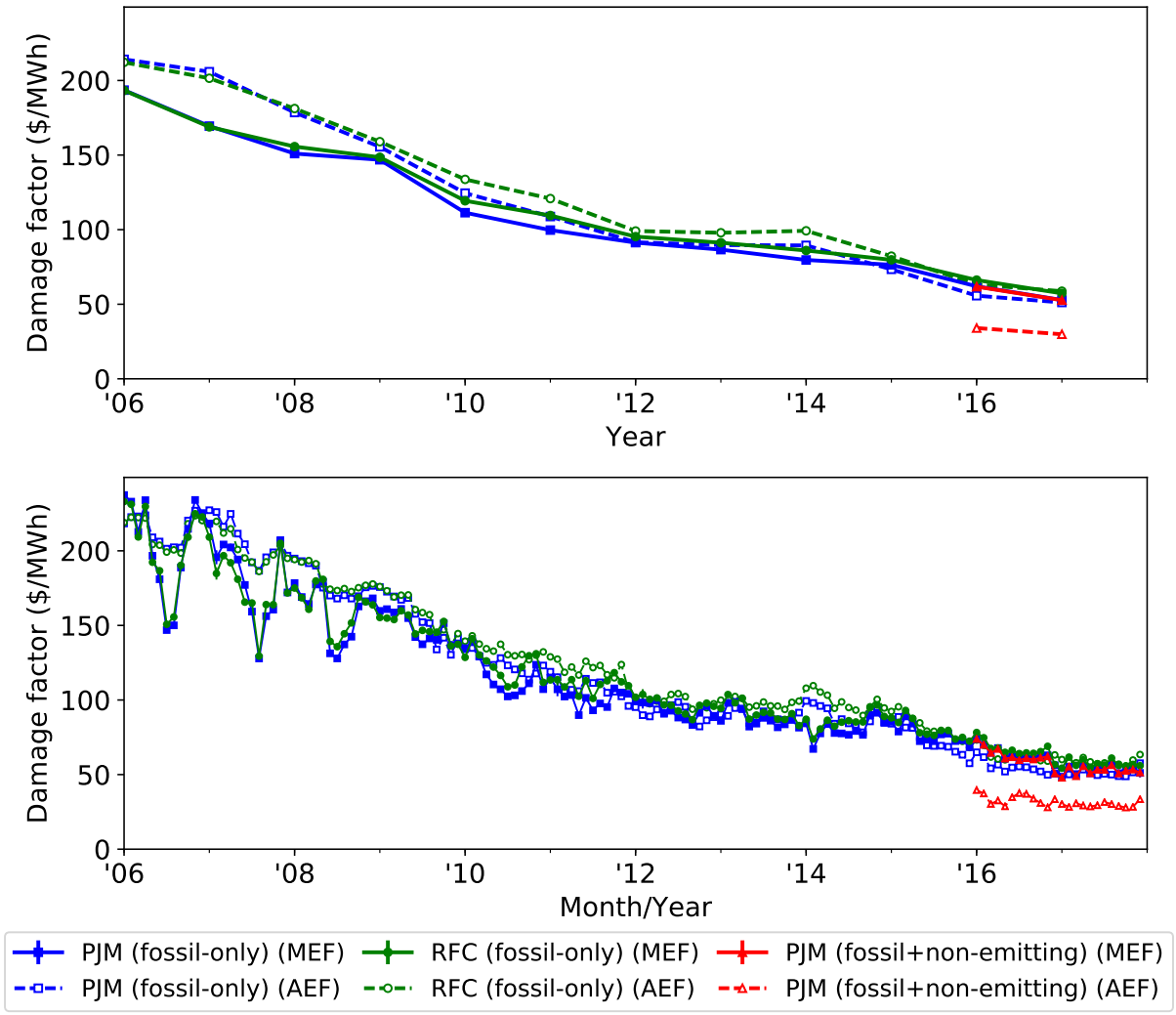


Figure A.2: Annual (top) and monthly (bottom) average and marginal factors over time for total damages under the EASIUR damage model (i.e., health damages from SO_2, NO_x, and $PM_{2.5}$, and climate change damages from CO_2 in 2010 dollars) in PJM and RFC. Error bars for marginal factors (narrow) represent regression standard errors and do not reflect the uncertainty in the underlying data. Caption entries are defined as in Figure 3.2.

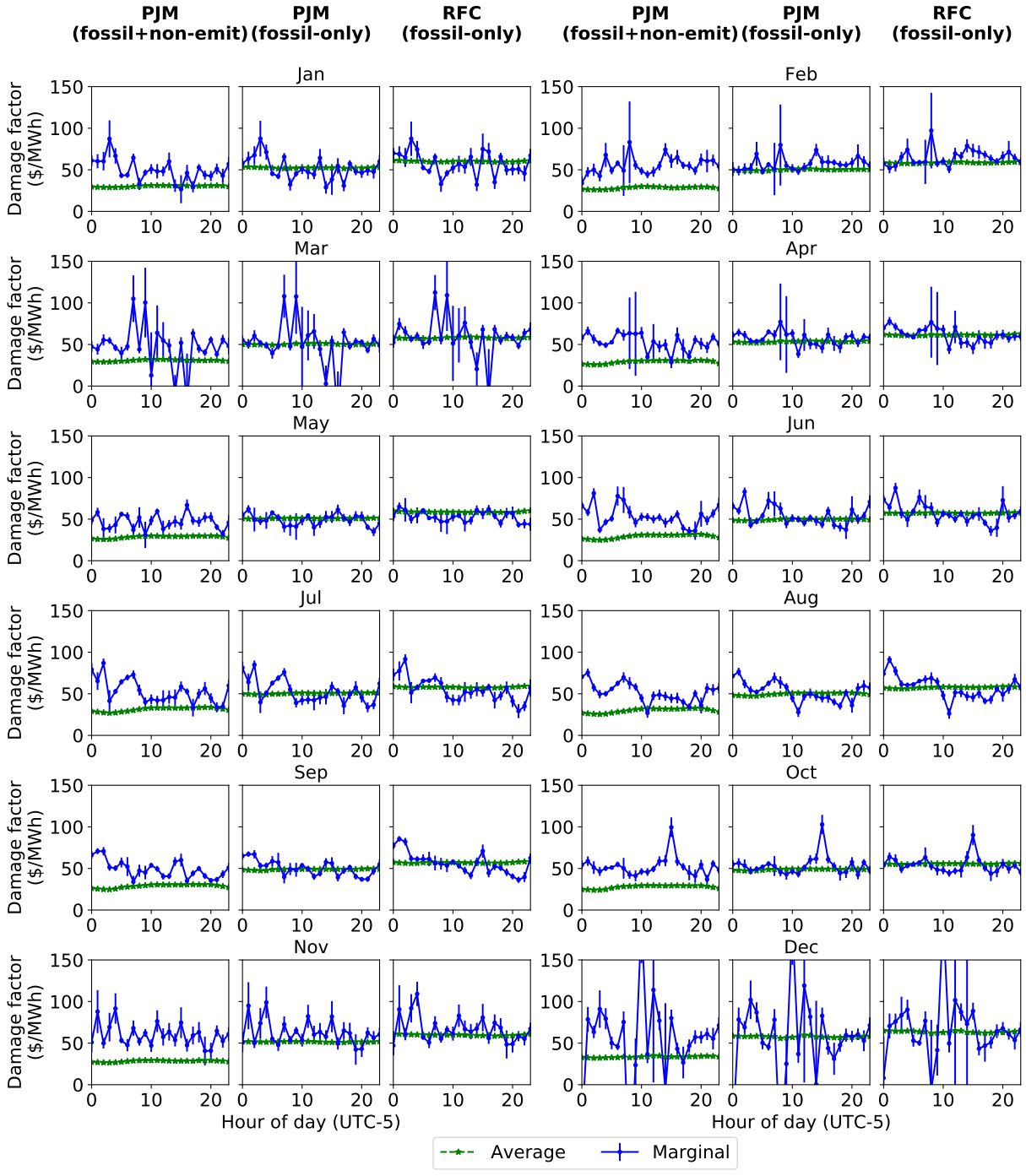
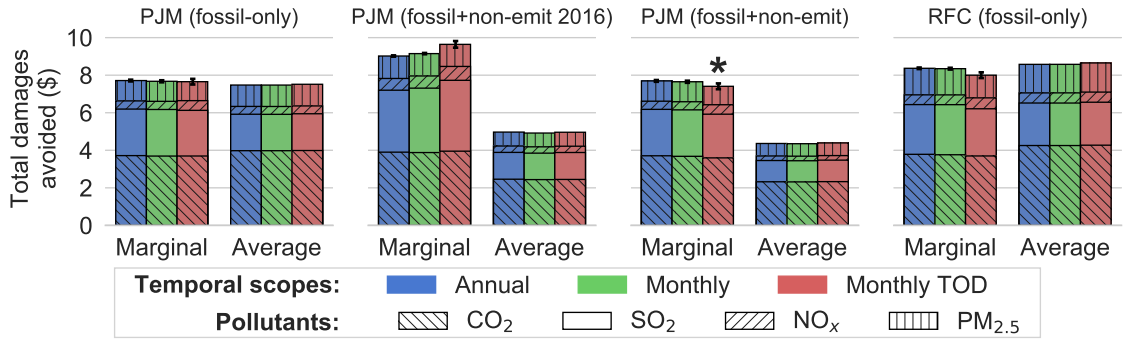
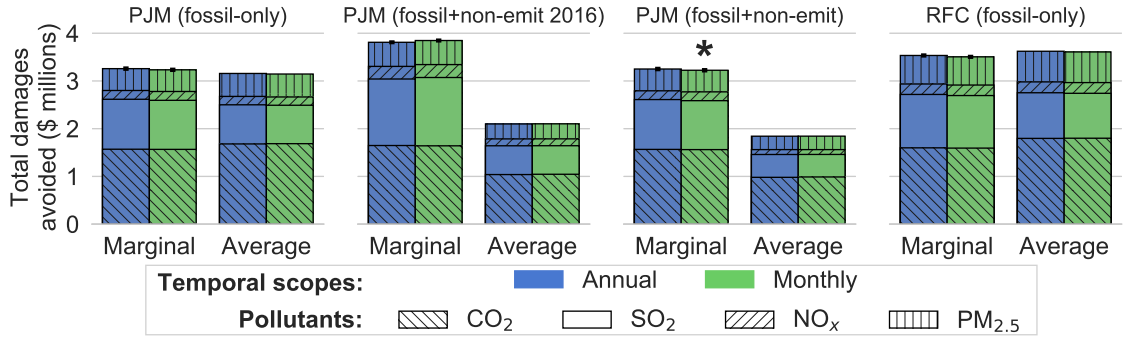


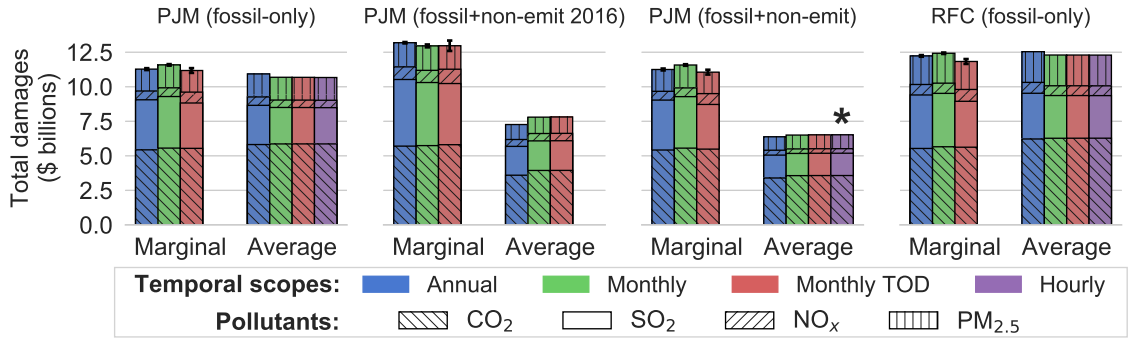
Figure A.3: Monthly time of day total damage factors under the EASIUR damage model (incorporating health and climate change damages for CO_2, SO_2, NO_x, and $PM_{2.5}$, in 2010 dollars), in all months of 2017. Error bars for marginal factors represent regression standard errors and do not reflect the uncertainty in the underlying data. PJM (fossil+non-emitting), PJM (fossil-only), and RFC (fossil-only) are as defined in Figure 3.2.



(a) Total annual damages avoided for a 2017 nighttime building level lighting intervention in PJM that induces a daily 100W reduction from 8pm to midnight.



(b) Total damages avoided for 2017 PJM historical demand response, assuming complete load shedding.



(c) Total damages from 2017 PJM summer metered load (June-August). As we do not estimate hourly-level marginal factors, and since hourly level 2016 factors should not be applied to 2017, we omit hourly-level estimates in these cases.

Figure A.4: Effects of interventions and loads evaluated as assessed with damage factors under the EASIUR damage model (incorporating health and climate change damages for CO_2, SO_2, NO_x, and $PM_{2.5}$, in 2010 dollars). Baseline factor effects are indicated with an asterisk. Error bars (narrow) represent propagated regression standard errors. PJM (fossil-only), PJM (fossil+non-emitting), PJM (fossil+non-emitting 2016), and RFC (fossil-only) are as defined in Figure 3.4.

- The distinctions between PJM and RFC factor assessments are within 8% (comparing both the fossil+non-emitting baseline and its fossil-only counterpart to the RFC, fossil-only counterpart to the baseline).
- Using the PJM fossil-only counterpart to the baseline yields an estimate 3% higher than with the baseline fossil+non-emitting marginal factor.
- The distinction between annual, monthly, and monthly TOD factors yields differences in assessments of less than 4%.

A.3.4 Effects of historical demand response

Assuming complete load shedding under the historical demand response profile shown in Figure 3.5, Figure A.4b shows the damages avoided by 2017 demand response in PJM (as described in Section 3.3.4) under the EASIUR damage model. We note the following results for damage factors under EASIUR (which were not previously reported). As a reminder, our baseline is marginal monthly fossil+non-emitting damage factors (for PJM in 2017).

- Estimates for the health, environmental, and climate change damages avoided range from \$1.8 to \$3.9 million depending on the factor used (with a baseline estimate of \$3.2 million).
- The *average fossil+non-emitting* counterparts to the baseline underestimate total damages avoided by 43%, and the *average fossil-only* counterparts to the baseline underestimate damage reductions by 2%.
- Outdated 2016 monthly fossil+non-emitting MEFs overestimate total damages avoided by 19% compared to their 2017 counterpart (i.e. the baseline).
- The RFC counterpart to the EASIUR baseline overestimates total damages avoided by 9% compared to the baseline.
- The distinctions between annual vs. monthly factors and fossil-only vs. fossil+non-emitting factors lead to differences in estimates of under 1%.

A.3.5 Effects of historical summer load

Figure A.4c shows the damages produced by summer load in PJM (June-August 2017, as described in Section 3.3.5) under the EASIUR damage model. We note the following results for damages under EASIUR (which were not previously reported). As a reminder, our baseline is *average* monthly TOD *fossil+non-emitting* damage factors (for PJM in 2017).

- Estimates for the health, environmental, and climate change damages of summer load range from \$6.4 to \$13 billion depending on the factor used (with a baseline estimate of \$6.5 billion).
- The fossil-only counterparts to the baseline overestimate damages avoided by 64%.
- The marginal monthly TOD counterparts to the baseline overestimate total damages by 70%.

Table A.4: Damages avoided ($ millions) for 2017 PJM historical demand response, assuming the extreme case where all reported reductions are actually load shifts from the hour with the highest damage factors to the hour with the lowest damage factors. This assessment was performed *separately* for each of total, CO_2-only, SO_2-only, NO_x-only, and $PM_{2.5}$-only factors under the AP2 and EASIUR damage models, using PJM fossil+non-emitting monthly TOD marginal factors for 2017.

	Total damage factor	CO_2-only	SO_2-only	NO_x-only	$PM_{2.5}$-only
AP2	7.09 ±1.14	0.59 ±0.06	6.67 ±1.14	0.53 ±0.08	0.32 ±0.04
EASIUR	4.4 ±0.58	0.59 ±0.06	3.9 ±0.61	0.34 ±0.04	0.32 ±0.03

A.4 Sensitivity analysis for historical demand response

The demand response results reported in Sections 3.3.4 and A.3.4 do not account for load shifting but instead assume complete load shedding. To understand the impact of this assumption, we assess the potential damage effects of an extreme load shifting scenario. That is, we analyze the extreme case in which all DR reductions PJM reports are actually load shifts from the hour with the highest damage factors to the hour with the lowest damage factors within each month (as estimated by using monthly TOD factors). We perform this assessment separately for each of total, CO_2-only, SO_2-only, NO_x-only, and $PM_{2.5}$-only factors under the AP2 and EASIUR damage models, using PJM fossil+non-emitting monthly TOD marginal factors for 2017. The results of this analysis are shown in Table A.4. We find that under total AP2 damage factors, this extreme-case load-shifting would lead to $7.09 million in damages avoided. (For context, our previously-reported baseline estimate for damages avoided under complete load shedding was $3.7 million under monthly factors.) By the same argument, however, shifting load *to* the hour with the highest damage factors *from* the hour with the lowest damage factors within each month would lead to $7.09 million in *damages incurred*. As such, load shifting can have a large impact on DR assessments, warranting the release of more granular data to enable accurate assessments.

A.5 Comparison to PJM-published emissions factors

PJM has published estimates for its annual and monthly marginal on-peak, marginal off-peak, and system average emissions rates for CO_2, SO_2, and NO_x from 2012-16 [PJM17a]. To calculate marginal emissions rates, PJM first estimates generator-specific annual emissions rates for all PJM generators using PJM generation data as well as emissions data from sources including CEMS and eGRID. For each five-minute interval, PJM then identifies which generators are marginal (which is market-sensitive information), and then calculates the marginal emissions rate for that five-minute interval as the mean emissions rate for all marginal generators. These five-minute marginal emissions rates are then averaged to the monthly or annual levels.

We compare PJM's marginal and average factor estimates to those we obtained via the methods described in Chapter 3. Specifically, we compare PJM's marginal on- and off-peak estimates to our fossil-only MEFs, as we do not distinguish between on- and off-peak hours

Table A.5: Percent difference of PJM's published (on-peak and off-peak) annual marginal emissions factors from our annual marginal emissions factor estimates for CO_2, SO_2, and NO_x in each year from 2012-16.

Year	***CO_2***		***SO_2***		***NO_x***	
	On-peak	**Off-peak**	**On-peak**	**Off-peak**	**On-peak**	**Off-peak**
2012	-1	-9	-8	-14	-2	-24
2013	11	14	27	9	29	10
2014	11	15	71	103	30	29
2015	9	2	22	26	41	14
2016	9	0	-4	-20	38	7

when calculating our annual and monthly factors. We also compare PJM's system average emissions rates to our fossil+non-emitting AEFs, as PJM's average rates include nuclear and renewable sources [PJM17c]. A plot of these factors at the annual and monthly levels is shown in Figures A.5–A.7, with PJM's factors (originally reported in lb/MWh) converted to units of kg/MWh.

At the annual level, both PJM's marginal on- and off-peak estimates overall exceed our MEF estimates from 2012-16. Specifically, PJM's marginal off-peak estimates for CO_2, SO_2, and NO_x exceed our MEF estimates by a mean of 4%, 21%, and 7%, respectively, in these years. PJM's on-peak estimates for CO_2, SO_2, and NO_x exceed our MEF estimates by a mean of 8%, 21%, and 27%, respectively. However, the relationship between our factors and PJM's published factors is not consistent across all years (see Table A.5). Qualitatively, PJM's published monthly-level marginal factors demonstrate significantly more intra-annual variability than ours, especially for SO_2 and NO_x.

We compare average factor estimates for 2016, as this is the only full year from 2012-16 for which PJM data was publicly available for us to calculate fossil+non-emitting AEFs. In 2016, PJM's average factors for CO_2, SO_2, and NO_x exceeded our AEF estimates by 6%, 70%, and 20%, respectively. Qualitatively, PJM's CO_2 monthly average factors demonstrate similar intra-annual variability to ours, whereas PJM's SO_2 and NO_x monthly factors vary significantly more than ours.

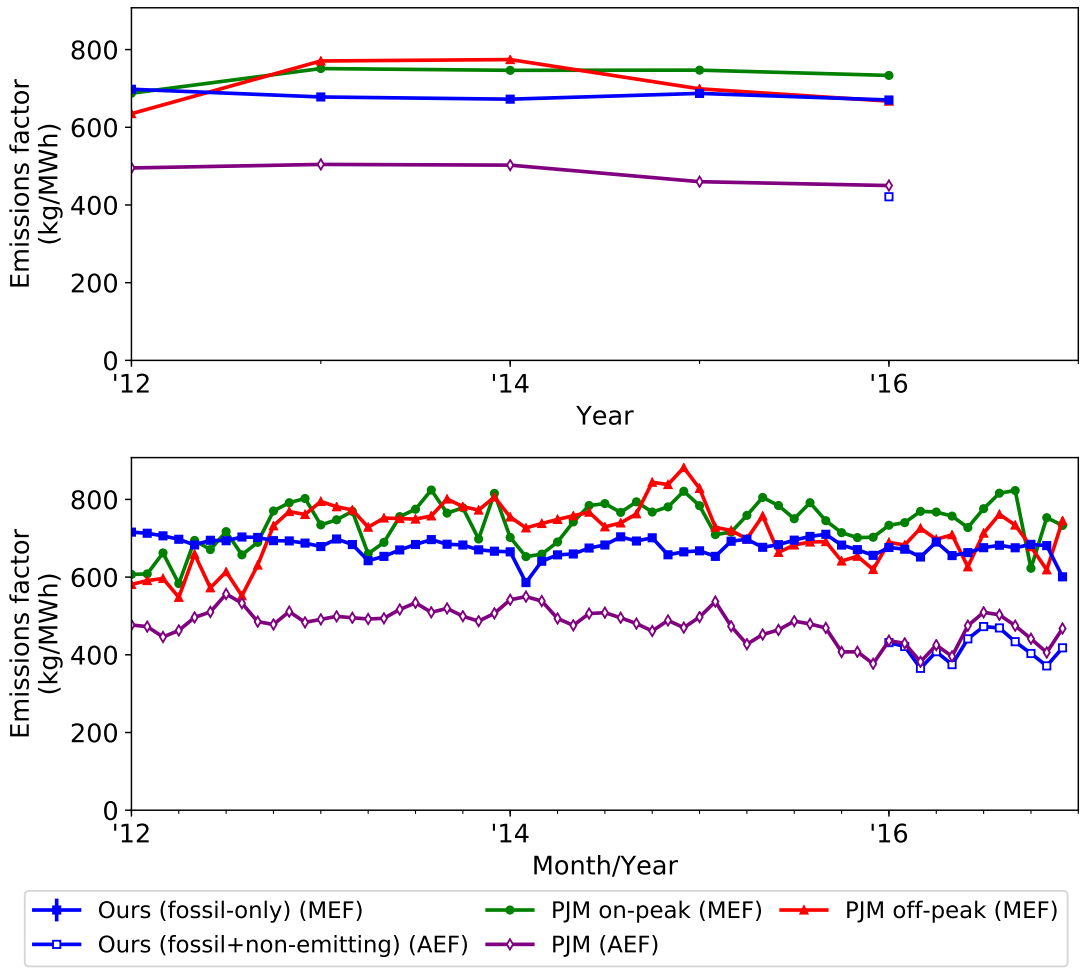


Figure A.5: Annual and monthly average and marginal factors over time for CO_2 as estimated using our method and as published by PJM. Error bars for our marginal factors (narrow) represent regression standard errors; no uncertainty was reported for PJM's published factors. Ours (fossil-only) (MEF) = our marginal emissions factor estimates using only fossil fuel generation in PJM; Ours (fossil+non-emitting) (AEF) = our average emissions factor estimates using fossil fuel and non-emitting generation in PJM; PJM on-peak (MEF) = PJM's published on-peak marginal emissions factors; PJM off-peak (MEF) = PJM's published off-peak marginal emissions factors; PJM (AEF) = PJM's published system average emissions factors.

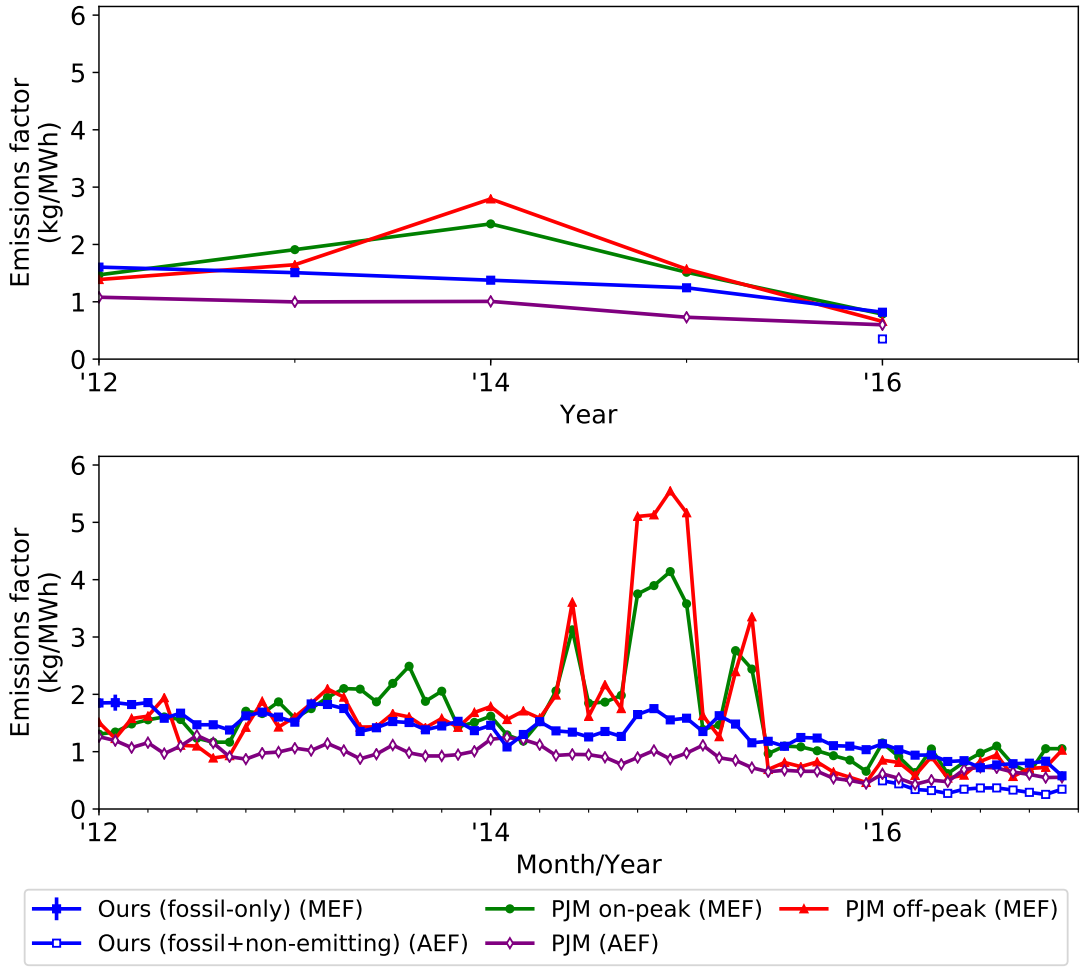


Figure A.6: Annual and monthly average and marginal factors over time for SO_2 as estimated using our method and as published by PJM. Error bars for our marginal factors (narrow) represent regression standard errors; no uncertainty was reported for PJM's published factors. Caption entries are as defined in Figure A.5.

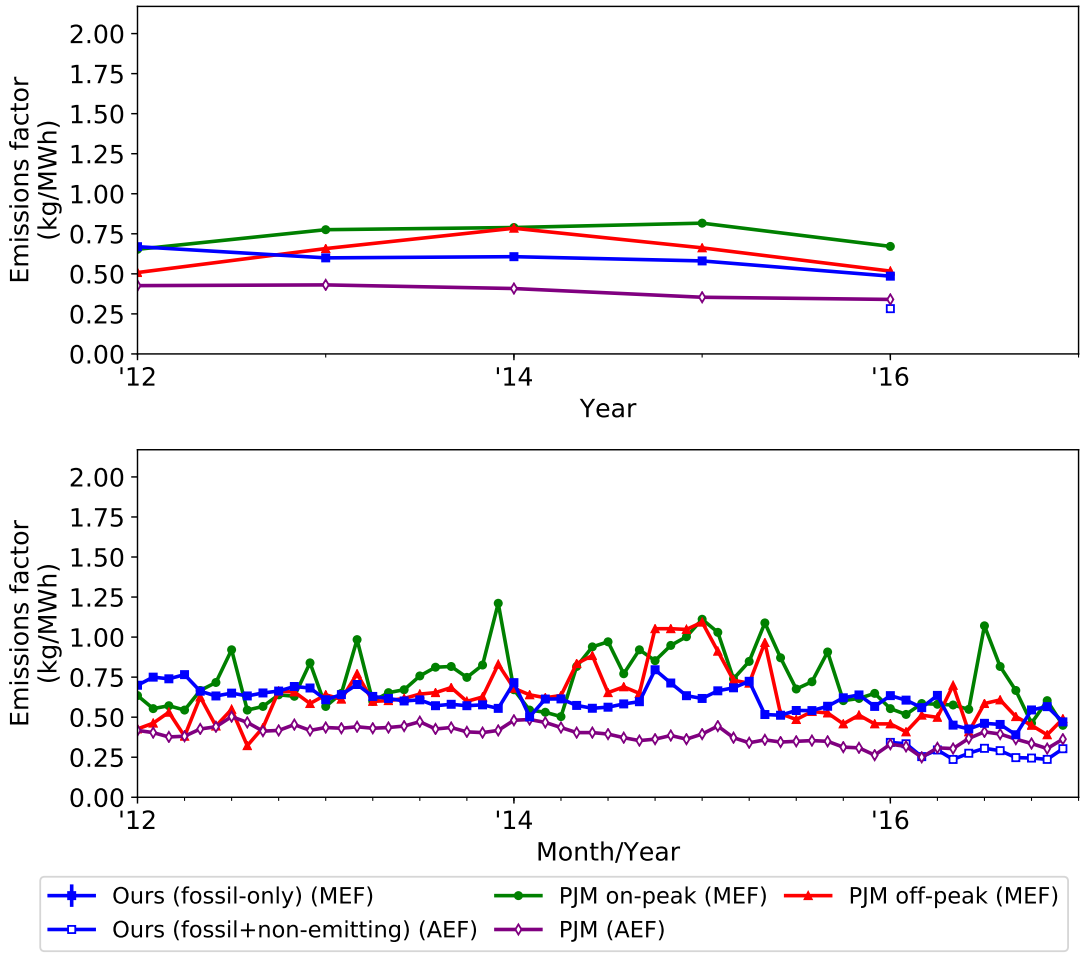


Figure A.7: Annual and monthly average and marginal factors over time for NO_x as estimated using our method and as published by PJM. Error bars for our marginal factors (narrow) represent regression standard errors; no uncertainty was reported for PJM's published factors. Caption entries are as defined in Figure A.5.

Appendix B

Approximating Optimization Problems with Hard Constraints

B.1 Details of DC3 for ACOPF

This section provides additional details on we apply DC3 (Chapter 6) to the problem of ACOPF.

B.1.1 Problem setting

We consider the problem of ACOPF defined in Section 6.4.3.

Let $\mathcal{B}$ denote the overall set of buses (i.e., nodes in the power network). In any instance of ACOPF there exists a set $\mathcal{D} \subseteq \mathcal{B}$ of *load (demand) buses* at which p_g and q_g are identically zero, as well as a set $\mathcal{R} \subseteq \mathcal{B}$ of *reference (slack) buses* at which p_g and q_g are potentially nonzero, and where the voltage angle $\angle v$ is known. Let $\mathcal{G} = \mathcal{B} \setminus (\mathcal{D} \cup \mathcal{R})$ be the remaining *generator buses* at which p_g and q_g are potentially nonzero, but where the voltage angle $\angle v$ is not known.

Then, we may rewrite Equation (6.7) as follows, where $v \equiv |v|e^{i\angle v}$ and $W \equiv W_r + W_i i$:

$$\begin{aligned}
\underset{p_g, q_g, |v|, \angle v \in \mathbb{R}^{|\mathcal{B}|}}{\text{minimize}} \quad & p_g^T \operatorname{diag}(c_q) p_g + c_a^T p_g && \text{(B.1a)}\\
\text{subject to} \quad & p_g^{\min} \leq p_g \leq p_g^{\max} && \text{(B.1b)}\\
& q_g^{\min} \leq q_g \leq q_g^{\max} && \text{(B.1c)}\\
& v^{\min} \leq |v| \leq v^{\max} && \text{(B.1d)}\\
& (\angle v)_{\mathcal{R}} = \phi_{\text{slack}} && \text{(B.1e)}\\
& (p_g)_{\mathcal{D}} = (q_g)_{\mathcal{D}} = 0 && \text{(B.1f)}\\
& p_g - p_d - \operatorname{diag}(v_r)(W_r v_r - W_i v_i) - \operatorname{diag}(v_i)(W_i v_r + W_r v_i) = 0 && \text{(B.1g)}\\
& q_g - q_d + \operatorname{diag}(v_r)(W_i v_r + W_r v_i) - \operatorname{diag}(v_i)(W_r v_r - W_i v_i) = 0 && \text{(B.1h)}\\
& \quad \text{where } v_r = |v|\cos(\angle v) \text{ and } v_i = |v|\sin(\angle v)
\end{aligned}$$

(While we write the problem in this form to minimize notation, in practice, some of the constraints in the above problem – e.g., (B.1b), (B.1c), and (B.1f) – can be condensed.)

B.1.2 Overall approach

As pointed out in [ZB20], given p_d, q_d, $(p_g)_{\mathcal{G}}$ and $|v|_{\mathcal{B}\setminus\mathcal{D}}$, the remaining variables $(p_g)_{\mathcal{R}}$, $(q_g)_{\mathcal{B}\setminus\mathcal{D}}$, $|v|_{\mathcal{D}}$, and $(\angle v)_{\mathcal{B}\setminus\mathcal{R}}$ can be recovered via the power flow equations (B.1g)–(B.1h).

As such, our implementation of DC3 may be outlined as follows.

- **Input:** $x = \begin{bmatrix} p_d^T & q_d^T \end{bmatrix}^T$. (The constant $\angle v_{\mathcal{R}}$ is fixed across problem instances.)
- **Neural network:** Output $\alpha \in [0,1]^{|\mathcal{G}|}$ and $\beta \in [0,1]^{|\mathcal{B}|-|\mathcal{D}|}$ (applying a sigmoid to the final layer of the network), and compute:

$$(p_g)_{\mathcal{G}} = \alpha(p_g^{\min})_{\mathcal{G}} + (1-\alpha)(p_g^{\max})_{\mathcal{G}},$$

$$|v|_{\mathcal{B}\setminus\mathcal{D}} = \beta v_{\mathcal{B}\setminus\mathcal{D}}^{\min} + (1-\beta)v_{\mathcal{B}\setminus\mathcal{D}}^{\max}.$$

  (In other words, output a partial set of variables, and enforce box constraints on this set of variables via sigmoids in the neural network.)
- **Completion procedure:** Given $z = \begin{bmatrix} (p_g)_{\mathcal{G}}^T & |v|_{\mathcal{B}\setminus\mathcal{D}}^T \end{bmatrix}^T$, solve Equations (B.1g)–(B.1h) for the remaining quantities $(p_g)_{\mathcal{R}}$, $(q_g)_{\mathcal{B}\setminus\mathcal{D}}$, $|v|_{\mathcal{D}}$, and $(\angle v)_{\mathcal{B}\setminus\mathcal{R}}$ as described below. Output all decision variables $y = \begin{bmatrix} (p_g)_{\mathcal{G}}^T & (q_g)_{\mathcal{G}}^T & |v|^T & (\angle v)^T \end{bmatrix}^T$.
- **Correction procedure:** Correct y using the gradient-based feasibility correction procedure described in Section 6.3.2.
- **Backpropagate:** Compute the loss and backpropagate as described below to update neural network parameters. Repeat until convergence.

We now describe the forward and backward passes through the completion procedure, where we have introduced several "tricks" that significantly reduce the computational cost, including some that are specific to the ACOPF setting.

For ease of notation throughout, we will let $J \in \mathbb{R}^{1+2|\mathcal{B}|}$ denote the Jacobian of the equality constraints (B.1e)–(B.1h) with respect to the complete vector y of decision variables. This matrix, which we pre-compute, will be useful throughout our computations.

B.1.3 Solving the completion

The neural network outputs $(p_g)_{\mathcal{G}}$ and $|v|_{\mathcal{B}\setminus\mathcal{D}}$ are the inputs to the completion procedure. Theoretically, we could use Newton's method to solve for all additional variables from the equality constraints. However, in practice we find that solving for all variables using Newton's method is sometimes unstable. Fortunately, we can divide the completion procedure into two substeps, where Step 1 invokes Newton's method to identify some of the variables, while Step 2 solves in closed form for the others. This greatly improves the stability of the completion procedure.

Step 1. Compute $|v|_{\mathcal{D}}$, $(\angle v)_{\mathcal{B}\setminus\mathcal{R}}$ via Newton's method using real power flow constraints (B.1g) at buses $\mathcal{B}\setminus\mathcal{R}$, and reactive power constraints (B.1h) at buses $\mathcal{D}$ (note that the number of equations matches the number of variables being identified). We initialize Newton's method by fixing variables determined by the neural network and those already determined in Step 1 of the completion process, and initializing $|v|_{\mathcal{D}}$, $(\angle v)_{\mathcal{B}\setminus\mathcal{R}}$ at generic initial values (these are provided in the ACOPF task setup and are typically used to initialize state-of-the-art solvers). Note that the remaining variables, those determined in Step 2 of the completion process, do not actually appear in the relevant equality constraints and therefore do not need to be set.

Let $J_{\text{Step 1}}$ denote the submatrix of J corresponding to the equality constraints $(\text{B.1g})_{\mathcal{B}\setminus\mathcal{R}}$ and $(\text{B.1h})_{\mathcal{D}}$ and the voltage variables $|v|_{\mathcal{D}}$ and $(\angle v)_{\mathcal{B}\setminus\mathcal{R}}$ that we are solving in this step. At the tth step of Newton's method, let $h_{\text{Step 1}}(t) \in \mathbb{R}^{|\mathcal{B}|-|\mathcal{R}|+|\mathcal{D}|}$ denote the vector of values on the left-hand side of the relevant equality constraints (which we wish to equal identically zero), evaluated at the current setting of all problem variables. Our Newton's method updates are then

$$\begin{bmatrix} |v|_{\mathcal{D}} \\ (\angle v)_{\mathcal{B}\setminus\mathcal{R}} \end{bmatrix}_{t+1} = \begin{bmatrix} |v|_{\mathcal{D}} \\ (\angle v)_{\mathcal{B}\setminus\mathcal{R}} \end{bmatrix}_{t} - J_{\text{Step 1}}^{-1} \, h_{\text{Step 1}}(t). \tag{B.2}$$

Step 2. Compute the remaining variables $(p_g)_{\mathcal{R}}$ and $(q_g)_{\mathcal{B}\setminus\mathcal{D}}$ via the remaining equality constraints – that is, the real power flow equations (B.1g) at buses $\mathcal{R}$ and the reactive power flow questions (B.1h) at buses in $\mathcal{B}\setminus\mathcal{D}$.

Steps 1 and 2 together allow us to complete all the decision variables.

B.1.4 Backpropagating through the completion

Let z denote the input to the completion procedure and let z_1 and z_2 denote the variables respectively derived during Steps 1 and 2 of completion. That is:

$$z \equiv \begin{bmatrix} (p_g)_{\mathcal{G}} \\ |v|_{\mathcal{B}\setminus\mathcal{D}} \end{bmatrix}, \qquad z_1 \equiv \begin{bmatrix} |v|_{\mathcal{D}} \\ (\angle v)_{\mathcal{B}\setminus\mathcal{R}} \end{bmatrix}, \qquad z_2 \equiv \begin{bmatrix} (p_g)_{\mathcal{R}} \\ (q_g)_{\mathcal{B}\setminus\mathcal{D}} \end{bmatrix}.$$

Then, we wish to backpropagate gradients of an arbitrary loss function $\ell(x, z, z_1, z_2)$, both in order to train our neural network and to perform our gradient-based correction procedure. That is, we must compute the total derivative $\mathrm{d}\ell/\mathrm{d}z$ given the partial derivatives $\partial\ell/\partial z, \partial\ell/\partial z_1$, and $\partial\ell/\partial z_2$.

Applying the chain rule through the two steps of the completion procedure, we have:

$$\frac{\mathrm{d}\ell}{\mathrm{d}z} = \frac{\partial\ell}{\partial z} + \frac{\partial\ell}{\partial z_1}\frac{\partial z_1}{\partial z} + \frac{\partial\ell}{\partial z_2}\frac{\partial z_2}{\partial z} + \frac{\partial\ell}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial z}. \tag{B.3}$$

We now consider each of these terms in this equality, except for $\partial\ell/\partial z, \partial\ell/\partial z_1, \partial\ell/\partial z_2$, which we may assume have already been computed.

Step 2. Let $J_{\text{Step 2}}$ denote the submatrix of J corresponding to the partial derivatives of the equality constraints used in Step 2 with respect to the voltage variables $|v|^T, (\angle v)^T$. Then, we have:

$$\frac{\partial \begin{bmatrix} (p_g)_{\mathcal{R}}^T & (q_g)_{\mathcal{B}\setminus\mathcal{D}}^T \end{bmatrix}^T}{\partial \begin{bmatrix} |v|^T & (\angle v)^T \end{bmatrix}^T} = -J_{\text{Step 2}}. \tag{B.4}$$

As $\frac{\partial (q_g)_{\mathcal{B}\setminus\mathcal{D}}}{\partial (p_g)_{\mathcal{G}}} = 0$, this gives us both the terms $\partial z_2/\partial z_1, \partial z_2/\partial z$ in (B.3).

Step 1. Consider the total differential through the equality constraints $(\text{B.1g})_{\mathcal{B}\setminus\mathcal{R}}$ and $(\text{B.1h})_{\mathcal{D}}$, where all terms besides W_r and W_i are viewed as parameters through which to differentiate. We rearrange this total differential to put differentials of input quantities to Step 1 on one side and differentials of outputs from Step 1 on the other side:

$$J_{\text{Step 1}} \begin{bmatrix} \mathrm{d}(|v|)_{\mathcal{D}} \\ \mathrm{d}(\angle v)_{\mathcal{B}\setminus\mathcal{R}} \end{bmatrix} = \begin{bmatrix} -\mathrm{d}(p_g)_{\mathcal{B}\setminus\mathcal{R}} + \mathrm{d}(p_d)_{\mathcal{B}\setminus\mathcal{R}} \\ \mathrm{d}(q_d)_{\mathcal{D}} \end{bmatrix} - J_{\text{Step 1b}} \begin{bmatrix} \mathrm{d}|v|_{\mathcal{B}\setminus\mathcal{D}} \\ \mathrm{d}(\angle v)_{\mathcal{R}} \end{bmatrix}, \tag{B.5}$$

where $J_{\text{Step 1}}$ is as in the forward pass of Step 1, and $J_{\text{Step 1b}}$ denotes the submatrix of J corresponding to the equality constraints $(\text{B.1g})_{\mathcal{B}\setminus\mathcal{R}}$ and $(\text{B.1h})_{\mathcal{D}}$ and the voltage variables $(|v|)_{\mathcal{B}\setminus\mathcal{D}}, (\angle v)_{\mathcal{R}}$.

While we could compute $\begin{bmatrix} \mathrm{d}|v|_{\mathcal{D}} \\ \mathrm{d}(\angle v)_{\mathcal{B}\setminus\mathcal{R}} \end{bmatrix}$ explicitly, we can improve efficiency by not computing and storing a large intermediate Jacobian explicitly. Instead, we can directly compute what we need, which is the product of this Jacobian with matrices of smaller dimensions. Define

$$K \equiv \left(\frac{\partial \ell}{\partial z_1} + \frac{\partial \ell}{\partial z_2} \frac{\partial z_2}{\partial z_1} \right) J_{\text{Step 1}}^{-1},$$

using our computation (B.4) above to evaluate $\partial z_2/\partial z_1$. Note furthermore that $J_{\text{Step 1}}^{-1}$ was already computed in the forward pass, meaning that we do not need to perform an additional matrix inversion.

Now, we can use (B.5) to derive:

$$\left(\frac{\partial \ell}{\partial z_1} + \frac{\partial \ell}{\partial z_2} \frac{\partial z_2}{\partial z_1} \right) \mathrm{d}z_1 = K \left(\begin{bmatrix} -\mathrm{d}(p_g)_{\mathcal{B}\setminus\mathcal{R}} + \mathrm{d}(p_d)_{\mathcal{B}\setminus\mathcal{R}} \\ \mathrm{d}(q_d)_{\mathcal{D}} \end{bmatrix} - J_{\text{Step 1b}} \begin{bmatrix} \mathrm{d}(|v|)_{\mathcal{B}\setminus\mathcal{D}} \\ \mathrm{d}(\angle v)_{\mathcal{R}} \end{bmatrix} \right),$$

which gives us

$$\frac{\partial \ell}{\partial z_1} \frac{\partial z_1}{\partial z} + \frac{\partial \ell}{\partial z_2} \frac{\partial z_2}{\partial z_1} \frac{\partial z_1}{\partial z}.$$

Putting it all together. Combining our logic described above for Steps 2 and 1, we can recover all terms in (B.3) and therefore identify $\mathrm{d}\ell/\mathrm{d}z$.

Appendix C

Enforcing Robust Control Guarantees within Neural Network Policies

C.1 Details on robust control specifications

As described in Section 7.3.1, for many dynamical systems of the form (7.1), it is possible to specify a set of linear, time-invariant policies guaranteeing infinite-horizon exponential stability via a set of LMIs. Here, we derive the LMI (7.4) provided in the main text for the NLDI system (7.3), and additionally describe relevant LMI systems for systems characterized by polytopic linear differential inclusions (PLDIs) and for H_∞ control settings.

C.1.1 Exponential stability in NLDIs

Consider the general NLDI system (7.3). We seek to design a time-invariant control policy $u(t) = Kx(t)$ and a quadratic Lyapunov function $V(x) = x^T P x$ with $P \succ 0$ for this system that satisfy the exponential stability criterion $\dot{V}(x) \leq -\alpha V(x),\ \forall t$. We derive an LMI characterizing such a controller and Lyapunov function, closely following and expanding upon the derivation provided in [Boy+94].

Specifically, consider the NLDI system (7.3), reproduced below:

$$\dot{x} = Ax + Bu + Gw, \quad \|w\|_2 \leq \|Cx + Du\|_2. \tag{C.1}$$

The time derivative of this Lyapunov function along the trajectories of the closed-loop system is

$$\begin{aligned}
\dot{V}(x) &= \dot{x}^T P x + x^T P \dot{x} \\
&= (Ax + Bu + Gw)^T P x + x^T P (Ax + Bu + Gw) \\
&= ((A + BK)x + Gw)^T P x + x^T P ((A + BK)x + Gw) \\
&= \begin{bmatrix} x \\ w \end{bmatrix}^T \begin{bmatrix} (A + BK)^T P + P(A + BK) & PG \\ G^T P & 0 \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix}.
\end{aligned} \tag{C.2}$$

The exponential stability condition $\dot{V}(x) \leq -\alpha V(x)$ is thus implied by inequality

$$\begin{bmatrix} x \\ w \end{bmatrix}^T M_1 \begin{bmatrix} x \\ w \end{bmatrix} := \begin{bmatrix} x \\ w \end{bmatrix}^T \begin{bmatrix} (A+BK)^T P + P(A+BK) + \alpha P & PG \\ G^T P & 0 \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} \leq 0. \tag{C.3}$$

Additionally, the norm bound on w can be equivalently expressed as

$$\begin{bmatrix} x \\ w \end{bmatrix}^T M_2 \begin{bmatrix} x \\ w \end{bmatrix} := \begin{bmatrix} x \\ w \end{bmatrix}^T \begin{bmatrix} (C+DK)^T (C+DK) & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} \geq 0. \tag{C.4}$$

Using the S-procedure, it follows that for some $\lambda \geq 0$, the following matrix inequality is a sufficient condition for exponential stability:

$$M_1 + \lambda M_2 \preceq 0. \tag{C.5}$$

Using Schur Complements, this matrix inequality is equivalent to

$$(A+BK)^T P + P(A+BK) + \alpha P + \lambda (C+DK)^T (C+DK) + \frac{1}{\lambda} PGG^T P \preceq 0. \tag{C.6}$$

Left- and right-multiplying both sides by P^{-1}, and making the change of variables $S = P^{-1}$, $Y = KS$, and $\mu = 1/\lambda$, we obtain

$$SA^T + AS + Y^T B^T + BY + \alpha S + \frac{1}{\mu} \left(SC^T + Y^T D^T\right) (CS + DY) + \mu GG^T \preceq 0. \tag{C.7}$$

Using Schur Complements again on this inequality, we obtain our final system of linear matrix inequalities as

$$\begin{bmatrix} AS + SA^T + \mu GG^T + BY + Y^T B^T + \alpha S & SC^T + Y^T D^T \\ CS + DY & -\mu I \end{bmatrix} \preceq 0, \quad S \succ 0, \quad \mu > 0, \tag{C.8}$$

where then $K = YS^{-1}$ and $P = S^{-1}$. Note that the first matrix inequality is homogeneous; we can therefore assume $\mu = 1$ (and therefore, $\lambda = 1$), without loss of generality.

C.1.2 Exponential stability in PLDIs

Consider the setting of polytopic linear differential inclusions (PLDIs), where the dynamics are of the form

$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \quad (A(t), B(t)) \in \text{Conv}\{(A_1, B_1), \ldots, (A_L, B_L)\}. \tag{C.9}$$

Here, $A(t) \in \mathbb{R}^{s \times s}$ and $B(t) \in \mathbb{R}^{s \times a}$ can vary arbitrarily over time, as long as they lie in the convex hull (denoted Conv) of the set of points above, where $A_i \in \mathbb{R}^{s \times s}, B_i \in \mathbb{R}^{s \times a}$ for $i = 1, \ldots, L$.

We seek to design a time-invariant control policy $u(t) = Kx(t)$ and quadratic Lyapunov function $V(x) = x^T P x$ with $P \succ 0$ for this system that satisfy the exponential stability criterion $\dot{V}(x) \leq -\alpha V(x)$, $\forall t$. Such a controller and Lyapunov function exist if there exist $S \in \mathbb{R}^{s \times s} \succ 0$ and $Y \in \mathbb{R}^{a \times s}$ such that

$$A_i S + B_i Y + S A_i^T + Y^T B_i^T + \alpha S \preceq 0, \quad \forall i = 1, \ldots, L, \tag{C.10}$$

where then $K = YS^{-1}$ and $P = S^{-1}$. The derivation of this LMI follows similarly to that for exponential stability in NLDIs, and is well-described in [Boy+94].

C.1.3 H_∞ control

Consider the following H_∞ control setting with linear time-invariant dynamics

$$\dot{x}(t) = Ax(t) + Bu(t) + Gw(t), \quad w \in \mathcal{L}_2, \tag{C.11}$$

where A, B, and G are time-invariant as for the NLDI case, and where we define $\mathcal{L}_2$ as the set of time-dependent signals with finite $\mathcal{L}_2$ norm.[1]

In cases such as these with larger or more unstructured disturbances, it may not be possible to guarantee asymptotic convergence to an equilibrium. In these cases, our goal is to construct a robust controller with bounds on the extent to which disturbances affect some performance output (e.g., LQR cost), as characterized by the $\mathcal{L}_2$ gain of the disturbance-to-output map. Specifically, we consider the stability requirement that this $\mathcal{L}_2$ gain be bounded by some parameter $\gamma > 0$ when disturbances are present, and that the system be exponentially stable in the disturbance-free case. This requirement can be characterized via the condition that for all t and some $\sigma \geq 0$,

$$\mathcal{E}(x, \dot{x}, u) := \dot{V}(x) + \alpha V(x) + \sigma \left(x^T Q x + u^T R u - \gamma^2 \|w\|_2^2\right) \leq 0. \tag{C.12}$$

We note that when $\mathcal{E}(x(t), \dot{x}(t), u(t)) \leq 0$ for all t, both of our stability criteria are met. To see this, note that integrating both sides of (C.12) from 0 to ∞ and ignoring the non-negative terms on the left hand side after integration yields

$$\int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t)) dt \leq \gamma^2 \int_0^\infty \|w(t)\|_2^2 dt + (1/\sigma) V(x(0)). \tag{C.13}$$

This is precisely the desired bound on the $\mathcal{L}_2$ gain of the disturbance-to-output map (see [KG02]). We also note that in the disturbance-free case, substituting $w = 0$ into (C.12) yields

$$\dot{V}(x) \leq -\alpha V(x) - \sigma \left(x^T Q x + u^T R u\right) \leq -\alpha V(x), \tag{C.14}$$

where the last inequality follows from the non-negativity of the LQR cost; this is precisely our condition for exponential stability.

We now seek to design a time-invariant control policy $u(t) = Kx(t)$ and quadratic Lyapunov function $V(x) = x^T P x$ with $P \succ 0$ that satisfies the above condition. In particular, we can write

$$\mathcal{E}\left(x(t), (A + BK)x(t) + Gw(t), Kx(t)\right) = \begin{bmatrix} x(t) \\ w(t) \end{bmatrix}^T M_1 \begin{bmatrix} x(t) \\ w(t) \end{bmatrix}, \tag{C.15}$$

where

$$M_1 := \begin{bmatrix} (A + BK)^T P + P(A + BK) + \alpha P + \sigma(Q + K^T R K) & PG \\ G^T P & -\gamma^2 \sigma I \end{bmatrix}. \tag{C.16}$$

[1] The $\mathcal{L}_2$ norm of a time-dependent signal $w(t)\colon [0, \infty) \to \mathbb{R}^d$ is defined as $\sqrt{\int_0^\infty \|w(t)\|_2^2 dt}$.

Therefore, we seek to find a $P \in \mathbb{R}^{s\times s} \succ 0$ and $K \in \mathbb{R}^{s\times a}$ that satisfy $M_1 \preceq 0$, for some design parameters $\alpha > 0$ and $\sigma > 0$. Using Schur complements, the matrix inequality $M_1 \preceq 0$ is equivalent to

$$(A+BK)^T P + P(A+BK) + \alpha P + \sigma(Q + K^T R K) + PGG^T P/(\gamma^2\sigma) \preceq 0. \tag{C.17}$$

As in Appendix C.1.1, we left- and right-multiply both sides by P^{-1}, and make the change of variables $S = P^{-1}$, $Y = KS$, and $\mu = 1/\sigma$ to obtain

$$SA^T + AS + Y^T B^T + BY + \alpha S + \frac{1}{\mu}\left((SQ^{1/2})(Q^{1/2}S) + (Y^T R^{1/2})(R^{1/2}Y)\right) + \mu GG^T/\gamma^2 \preceq 0.$$

Using Schur Complements again, we obtain the LMI

$$\begin{bmatrix} SA^T + AS + Y^T B^T + BY + \alpha S + \mu GG^T/\gamma^2 & \begin{bmatrix} SQ^{1/2} & Y^T R^{1/2}\end{bmatrix} \\ \begin{bmatrix} Q^{1/2}S \\ R^{1/2}Y \end{bmatrix} & -\mu I \end{bmatrix} \preceq 0, \quad S \succ 0, \quad \mu > 0, \tag{C.18}$$

where then $K = YS^{-1}$, $P = S^{-1}$, and $\sigma = 1/\mu$.

C.2 Derivation of sets of stabilizing policies and associated projections

We describe the construction of the set of actions $\mathcal{C}(x)$, defined in Equation (7.7), for PLDI systems (C.9) and H_∞ control settings (C.11). (The relevant formulations for the NLDI system (7.3) are described in the main text.)

C.2.1 Exponential stability in PLDIs

For the general PLDI system (C.9), relevant sets of exponentially stabilizing actions $\mathcal{C}_{\text{PLDI}}$ are given by the following theorem.

Theorem 2. *Consider the PLDI system (C.9), some stability parameter $\alpha > 0$, and a Lyapunov function $V(x) = x^T P x$ with P satisfying (C.10). Assuming P exists, define*

$$\mathcal{C}_{PLDI}(x) := \left\{ u \in \mathbb{R}^a \mid \begin{bmatrix} 2x^T P B_1 \\ 2x^T P B_2 \\ \vdots \\ 2x^T P B_L \end{bmatrix} u \leq - \begin{bmatrix} x^T(\alpha P + 2PA_1)x \\ x^T(\alpha P + 2PA_2)x \\ \vdots \\ x^T(\alpha P + 2PA_L)x \end{bmatrix} \right\}$$

for all states $x \in \mathbb{R}^s$. For all x, $\mathcal{C}_{PLDI}(x)$ is a non-empty set of actions that satisfy the exponential stability condition (7.2). Further, $\mathcal{C}_{PLDI}(x)$ is a convex set in u.

Proof. We seek to find a set of actions such that the condition (7.2) is satisfied along *all* possible trajectories of (C.9), i.e., for *any* allowable instantiation of $(A(t), B(t))$. A set of actions satisfying this condition at a given x is given by

$$\mathcal{C}_{\text{PLDI}}(x) := \{u \in \mathbb{R}^a \mid \dot{V}(x) \leq -\alpha V(x) \ \ \forall (A(t), B(t)) \in \text{Conv}\{(A_1, B_1), \ldots, (A_L, B_L)\}.$$

Expanding the left side of the inequality above, we see that for some coefficients $\gamma_i \in \mathbb{R} \geq 0, i = 1, \ldots, L$ satisfying $\sum_{i=1}^{L} \gamma_i(t) = 1$,

$$\begin{aligned}\dot{V}(x) &= \dot{x}^T P x + x^T P \dot{x} = 2x^T P \left(A(t)x + B(t)u\right) \\ &= 2x^T P \left(\sum_{i=1}^{L} \gamma_i(t) A_i x + \gamma_i(t) B_i u\right) = \sum_{i=1}^{L} \gamma_i \left(2x^T P \left(A_i x + B_i u\right)\right)\end{aligned}$$

by definition of the PLDI dynamics and of the convex hull. Thus, if we can ensure

$$2x^T P \left(A_i x + B_i u\right) \leq -\alpha V(x) = -\alpha x^T P x, \ \forall i = 1, \ldots, L,$$

then we can ensure that exponential stability holds. Rearranging this condition and writing it in matrix form yields an inequality of the desired form. We note that by definition of the specifications (C.10), there is some K corresponding to P such that the policy $u = Kx$ satisfies all of the above inequalities; thus, $Kx \in \mathcal{C}_{\text{PLDI}}(x)$, and $\mathcal{C}_{\text{PLDI}}(x)$ is non-empty. Further, as the above inequality represents a linear constraint in u, this set is convex in u. □

We note that the relevant projection $\mathcal{P}_{\mathcal{C}_{\text{PLDI}}(x)}$ represents a projection onto an intersection of halfspaces, and can thus be implemented via differentiable quadratic programming [AK17].

C.2.2 H_∞ control

For the H_∞ control system (C.11), relevant sets of actions satisfying the condition (C.12) are given by the following theorem.

Theorem 3. *Consider the system (C.11), some stability parameter $\alpha > 0$, and a Lyapunov function $V(x) = x^T P x$ with P satisfying Equation (C.18). Assuming P exists, define*

$$\mathcal{C}_{H_\infty}(x) := \left\{u \in \mathbb{R}^a \mid u^T R u + (2B^T P x)^T u + x^T \left(PA + A^T P + \alpha P + Q + \gamma^{-2} P G G^T P\right) x \leq 0\right\}$$

for all states $x \in \mathbb{R}^s$. For all x, $\mathcal{C}_{H_\infty}(x)$ is a non-empty set of actions that guarantee condition (C.12), i.e., that the $\mathcal{L}_2$ gain of the disturbance-to-output map is bounded by γ and that the system is exponentially stable in the disturbance-free case. Further, $\mathcal{C}_{H_\infty}(x)$ is convex in u.

Proof. We seek to find a set of actions such that the condition $\mathcal{E}(x, \dot{x}, u) \leq 0$ is satisfied along *all* possible trajectories of (C.11), where $\mathcal{E}$ is defined as in (C.12). A set of actions satisfying this condition at a given x is given by

$$\mathcal{C}_{H_\infty}(x) := \{u \in \mathbb{R}^a \mid \sup_{w \in \mathcal{L}_2} \ \mathcal{E}(x, \dot{x}, u) \leq 0, \ \dot{x} = Ax + Bu + Gw\}.$$

To begin, we note that

$$\begin{aligned}\mathcal{E}(x, Ax + Bu + Gw, u) = x^T P(Ax + Bu + Gw) + (Ax + Bu + Gw)^T P x + \alpha x^T P x \\ + \sigma \left(x^T Q x + u^T R u - \gamma^2 \|w\|_2^2\right)\end{aligned}$$

We then maximize $\mathcal{E}$ over w:

$$w^\star = \arg\max_w \ \mathcal{E}(x, Ax + Bu + Gw, u) = G^T P x/(\sigma\gamma^2). \tag{C.19}$$

Therefore,

$$\mathcal{C}_{H_\infty}(x) = \{u \mid \mathcal{E}(x, Ax + Bu + Gw^\star, u, w^\star) \leq 0\}. \tag{C.20}$$

Expanding and rearranging terms, this becomes

$$\mathcal{C}_{\mathcal{H}_\infty}(x) = \{u \mid u^T(\sigma R)u + (2B^T Px)^T u + x^T \left(PA + A^T P + \alpha P + \sigma Q + PGG^T P/(\sigma\gamma^2)\right) x \leq 0\}. \tag{C.21}$$

We note that by definition of the specifications (C.18), there is some K corresponding to P such that the policy $u = Kx$ satisifies the conditions above (see (C.17)); thus, $Kx \in \mathcal{C}_{H_\infty}$, and $\mathcal{C}_{H_\infty}$ is non-empty. We note further that $\mathcal{C}_{\mathcal{H}_\infty}$ is an ellipsoid in the control action space, and is thus convex in u. □

We rewrite the set $\mathcal{C}_{H_\infty}(x)$ such that the projection $\mathcal{P}_{\mathcal{C}_{H_\infty}(x)}$ can be viewed as a second-order cone projection, in order to leverage our fast custom solver (Appendix C.3). In particular, defining $\tilde{P} = \sigma R$, $\tilde{q} = B^T Px$, and $\tilde{r} = x^T \left(PA + A^T P + \alpha P + \sigma Q + PGG^T P/(\sigma\gamma^2)\right) x$, we can rewrite the ellipsoid above as

$$\mathcal{C}_{H_\infty}(x) = \{u \mid u^\top \tilde{P} u + 2\tilde{q}^\top u + \tilde{r} \leq 0\}. \tag{C.22}$$

We note that as $\tilde{P} \succ 0$ and $\tilde{r} - \tilde{q}^\top \tilde{P}^{-1}\tilde{q} < 0$, this ellipsoid is non-empty (see, e.g., section B.1 in [BV04]). We can then rewrite the ellipsoid as

$$\mathcal{C}_{H_\infty}(x) = \{u \mid \|\tilde{A}u + \tilde{b}\|_2 \leq 1\} \tag{C.23}$$

where $\tilde{A} = \sqrt{\frac{\tilde{P}}{\tilde{q}^\top \tilde{P}^{-1}\tilde{q} - \tilde{r}}}$ and $\tilde{b} = \sqrt{\frac{P}{q^\top P^{-1} q - r}} P^{-1} q$. The constraint $\|\tilde{A}u + \tilde{b}\|_2 \leq 1$ is then a second-order cone constraint in u.

C.3 A fast, differentiable solver for second-order cone projection

In order to construct the robust policy class described in Section 7.4 for the general NLDI system (7.3) and the H_∞ setting (C.11), we must project a nominal (neural network-based) policy onto the second-order cone constraints described in Theorem 1 and Appendix C.2.2, respectively. As this projection operation does not necessarily have a closed form, we implement it via a custom differentiable optimization solver.

More generally, consider a set of the form

$$\mathcal{C} = \{x \in \mathbb{R}^n \mid \|Ax + b\|_2 \leq c^T x + d\} \tag{C.24}$$

for some $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $d \in \mathbb{R}$. Given some input $y \in \mathbb{R}^n$, we seek to compute the second-order cone projection $\mathcal{P}_{\mathcal{C}}(y)$ by solving the problem

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\|x-y\|_2^2 \\ \text{subject to} & \|Ax+b\|_2 \leq c^T x + d. \end{array} \tag{C.25}$$

Let $\mathcal{F}$ denote the ℓ_2 norm cone, i.e., $\mathcal{F} := \{(w,t) \mid \|w\|_2 \leq t\}$. Introducing the auxiliary variable $z \in \mathbb{R}^{m+1}$, we can then rewrite the above optimization problem equivalently as

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n,\, z\in\mathbb{R}^{m+1}}{\text{minimize}} & \frac{1}{2}\|x-y\|_2^2 + \mathbf{1}_{\mathcal{F}}(z) \\ \text{subject to} & z = \begin{bmatrix} Ax+b \\ c^T x + d \end{bmatrix} =: Gx + h, \end{array} \tag{C.26}$$

where for brevity we define $G = \begin{bmatrix} A \\ c^T \end{bmatrix}$ and $h = \begin{bmatrix} b \\ d \end{bmatrix}$, and where $\mathbf{1}_{\mathcal{F}}$ denotes the indicator function for membership in the set $\mathcal{F}$.

We describe our fast solution technique for computing this projection, as well as our method for obtaining gradients through the solution.

C.3.1 Computing the projection

We construct a fast solver for problem (C.26) using an accelerated projected dual gradient method. Specifically, define $\mu = \mathbb{R}^{m+1}$ as the dual variable on the equality constraint in Equation (C.26). The Lagrangian for this problem can then be written as

$$\mathscr{L}(x,z,\mu) = \frac{1}{2}\|x-y\|_2^2 + \mathbf{1}_{\mathcal{F}}(z) + \mu^T(z - Gx - h), \tag{C.27}$$

and the dual problem is given by $\max_{\mu} \min_{x,z} \mathscr{L}(x,z,\mu)$. To form the dual problem, we minimize the Lagrangian with respect to x and z as

$$\inf_{x,z} \mathscr{L}(x,z,\mu) = \inf_{x} \ \frac{1}{2}\left\{\|x-y\|_2^2 - \mu^T G x\right\} + \inf_{z}\{\mu^T z + \mathbf{1}_{\mathcal{F}}(z)\} - \mu^T h. \tag{C.28}$$

We note that the first term on the right side is minimized at $x^\star(\mu) = y + G^T\mu$. Thus, we see that

$$\inf_{x} \ \frac{1}{2}\{\|x-y\|_2^2 - \mu^T G x\} = -\frac{1}{2}\mu^T G G^T \mu - \mu^T G y. \tag{C.29}$$

For the second term, denote $\mu = (\tilde{\mu}, s)$ and $z = (\tilde{z}, t)$. We can then rewrite this term as

$$\inf_{z}\{\mu^T z + \mathbf{1}_{\mathcal{F}}(z)\}\} = \inf_{t\geq 0} \ \inf_{\tilde{z}} \ \{t\cdot s + \tilde{\mu}^T\tilde{z} \mid \|\tilde{z}\|_2 \leq t\}. \tag{C.30}$$

For a fixed $t \geq 0$, the above objective is minimized at $\tilde{z} = -t\tilde{\mu}/\|\tilde{\mu}\|_2$. (The problem is infeasible for $t < 0$.) Substituting this minimizer into (C.30) and minimizing the result over $t \geq 0$ yields

$$\inf_{z}\{\mu^T z + \mathbf{1}_{\mathcal{F}}(z)\} = \inf_{t\geq 0} \ t(s - \|\tilde{\mu}\|_2) = -\mathbf{1}_{\mathcal{F}}(\mu) \tag{C.31}$$

where the last identity follows from definition of the second-order cone $\mathcal{F}$. Hence the negative dual problem becomes

$$\underset{\mu}{\text{minimize}} \quad \frac{1}{2}\mu^T GG^T\mu + \mu^T(Gy+h) + \mathbf{1}_{\mathcal{F}}(\mu). \tag{C.32}$$

We now solve this problem via Nesterov's accelerated projected dual gradient method [Nes13]. For notational brevity, define $f(\mu) := \frac{1}{2}\mu^T GG^T\mu + \mu^T(Gy+h)$. Then, starting from arbitrary $\mu^{(-1)}, \mu^{(0)} \in \mathbb{R}^{m+1}$ we perform the iterative updates

$$\begin{aligned} \nu^{(k)} &= \mu^{(k)} + \beta^{(k)}(\mu^{(k)} - \mu^{(k-1)}) \\ \mu^{(k+1)} &= \mathcal{P}_{\mathcal{F}}\left(\nu^{(k)} - \frac{1}{L_f}\nabla f(\nu^{(k)})\right), \end{aligned} \tag{C.33}$$

where $L_f = \lambda_{\max}(GG^T)$ is the Lipschitz constant of f, and $\mathcal{P}_{\mathcal{F}}$ is the projection operator onto $\mathcal{F}$ (which has a closed form solution; see [Bau96]). Letting $m_f = \lambda_{\min}(GG^T)$ denote the strong convexity constant of f, the momentum parameter is then scheduled as [Nes13]

$$\beta^k = \begin{cases} \dfrac{k-1}{k+2} & \text{if } m_f = 0 \\ \dfrac{\sqrt{L_f} - \sqrt{m_f}}{\sqrt{L_f} + \sqrt{m_f}} & \text{if } m_f > 0. \end{cases} \tag{C.34}$$

After computing the optimal dual variable $\mu^\star$, i.e., the fixed point of (C.33), the optimal primal variable can be recovered via the equation $x^\star = y + G^T\mu^\star$ (as can be observed from the first-order conditions of the Lagrangian (C.27)).

C.3.2 Obtaining gradients

In order to incorporate the above projection into our neural network, we need to compute the gradients of all problem variables (i.e., G, h, and y) through the solution $x^\star$. In particular, we note that $x^\star$ has a direct dependence on both G and y, and an indirect dependence on all of G, h, and y through $\mu^\star$.

To compute the relevant gradients through $\mu^\star$, we apply the implicit function theorem to the fixed point of the update equations (C.33). Specifically, as these updates imply that $\mu^\star = \nu^\star$, their fixed point can be written as

$$\mu^\star = \mathcal{P}_{\mathcal{F}}\left(\mu^\star - \frac{1}{L_f}\nabla f(\mu^\star)\right). \tag{C.35}$$

Define $M := \frac{\partial \mathcal{P}_{\mathcal{F}}(\cdot)}{\partial(\cdot)}\big|_{(\cdot)=\mu^\star - \frac{1}{L_f}\nabla f(\mu^\star)}$, and note that $\nabla f(\mu^\star) = GG^T\mu^\star + Gy + h$. The differential of the above fixed-point equation is then given by

$$\mathrm{d}\mu^\star = M \times \left(\mathrm{d}\mu^\star - \frac{1}{L_f}\left(\mathrm{d}GG^T\mu^\star + G\mathrm{d}G^T\mu^\star + GG^T\mathrm{d}\mu^\star + \mathrm{d}Gy + G\mathrm{d}y + \mathrm{d}h\right)\right). \tag{C.36}$$

Rearranging terms to separate the differentials of problem outputs from problem variables, we see that

$$\left(I - M + \frac{1}{L_f} MGG^T\right) \mathrm{d}\mu^\star = -\frac{1}{L_f} M \left(\mathrm{d}GG^T\mu^\star + G\mathrm{d}G^T\mu^\star + \mathrm{d}Gy + G\mathrm{d}y + \mathrm{d}h\right), \quad \text{(C.37)}$$

where I is the identity matrix of appropriate size.

As described in e.g. [AK17], we can then use these equations to form the Jacobian of $\mu^\star$ with respect to any of the problem variables by setting the differential of the relevant problem variable to I and of all other problem variables to 0; solving the resulting equation for $\mathrm{d}\mu^\star$ then yields the value of the desired Jacobian. However, as these Jacobians can be large depending on problem size, we rarely want to form them explicitly. Instead, given some backward pass vector $\frac{\partial \ell}{\partial \mu^\star} \in \mathbb{R}^{1\times(m+1)}$ with respect to the optimal dual variable, we want to directly compute the gradient of the loss with respect to the problem variables: e.g., for y, we want to directly form the result of the product $\frac{\partial \ell}{\partial \mu^\star}\frac{\partial \mu^\star}{\partial y} \in \mathbb{R}^{1\times n}$. We do this via a similar method as presented in [AK17], and refer the reader there for a more in-depth explanation of the method described below.

Define $J := I - M + \frac{1}{L_f} MGG^T$ to represent the coefficient of $\mathrm{d}\mu^\star$ on the left side of Equation (C.37). Given $\frac{\partial \ell}{\partial \mu^\star}$, we then compute the intermediate term

$$\mathrm{d}_\mu := -J^{-T}\left(\frac{\partial \ell}{\partial \mu^\star}\right)^T. \quad \text{(C.38)}$$

We can then form the relevant gradient terms directly as

$$\begin{aligned}
\left(\frac{\partial \ell}{\partial \mu^\star}\frac{\partial \mu^\star}{\partial G}\right)^T &= \frac{1}{L_f} M \left(\mathrm{d}_\mu (G^T\mu^\star)^T + \mu^\star (G^T \mathrm{d}_\mu)^T + \mathrm{d}_\mu y^T\right) \\
\left(\frac{\partial \ell}{\partial \mu^\star}\frac{\partial \mu^\star}{\partial h}\right)^T &= \frac{1}{L_f} M \mathrm{d}_\mu \\
\left(\frac{\partial \ell}{\partial \mu^\star}\frac{\partial \mu^\star}{\partial y}\right)^T &= \frac{1}{L_f} G^T M \mathrm{d}_\mu.
\end{aligned} \quad \text{(C.39)}$$

In these computations, we note that as our solver returns $x^\star$, the backward pass vector we are given is actually $\frac{\partial \ell}{\partial x^\star} \in \mathbb{R}^{1\times n}$; thus, we compute $\frac{\partial \ell}{\partial \mu^\star} = \frac{\partial \ell}{\partial x^\star}\frac{\partial x^\star}{\partial \mu^\star} = \frac{\partial \ell}{\partial x^\star} G^T$ for use in Equation (C.38).

Accounting additionally for the direct dependence of some of the problem variables on

$x^\star$ (recalling that $x^\star = y + G^T u^\star$), the desired gradients are then given by

$$\begin{aligned}
\left(\frac{\partial \ell}{\partial G}\right)^T &= \left(\frac{\partial \ell}{\partial x^\star}\frac{\partial x^\star}{\partial G} + \frac{\partial \ell}{\partial x^\star}\frac{\partial x^\star}{\partial u^\star}\frac{\partial u^\star}{\partial G}\right)^T = \mu^\star \frac{\partial \ell}{\partial x^\star} + \frac{1}{L_f} M \left(\mathrm{d}_\mu (G^T \mu^\star)^T + \mu^\star (G^T \mathrm{d}_\mu)^T + \mathrm{d}_\mu y^T\right) \\
\left(\frac{\partial \ell}{\partial h}\right)^T &= \left(\cancel{\frac{\partial \ell}{\partial x^\star}\frac{\partial x^\star}{\partial h}}^{0} + \frac{\partial \ell}{\partial x^\star}\frac{\partial x^\star}{\partial u^\star}\frac{\partial u^\star}{\partial h}\right)^T = \frac{1}{L_f} M \mathrm{d}_\mu \\
\left(\frac{\partial \ell}{\partial y}\right)^T &= \left(\frac{\partial \ell}{\partial x^\star}\frac{\partial x^\star}{\partial y} + \frac{\partial \ell}{\partial x^\star}\frac{\partial x^\star}{\partial u^\star}\frac{\partial u^\star}{\partial y}\right)^T = \left(\frac{\partial \ell}{\partial x^\star}\right)^T + \frac{1}{L_f} G^T M \mathrm{d}_\mu.
\end{aligned} \tag{C.40}$$

C.4 Writing the cart-pole problem as an NLDI

In the cart-pole task, our goal is to balance an inverted pendulum resting on top of a cart by exerting horizontal forces on the cart. Specifically, the state of this system is defined as $x = \begin{bmatrix} p_x, \dot{p_x}, \varphi, \dot{\varphi} \end{bmatrix}^T$, where p_x is the cart position and φ is the angular displacement of the pendulum from its vertical position; we seek to stabilize the system at $x = \vec{0}$ by exerting horizontal forces $u \in \mathbb{R}$ on the cart. For a pendulum of length ℓ and mass m_p, and for a cart of mass m_c, the dynamics of the system are (as described in [Ted09]):

$$\dot{x} = \begin{bmatrix} \dot{p_x} \\ \frac{u + m_p \sin\varphi (\ell \dot{\varphi}^2 - g\cos\varphi)}{m_c + m_p \sin^2\varphi} \\ \dot{\varphi} \\ \frac{(m_c + m_p) g \sin\varphi - u\cos\varphi - m_p \ell \dot{\varphi}^2 \cos\varphi \sin\varphi}{l(m_c + m_p \sin^2\varphi)} \end{bmatrix}, \tag{C.41}$$

where $g = 9.81$ m/s^2 is the acceleration due to gravity. We rewrite this system as an NLDI by defining $\dot{x} = f(x, u)$ and then linearizing the system about its equilibrium point as

$$\dot{x} = \mathrm{J}_f(0,0) \begin{bmatrix} x \\ u \end{bmatrix} + I_n w, \quad \|w\| \leq \|Cx + Du\|, \tag{C.42}$$

where J_f is the Jacobian of the dynamics, $w = f(x,u) - \mathrm{J}_f(0,0) \begin{bmatrix} x & u \end{bmatrix}^T$ is the linearization error, and I_n is the $n \times n$ identity matrix. We bound this linearization error by numerically obtaining the matrices C and D, assuming that x and u are within a neighborhood of the origin. We describe this process in more detail below. As a note, while we employ an NLDI here to characterize the linearization error, it is also possible to characterize this error via polytopic uncertainty (see Appendix C.10); we choose to use an NLDI here as it yields a much smaller problem description than a PLDI in this case.

C.4.1 Deriving $\mathrm{J}_f(0,0)$

For $\dot{x} = f(x,u)$, we see that

$$\mathrm{J}_f(x,u) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \partial\ddot{p_x}/\partial\varphi & \partial\ddot{p_x}/\partial\dot{\varphi} & \partial\ddot{p_x}/\partial u \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \partial\ddot{\varphi}/\partial\varphi & \partial\ddot{\varphi}/\partial\dot{\varphi} & \partial\ddot{\varphi}/\partial u, \end{bmatrix}, \tag{C.43}$$

where

$$\frac{\partial\ddot{p_x}}{\partial\varphi} = \frac{m_p\cos\varphi\left(\dot{\varphi}^2 l - g\cos\varphi\right) + g m_p\sin^2\varphi}{m_c + m_p\sin^2\varphi} - \frac{2m_p\sin\varphi\cos\varphi\left(m_p\sin\varphi\left(\dot{\varphi}^2 l - g\cos\varphi\right) + u\right)}{\left(m_c + m_p\sin^2\varphi\right)^2},$$

$$\frac{\partial\ddot{p_x}}{\partial\dot{\varphi}} = \frac{2\dot{\varphi} l m_p\sin\varphi}{m_c + m_p\sin^2\varphi},$$

$$\frac{\partial\ddot{p_x}}{\partial u} = \frac{1}{m_c + m_p\sin^2\varphi},$$

$$\frac{\partial\ddot{\varphi}}{\partial\varphi} = \frac{\begin{matrix} g(m_c+m_p)\cos\varphi + \dot{\varphi}^2 l m_p\sin^2\varphi \\ -\dot{\varphi}^2 l m_p\cos^2\varphi + u\sin\varphi \end{matrix}}{l\left(m_c + m_p\sin^2\varphi\right)} - \frac{\begin{matrix} 2m_p\sin\varphi\cos\varphi(g(m_c+m_p)\sin\varphi \\ -\dot{\varphi}^2 l m_p\sin\varphi\cos\varphi - u\cos\varphi \end{matrix}}{l\left(m_c + m_p\sin^2\varphi\right)^2},$$

$$\frac{\partial\ddot{\varphi}}{\partial\dot{\varphi}} = \frac{-2\dot{\varphi} m_p\sin\varphi\cos\varphi}{m_c + m_p\sin^2\varphi},$$

$$\frac{\partial\ddot{\varphi}}{\partial u} = \frac{-\cos\varphi}{l(m_c + m_p\sin^2\varphi)}.$$

We thus see that

$$\mathrm{J}_f(0,0) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -m_p g/m_c & 0 & 1/m_c \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & g(m_c+m_p)/l m_c & 0 & -1/m_c \end{bmatrix}. \tag{C.44}$$

C.4.2 Obtaining C and D

We then seek to construct matrices C and D that bound the linearization error w between the true dynamics $\dot{x}$ and our first-order linear approximation $\mathrm{J}_f(0,0)\begin{bmatrix} x \\ u \end{bmatrix}$. To do so, we bound the error of this approximation entry-wise: that is, for each entry $i = 1,\ldots,s$, we want to find F_i such that for all x in some region $\underline{x} \le x \le \bar{x}$, and all u in some region $\underline{u} \le u \le \bar{u}$,

$$w_i^2 = \left(\nabla f_i(0)\begin{bmatrix} x \\ u \end{bmatrix} - \dot{x}_i\right)^2 \le \begin{bmatrix} x \\ u \end{bmatrix}^T F_i \begin{bmatrix} x \\ u \end{bmatrix}. \tag{C.45}$$

Then, given the matrix

$$M = \begin{bmatrix} F_1^{T/2} & F_2^{T/2} & F_3^{T/2} & F_4^{T/2} & F_5^{T/2} & F_6^{T/2} \end{bmatrix}^T \tag{C.46}$$

we can then obtain $C = M_{1:s}$ and $D = M_{s:s+m}$ (where the subscripts indicate column-wise indexing).

We solve separately for each F_i to minimize the difference between the right and left sides of Equation (C.45) (while enforcing that the right side is larger than the left side) over a discrete grid of points within $\underline{x} \leq x \leq \bar{x}$ and $\underline{u} \leq u \leq \bar{u}$. By assuming that F_i is symmetric, we are able to cast this as a linear program in the upper triangular entries of F_i.

To obtain the matrices C and D used for the cart-pole experiments in the main paper, we let $\bar{x} = \begin{bmatrix} 1.5 & 2 & 0.2 & 1.5 \end{bmatrix}^T$, $\bar{u} = 10$, $\underline{x} = -\bar{x}$, and $\underline{u} = -\bar{u}$. As each entry-wise difference in Equation (C.45) contained exactly three variables (i.e., a total of three entries from x and u), we solved each entry-wise linear program over a mesh grid of 50 points per variable.

C.5 Writing quadrotor as an NLDI

In the planar quadrotor setting, our goal is to stabilize a quadcopter in the two-dimensional plane by controlling the amount of force provided by the quadcopter's right and left thrusters. Specifically, the state of this system is defined as $x = \begin{bmatrix} p_x & p_z & \varphi & \dot{p_x} & \dot{p_z} & \dot{\varphi} \end{bmatrix}^T$, where (p_x, p_z) is the position of the quadcopter in the vertical plane and φ is its roll (i.e., angle from the horizontal position); we seek to stabilize the system at $x = \vec{0}$ by controlling the amount of force $u = \begin{bmatrix} u_r, u_l \end{bmatrix}^T$ from right and left thrusters. We assume that our action u is additional to a baseline force of $\begin{bmatrix} mg/2 & mg/2 \end{bmatrix}^T$ provided by the thrusters by default to prevent the quadcopter from falling. For a quadrotor with mass m, moment-arm ℓ for the thrusters, and moment of inertia J about the roll axis, the dynamics of this system are then given by (as modified from [Sin+20]):

$$\dot{x} = \begin{bmatrix} \dot{p_x} \cos\varphi - \dot{p_z} \sin\varphi \\ \dot{p_x} \sin\varphi + \dot{p_z} \cos\varphi \\ \dot{\varphi} \\ \dot{p_z}\dot{\varphi} - g \sin\varphi \\ -\dot{p_x}\dot{\varphi} - g \cos\varphi + g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1/m & 1/m \\ \ell/J & -\ell/J \end{bmatrix} u, \tag{C.47}$$

where $g = 9.81$ m/s^2. We linearize this system via a similar method as for the cart-pole setting, i.e., as in Equation (C.42). We describe this process in more detail below. We note that since the dependence of the dynamics on u is linear, we have that $D = 0$ for our resultant NLDI. As for cart-pole, while we employ an NLDI here to characterize the linearization error, it is also possible to characterize this error via polytopic uncertainty (see Appendix C.10); we choose to use an NLDI here as it yields a much smaller problem description than a PLDI in this case.

C.5.1 Deriving $\mathrm{J}_f(0,0)$

For $\dot{x} = f(x,u)$, we see that

$$\mathrm{J}_f(x,u) = \begin{bmatrix} 0 & 0 & -\dot{p}_x \sin\varphi - \dot{p}_z \cos\varphi & \cos\varphi & -\sin\varphi & 0 & 0 \\ 0 & 0 & \dot{p}_x \cos\varphi - \dot{p}_z \sin\varphi & \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -g\cos\varphi & 0 & \dot{\varphi} & \dot{p}_z & 0 \\ 0 & 0 & g\sin\varphi & -\dot{\varphi} & 0 & -\dot{p}_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{C.48}$$

and thus

$$\mathrm{J}_f(0,0) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{C.49}$$

C.5.2 Obtaining C and D

We obtain the matrices C and D via a similar method as described in Appendix C.4, though in practice we only consider the linearization error with respect to x (i.e., since the dynamics are linear with respect to u, we have $D = 0$). We let $\bar{x} = \begin{bmatrix} 1 & 1 & 0.15 & 0.6 & 0.6 & 1.3 \end{bmatrix}$ and $\underline{x} = -\bar{x}$. As for cart-pole, each entry wise difference in the equivalent of Equation (C.45) contained exactly three variables (i.e., a total of three entries from x and u), and each entry-wise linear program was solved over a mesh grid of 50 points per variable.

C.6 Details on the microgrid setting

For our experiments, we build upon the microgrid setting given in [LBR16]. In this system, the state $x \in \mathbb{R}^3$ captures voltage deviations, frequency deviations, and the amount of power generated by a diesel generator connected to the grid; the action $u \in \mathbb{R}^2$ describes the current associated with a storage device and a solar PV inverter; and the disturbance $w \in \mathbb{R}$ describes the difference between the amount of power demanded and the amount of power produced by solar panels on the grid. The authors also define a performance index $y \in \mathbb{R}^2$ which captures voltage and frequency deviations (i.e., two of the entries of the state x).

To construct an NLDI of the form (7.3) for this system, we directly use the A, B, and G matrices given in [LBR16]. We generate C i.i.d. from a normal distribution and let $D = 0$, to represent the fact that the disturbance w and the entries of the state x are correlated, but that w is likely not correlated with the actions u. Finally, we let Q and R be diagonal matrices with 1 in the entries corresponding to quantities represented in the performance index y, and with 0.1 in the rest of the diagonal entries, to emphasize that the variables in y are the most important in describing the performance of the system.

C.7 Generating an adversarial disturbance

In the NLDI settings explored in our experiments, we seek to construct an "adversarial" disturbance $w(t)$ that obeys the relevant norm bounds $\|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2$ while maximizing the loss. To do this, we use a model predictive control method where the actions taken are $w(t)$. Specifically, for each policy π, we model $w(t)$ as a neural network specific to that policy. Every 10 steps of a roll-out, we optimize $w(t)$ through gradient descent to maximize the loss over a horizon of 40 steps, subject to the constraint $\|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2$.

C.8 Additional experimental details

Initial states. To pick initial states in our experiments, for the synthetic settings, we sample each attribute of the state i.i.d. from a standard Gaussian distribution. For cart-pole and planar quadrotor, we sample uniformly from bounds chosen such that the non-robust LQR algorithm (under the original dynamics) did not go unstable. For cart-pole, these bounds were chosen to be $p_x \in [-1, 1]$, $\varphi \in [-0.1, 0.1]$, $\dot{p}_x = \dot{\varphi} = 0$. For planar quadrotor, these bounds were $p_x, p_z \in [-1, 1]$, $\varphi \in [-0.05, 0.05]$, $\dot{p}_x = \dot{p}_z = \dot{\varphi} = 0$.

Constructing NLDI bounds. Given these initial states, for the cart-pole and quadrotor settings, we needed to construct our NLDI disturbance bounds such that they would hold over the *entire* trajectory of the robust policy; if not, the robustness specification (C.8) would not hold, and our agent might in fact increase the Lyapunov function. To ensure this approximately, we used a simple heuristic: we ran the (non-robust) LQR agent for a full episode with 50 different starting conditions, and constructed an L_∞ ball around all states reached in any of these trajectories. We then used these L_∞ balls on the states to construct the matrices C and D for our disturbance bounds, using the procedure described in Appendices C.4 and C.5.

Computing infrastructure and runtime. All experiments were run on an XPS 13 laptop with an Intel i7 processor. The planar quadrotor and synthetic NLDI experiment with $D = 0$ took about 1 day to run (since the projections were simple half-space projections), while all the other synthetic domains and cart-pole took about 3 days to run. The majority of the run-time was in computing the adversarial disturbances for test-time evaluations.

Hyperparameter selection. For our experiments, we did not perform large parameter searches. The learning rate we chose for our model-based planner, (both robust and non-robust) remained constant for the different domains; we tried learning rates of $1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}$ and found 1×10^{-3} worked best for the non-robust version and 1×10^{-4} worked best for the robust version. For our PPO hyperparameters, we simply used those used in the original PPO paper.

One parameter we had to tune for each environment was the time step. In particular, we had to pick a time step high enough that we could run episodes for a reasonable total

Table C.1: Time (in seconds) taken to run each method on the test set of every environment for 50 episodes run in parallel.

Environment	LQR	MBP	PPO	Robust LQR	Robust MPC	RARL	Robust MBP*	Robust PPO*
Generic NLDI ($D = 0$)	0.63	0.61	0.84	0.57	718.06	0.71	0.73	0.94
Generic NLDI ($D \neq 0$)	0.64	0.62	0.83	0.58	824.86	0.81	15.13	25.38
Cart-pole	0.55	0.67	0.84	0.53	646.90	0.84	10.12	13.37
Quadrotor	0.95	0.98	1.19	0.88	3348.68	1.14	1.15	1.30
Microgrid	0.58	0.61	0.79	0.57	601.90	0.74	8.14	10.25
Generic PLDI	0.57	0.54	0.76	0.51	819.24	0.73	69.35	64.03
Generic H_∞	0.84	0.80	1.03	0.76	N/A	1.00	47.81	63.67

Table C.2: Time (in minutes) taken to train each method in every environment.

Environment	MBP	PPO	RARL	Robust MBP*	Robust PPO*
Generic NLDI ($D = 0$)	26.36	101.77	102.37	30.78	114.60
Generic NLDI ($D \neq 0$)	26.46	100.79	82.53	221.35	1158.28
Cart-pole	25.49	87.04	98.90	146.34	689.93
Quadrotor	41.24	131.48	112.95	46.13	159.06
Microgrid	23.03	112.52	87.71	113.61	436.64

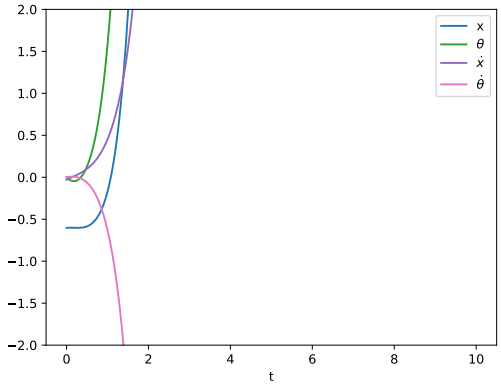
length of time (within which the non-robust agents would go unstable), but low enough to reasonably approximate a continuous-time setting (since, for our robustness guarantees, we assume the agent's actions evolve in continuous time). Our search space was small, however, consisting of 0.05, 0.02, 0.01, and 0.005 seconds.

Trajectory plots. Figure C.1 shows sample trajectories of different methods in the cart-pole domain under adversarial dynamics. The non-robust LQR and model-based planning approaches both diverge and the non-robust PPO doesn't diverge, but doesn't clearly converge after 10 seconds. The robust methods, on the other hand, all clearly converge after 10 seconds.

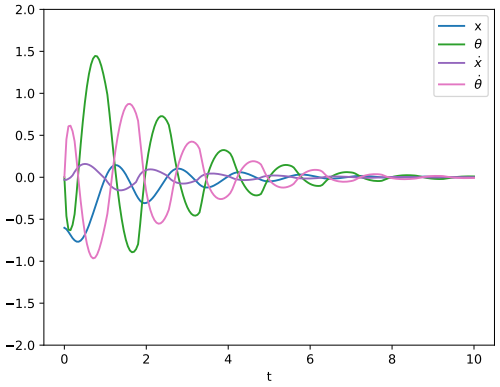
Runtime comparison. Tables C.1 and C.2 show the evaluation and training time of our methods and the baselines over 50 episodes run in parallel. In the NLDI cases where $D = 0$, i.e., Generic NLDI ($D = 0$) and Quadrotor, our projection adds only a very small computational cost. In the other cases, the additional computational cost is more significant, but our method is still far less expensive than the Robust MPC method.

C.9 Experiments for PLDIs and H_∞ control settings

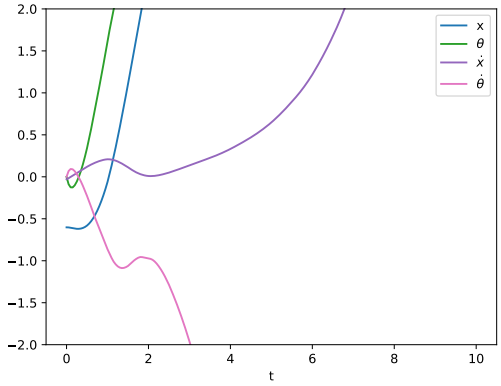
In addition to the NLDI settings explored in the main text, we test the performance of our method on PLDI and H_∞ control settings. As for the experiments in the main text, we choose a time discretization based on the speed at which the system evolves, and run each episode for



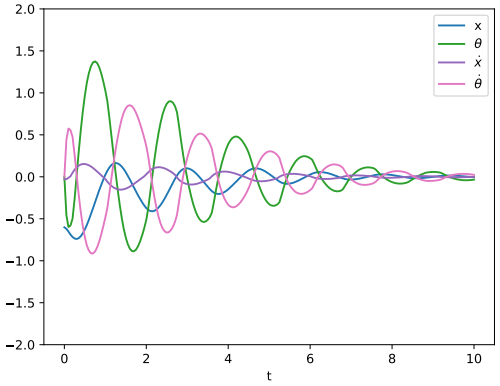
(a) LQR



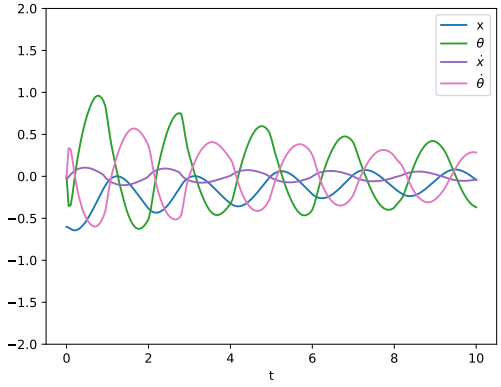
(b) Robust LQR



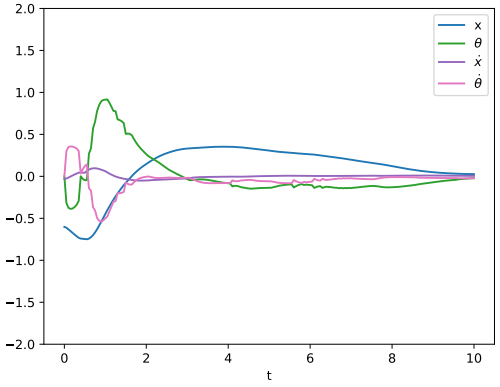
(c) MBP



(d) Robust MBP



(e) PPO



(f) Robust PPO

Figure C.1: Trajectories of 6 different methods on the cart-pole domain under adversarial dynamics.

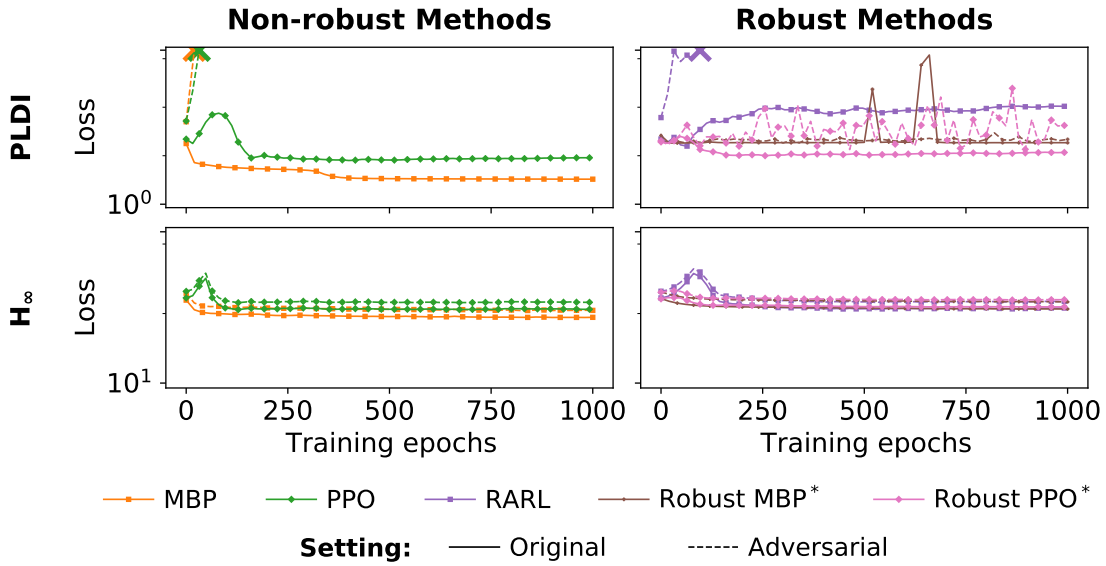


Figure C.2: Representative results for our experimental settings. For each training epoch (10 updates for the MBP model and 18 for PPO), we report average quadratic loss over 50 episodes, and use "X" to indicate cases where the relevant method became unstable. (Lower loss is better.) Our robust methods (denoted by *) improve performance over Robust LQR in the average case, while (unlike the non-robust methods) remaining stable under adversarial dynamics throughout the training process.

200 steps over this discretization. In both cases, we use a randomly generated LQR objective where the matrices $Q^{1/2}$ and $R^{1/2}$ are drawn i.i.d. from a standard normal distribution.

Synthetic PLDI setting. We generate PLDI instances (C.9) with $s = 5$, $a = 3$, and $L = 3$. Specifically, we generate convex hull matrices $(A_1, B_1), \ldots, (A_3, B_3)$ i.i.d. from normal distributions, and generate $(A(t), B(t))$ by using a randomly-initialized neural network with softmax output to weight the convex hull matrices. Episodes were run for 2 seconds at a discretization of 0.01 seconds.

Synthetic H_∞ setting. We generate H_∞ control instances (C.11) with $s = 5$, $a = 3$, and $d = 2$ by generating matrices A, B and G i.i.d. from normal distributions. The disturbance $w(t)$ was produced using a randomly-initialized neural network, with its output scaled to satisfy the $\mathcal{L}_2$ bound on the disturbance. Specifically, we scaled the output of the neural network to satisfy an attenuating norm-bound on the disturbance; at time t, the norm-bound was given by $20 \times f(2 \times t/T)$, where T is the time horizon and f is the standard normal PDF function. Episodes were run for $T = 2$ seconds at a discretization of 0.01 seconds.

Results are given in Figure C.2 and Table C.3.

C.10 Notes on linearization via PLDIs and NLDIs

While we linearize the cart-pole and quadrotor dynamics via NLDIs in our experiments, we note that these dynamics can also be characterized via PLDIs. More generally, in this

Table C.3: Performance of various approaches, both robust (right) and non-robust (left), on domains of interest. We report average quadratic loss over 50 episodes under the original dynamics (O) and under an adversarial disturbance (A). For the original dynamics (O), the best performance for both non-robust methods and robust methods is in bold (lower loss is better). We use "*unstable*" to indicate cases where the relevant method became unstable. Our robust methods (denoted by *) improve performance over Robust LQR in the average case, while remaining stable under adversarial dynamics, whereas the non-robust methods either went unstable or received much larger losses.

Environment		**LQR**	**MBP**	**PPO**	**Robust LQR**	**Robust MPC**	**RARL**	**Robust MBP***	**Robust PPO***
Generic PLDI	O	96.3	**3.3**	8.0	19.2	19.2	15.8	18.6	**10.2**
	A	———	*unstable*	———	43.3	44.1	*unstable*	21.9	16.1
Generic H_∞	O	181	**88**	114	165	N/A	**115**	116	125
	A	219	112	143	206	N/A	145	147	158

section, we show how we can use the framework of PLDIs to model linearization errors arising in the analysis of nonlinear systems.

Consider the nonlinear dynamical system

$$\dot{x} = f(x, u) \text{ with } f(0, 0) = 0. \tag{C.50}$$

for $x \in \mathbb{R}^s$ and $u \in \mathbb{R}^a$. Define $\xi = (x, u)$. We would like to represent the above system as a PLDI in the region $\mathcal{R} := \{\xi \mid \underline{\xi} \leq \xi \leq \bar{\xi}\}$ including the origin. The mean value theorem states that for each component of f, we can write

$$f_i(\xi) = f_i(0) + \nabla f_i(z)^T \xi, \tag{C.51}$$

for some $z = t\xi$, where $t \in [0, 1]$. Now, let $p = s + a$. Defining the Jacobian of f as

$$\mathrm{J}_f(z) = \begin{bmatrix} \nabla f_1(z)^T \\ \vdots \\ \nabla f_p(z)^T \end{bmatrix}, \tag{C.52}$$

and recalling that $f(0) = 0$, we can rewrite (C.51) as

$$f(\xi) = \mathrm{J}_f(z)\xi. \tag{C.53}$$

Now, suppose we can find component-wise bounds on the matrix $\mathrm{J}_f(z)$ over $\mathcal{R}$, i.e,

$$\underline{M} \leq \mathrm{J}_f(z) \leq \bar{M} \text{ for all } z \in \mathcal{R}. \tag{C.54}$$

We can then write

$$\mathrm{J}_f(z) = \sum_{1 \leq i,j \leq p} m_{ij}(t) E_{ij} \quad \text{with} \quad m_{ij}(t) \in [\underline{m}_{ij}, \bar{m}_{ij}], \tag{C.55}$$

where $E_{ij} = e_i e_j^T$ and e_i is the i-th unit vector in $\mathbb{R}^p$.

We now seek to bound the Jacobian using polytopic bounds. To do this, note that we can write

$$\mathrm{J}_f(z) = \sum_{\kappa=1}^{2^{p^2}} \gamma_\kappa A_\kappa \quad \gamma_\kappa \geq 0, \ \sum_\kappa \gamma_\kappa = 1, \tag{C.56}$$

where A_κ's are the vertices of the polytope in (C.55), i.e.,

$$A_\kappa \in \mathcal{V} = \left\{ \sum_{1 \leq i,j \leq p} m_{ij} E_{ij} \mid m_{ij} \in \{\underline{m}_{ij}, \bar{m}_{ij}\} \right\}. \tag{C.57}$$

Together, Equations (C.51), (C.53), (C.56), and (C.57) characterize the original nonlinear dynamics as a PLDI.

We note that this PLDI description is potentially very large; in particular, the size of $\mathcal{V}$ is exponential in the square of the number of non-constant entries in the Jacobian $\mathrm{J}_f(z)$, which could be as large as $2^{p^2} = 2^{(s+a)^2}$. This problem size may therefore become intractable for larger control settings.

We note, however, that we can in fact express this PLDI more concisely as an NLDI. More precisely, we would like to find matrices A, B, C parameterizing the form of NLDI below, which is equivalent to that presented in Equation (7.3) (see Chapter 4 of [Boy+94]):

$$Df(z) \in \{A + B\Delta C \mid \|\Delta\|_2 \leq 1\} \quad \text{for all } z \in \mathcal{R}. \tag{C.58}$$

It can shown that the solution to the SDP

$$\begin{aligned} &\text{minimize} \ \ \mathrm{tr}(V + W) \\ &\text{subject to} \ \ W \succ 0 \\ &\qquad \begin{bmatrix} V & (A_\kappa - A)^T \\ A_\kappa - A & W \end{bmatrix} \succeq 0, \ \forall A_\kappa \in \mathcal{V} \end{aligned} \tag{C.59}$$

yields the matrices A, B, and C with $V = C^T C$ and $W = BB^T$, which can be used to construct NLDI (C.58). While the NLDI here is more concise than the PLDI, the trade-off is that the NLDI norm bounds obtained via this method may be rather loose. As such, for our settings, we obtain NLDI bounds numerically (see Appendices C.4 and C.5), as these are tighter than NLDI specifications obtained via the above method (though they are potentially slightly inexact). An alternative approach would be to examine how to tighten the conversion from PLDIs to NLDIs, which has been explored in other work (e.g. [KRP13]).

Appendix D

Adversarial Robustness for Security-Constrained and Stochastic OPF

D.1 Full SCOPF formulation

In the N-k SCOPF formulation presented in Equation (10.6), for brevity, we abstracted away various device constraints and operational limits into the sets $\mathcal{X}, \mathcal{W}_{\text{base}}$, $\mathcal{Z}_i$, and $\mathcal{W}_i$. Here, we write out those constraints more explicitly to facilitate in-depth descriptions of components of our attack and defense in Appendices D.2 and D.3.

Specifically, let n_{bus} be the number of nodes (*buses*) on the power system, and let n_g be the number of generators. By convention, we designate one of these generators to be a *slack bus* whose voltage angle is fixed at a particular value. Our dispatch $x \in \mathbb{R}^{2n_g-1}$ then consists of the voltage magnitude at the slack bus, and the real power generation and voltage magnitude at all other generators, subject to device limits and operational constraints. As in Section 10.4.1, we let $\mathcal{C}$ represent the set of contingencies, and $z^{(i)} \in \mathbb{R}^{2n_g-1}$ represent slightly adjusted settings of the dispatch quantities that the power system operator can create after scheduling x and then observing some contingency $c^{(i)} \in \mathcal{C}$. We then write the N-k SCOPF problem as

$$\begin{aligned}
&\underset{x \in \mathbb{R}^{2n_g-1}}{\text{minimize}} \;\; f_{\text{base}}(x) + \sum_{(z^{(i)}, c^{(i)})} f_{\text{cont}}(z^{(i)}, c^{(i)}) \\
&\text{subject to} \;\; g_{\text{flow,base}}(x, w_{\text{base}}) = 0, \;\; h_{\text{base}}(x, w_{\text{base}}) \le 0, \;\; w_{\text{base}} \in \mathbb{R}^d \\
&\qquad\qquad z^{(i)} \in \begin{array}{l} \text{argmin}_{z^{(i)} \in \mathbb{R}^{2n_g-1}} f_{\text{cont}}(z^{(i)}, c^{(i)}) \\ \text{s.t. } g_{\text{flow,cont}}(z^{(i)}, w^{(i)}, x) = 0 \\ \qquad h_{\text{cont}}(z^{(i)}, w^{(i)}, x, c^{(i)}) \le 0 \\ \qquad w^{(i)} \in \mathbb{R}^d \end{array} \quad \forall c^{(i)} \in \mathcal{C},
\end{aligned} \tag{D.1}$$

where $h_{\text{base}} : \mathbb{R}^{2n_g-1} \times \mathbb{R}^d \to \mathbb{R}^{2\times(2n_g-1+d)}$ represents device and operational constraints (box constraints) on x and w_{base}, $h_{\text{cont}} : \mathbb{R}^{2n_g-1} \times \mathbb{R}^d \times \mathbb{R}^{2n_g-1} \times \mathcal{C} \to \mathbb{R}^{2\times(2n_g-1+d)}$ represents

device and operational constraints (box constraints) on $z^{(i)}$ and $w^{(i)}$, and all other quantities are as defined in Section 10.4.1.

We can then re-write our N-k SCOPF minimax formulation presented in Section 10.4.2 as

$$\begin{aligned}
&\underset{x\in\mathbb{R}^{2n_g-1}}{\text{minimize}} \max_{y\in\mathcal{Y}} && f_{\text{base}}(x) + f_{\text{cont}}(z,y) + \frac{1}{2}\|s\|_2^2 && \text{(D.2a)}\\
&\text{subject to} && g_{\text{flow,base}}(x, w_{\text{base}}) = 0, \quad h_{\text{base}}(x, w_{\text{base}}) \leq 0, \quad w_{\text{base}} \in \mathbb{R}^d && \text{(D.2b)}\\
&z, s \in && \begin{aligned}&\operatorname{argmin}_{z\in\mathbb{R}^{2n_g-1},\, s\in\mathbb{R}^{2n_{\text{bus}}}} f_{\text{cont}}(z,y) + \tfrac{1}{2}\|s\|_2^2\\ &\text{s. t. } g_{\text{flow,cont}}(z, w_{\text{cont}}, x) + s = 0\\ &\quad h_{\text{cont}}(z, w_{\text{cont}}, x, y) \leq 0\\ &\quad w_{\text{cont}} \in \mathbb{R}^d,\end{aligned} && \text{(D.2c)}
\end{aligned}$$

where $\mathcal{Y} := \{y : y \in [0,1]^{n_o}, \|y\|_1 \leq k\}$ is our outer relaxation to the contingency set, and $s \in \mathbb{R}^{2n_{\text{bus}}}$ are slack variables representing potential infeasibilities in the third-stage optimization problem.

D.2 Further details on the SCOPF attack

The innermost optimization problem in Equation (D.2c), represents the decision a power grid operator would make when faced with a particular partial contingency $y^\star$ after having made a base dispatch $\bar{x}$. In particular, the grid operator is looking to minimize the cost of power generation $f_{\text{cont}}(z, y^\star)$, as well as ensure the system remains feasible ($s = 0$). To solve for the grid response due to a particular partial contingency, we solve (D.2c) using a Newton-based approach [PAP20].

Specifically, let $\lambda \in \mathbb{R}^{2n_{\text{bus}}}$ be the dual variables on the power flow constraint, and $\mu \in \mathbb{R}^{2\times(2n_g-1+d)}$ be the dual variables on the inequality constraints. The Lagrangian of the optimization problem is given by

$$\mathcal{L} = f_{\text{cont}}(z, y^\star) + \frac{1}{2}\|s\|_2^2 + \lambda^T(g_{\text{flow,cont}}(z, w_{\text{cont}}, \bar{x}) + s) + \mu^T h_{\text{cont}}(z, w_{\text{cont}}, \bar{x}, y^\star). \quad \text{(D.3)}$$

The KKT conditions for stationarity, primal feasibility, complementary slackness, and dual feasibility are then given by

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial f_{\text{cont}}(z, y^\star)}{\partial z} + \left(\frac{\partial g_{\text{flow,cont}}(z, w_{\text{cont}}, \bar{x})}{\partial z}\right)^T \lambda + \left(\frac{\partial h_{\text{cont}}(z, w_{\text{cont}}, \bar{x}, y^\star)}{\partial z}\right)^T \mu = 0\\
&\frac{\partial \mathcal{L}}{\partial s} = s + \lambda = 0\\
&g_{\text{flow,cont}}(z, w_{\text{cont}}, \bar{x}) + s = 0\\
&\text{diag}(\mu) h_{\text{cont}}(z, w_{\text{cont}}, \bar{x}, y^\star) = 0\\
&\mu \geq 0.
\end{aligned} \quad \text{(D.4)}$$

The KKT conditions can be written as a set of equations $F_{\text{attack}}(z, s, \lambda, \mu) = 0$. Traditionally in the power systems literature, the equations in (D.4) are solved using a Newton's method [PAP20]. The iterative Newton's method starts with some initial estimates $z^0, s^0, \lambda^0, \mu^0$ for z, s, λ, μ. Then, at each iteration i, we construct a Jacobian J^i for F_{attack} and a corresponding right hand side vector b^i at our current estimate $z^{i-1}, s^{i-1}, \lambda^{i-1}, \mu^{i-1}$. We then solve the resultant set of linear equations

$$J^i \begin{pmatrix} z^i \\ s^i \\ \lambda^i \\ \mu^i \end{pmatrix} = b^i \tag{D.5}$$

to determine the next estimate for the solution, and iterate until convergence.

D.3 Further details on the SCOPF defense

The defense step of our approach entails adjusting our dispatch in a direction of increased robustness to the "worst-case" contingency $y^\star$ found in the attack stage, while also maintaining feasibility in the base case. To do so, we partially solve the minimization problem shown in Equation (10.13), as described in the main text. Specifically, let λ_{base} and λ_{cont} denote the dual variables on the power flow constraints in the base and contingency cases, respectively, and let μ_{base} and μ_{cont} denote the dual variables on the inequality constraints in the base and contingency cases, respectively. Then, the Lagrangian of optimization problem (10.13) is given by

$$\begin{aligned}\mathcal{L} = f_{\text{base}}(x) + f_{\text{cont}}(z, y^\star) + \frac{1}{2}\|s\|_2^2 + \lambda_{\text{base}}^T g_{\text{flow,base}}(x, w_{\text{base}}) \\ + \lambda_{\text{cont}}^T (g_{\text{flow,cont}}(z, w_{\text{cont}}, x) + s) + \mu_{\text{base}}^T h_{\text{base}}(x, w_{\text{base}}) + \mu_{\text{cont}}^T h_{\text{cont}}(z, w_{\text{cont}}, x, y^\star).\end{aligned} \tag{D.6}$$

The KKT conditions associated with this problem are given by

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial f_{\text{base}}(x)}{\partial x} + \left(\frac{\partial g_{\text{flow,base}}(x, w_{\text{base}})}{\partial x}\right)^T \lambda_{\text{base}} + \left(\frac{\partial h_{\text{base}}(x, w_{\text{base}})}{\partial x}\right)^T \mu_{\text{base}} \\ + \left(\frac{\partial g_{\text{flow,cont}}(z, w_{\text{cont}}, x)}{\partial x}\right)^T \lambda_{\text{cont}} + \left(\frac{\partial h_{\text{cont}}(z, w_{\text{cont}}, x, y^\star)}{\partial x}\right)^T \mu_{\text{cont}} = 0\end{aligned} \tag{D.7}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{\text{base}}} = g_{\text{flow,base}}(x, w_{\text{base}}) = 0 \tag{D.8}$$

$$\text{diag}(\mu_{\text{base}}) h_{\text{base}}(x, w_{\text{base}}) = 0 \tag{D.9}$$

$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial f_{\text{cont}}(z, y^\star)}{\partial z} + \cancelto{0}{\left(\frac{\partial g_{\text{flow,cont}}(z, w_{\text{cont}}, x)}{\partial z}\right)^T} \lambda_{\text{cont}} + \cancelto{0}{\left(\frac{\partial h_{\text{cont}}(z, w_{\text{cont}}, x, y^\star)}{\partial z}\right)^T} \mu_{\text{cont}} = 0 \tag{D.10}$$

$$\frac{\partial \mathcal{L}}{\partial s} = s + \lambda_{\text{cont}} = 0 \tag{D.11}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{\text{cont}}} = g_{\text{flow,cont}}(z, w_{\text{cont}}, x) + s = 0 \tag{D.12}$$

$$\text{diag}(\mu_{\text{cont}}) h_{\text{cont}}(z, w_{\text{cont}}, x, y^\star) = 0, \tag{D.13}$$

where a number of the terms in condition (D.10) cancel to zero due to the underlying structure of N-k SCOPF problem. We can group the KKT conditions together based on their relation to the base variables or the contingency variables. We define two vectors of equations that group the KKT conditions together: $F_{\text{base}}(x, w_{\text{base}}, \lambda_{\text{base}}, \mu_{\text{base}})$, representing the decoupled defense equations in blue above, and $F_{\text{cont}}(z, w_{\text{cont}}, \lambda_{\text{cont}}, \mu_{\text{cont}})$, representing the decoupled contingency optimization equations in red. We can then group the KKT conditions as follows:

$$\begin{aligned}
&\begin{pmatrix} \frac{\partial \mathcal{L}}{\partial x} \\ \frac{\partial \mathcal{L}}{\partial \lambda_{\text{base}}} \\ \text{diag}(\mu_{\text{base}}) h_{\text{base}}(x, w_{\text{base}}) \end{pmatrix} \\
&\equiv F_{\text{base}}(x, w_{\text{base}}, \lambda_{\text{base}}, \mu_{\text{base}}) + \begin{pmatrix} \left(\frac{\partial g_{\text{flow,cont}}(z, w_{\text{cont}}, x)}{\partial x}\right)^T \lambda_{\text{cont}} + \left(\frac{\partial h_{\text{cont}}(z, w_{\text{cont}}, x, y^\star)}{\partial x}\right)^T \mu_{\text{cont}} \\ 0 \\ 0 \end{pmatrix} = 0,
\end{aligned} \tag{D.14}$$

$$\begin{aligned}
&\begin{pmatrix} \frac{\partial \mathcal{L}}{\partial z} \\ \frac{\partial \mathcal{L}}{\partial s} \\ \frac{\partial \mathcal{L}}{\partial \lambda_{\text{cont}}} \\ \text{diag}(\mu_{\text{cont}}) h_{\text{cont}}(z, w_{\text{cont}}, x, y^\star) \end{pmatrix} \\
&\equiv F_{\text{cont}}(z, w_{\text{cont}}, \lambda_{\text{cont}}, \mu_{\text{cont}}) + \begin{pmatrix} 0 \\ 0 \\ g_{\text{flow,cont}}(z, w_{\text{cont}}, x) \\ \text{diag}(\mu_{\text{cont}}) h_{\text{cont}}(z, w_{\text{cont}}, x, y^\star) \end{pmatrix} = 0.
\end{aligned} \tag{D.15}$$

We notice the two KKT terms F_{base} and F_{cont} are independent but are coupled through two additional terms. For the N-k SCOPF application, these coupling terms represent the generator's contingency ramping constraints and the voltage set points from the base case. Due to the sparse nature of the grid, these couplings are weak, which makes these equations well-suited to solve using a decoupled Gauss-Seidel approach.

The non-linear Gauss-Seidel is an iterative method to solve the two sets of weakly coupled equations independently using the values of the coupled variables from the previous

iteration. Based on the KKT conditions above, we can write the Gauss-Siedel equations at iteration i as follows:

$$F_{\text{cont}}(z^i, w^i_{\text{cont}}, \lambda^i_{\text{cont}}) + \begin{pmatrix} 0 \\ 0 \\ g_{\text{flow,cont}}(z^i, w^i_{\text{cont}}, x^{i-1}) \\ \text{diag}(\mu^i_{\text{cont}}) h_{\text{cont}}(z^i, w^i_{\text{cont}}, x^{i-1}, y^\star) \end{pmatrix} = 0, \tag{D.16}$$

$$F_{\text{base}}(x^i, w^i_{\text{base}}, \lambda^i_{\text{base}}, \mu^i_{\text{base}}) + \begin{pmatrix} \frac{\partial g_{\text{flow,cont}}(z^i, w^i_{\text{cont}}, y^\star)}{\partial x}^T \lambda^i_{\text{cont}} + \left(\frac{\partial h_{\text{cont}}(z^i, w^i_{\text{cont}}, x^i, y^\star)}{\partial x} \right)^T \mu^i_{\text{cont}} \\ 0 \\ 0 \end{pmatrix} = 0. \tag{D.17}$$

By specifically ordering the Gauss-Seidel algorithm to first solve (D.16) and then (D.17), we can pass the updated contingency-specific coupling variables z^i and λ^i_{cont} obtained by solving (D.16) at the current iterate i to the solve of (D.17). The Gauss-Seidel algorithm for this specific application relaxes the coupling by, at iteration i, using the value of x from the previous iteration, as highlighted in green in (D.16). By ordering the updates in this way, we initiate the Gauss-Seidel iterations using the final solution we already obtained when solving for the worst-case attack $y^\star$ and its associated $z^\star$, which is the solution to (10.14b).

Rather than running the Gauss-Seidel iterations to convergence, we run only a single iteration in order to take a step in the dispatch x (before then restarting the attack phase). We can efficiently solve this single Gauss-Seidel since we reuse the solution from the attack stage for Equation (D.16), and therefore all that remains to solve is Equation (D.17).

D.4 GO competition scoring

Challenge 1 of the GO competition [ARP19] created a specific formulation for the grid constraints in Equation (D.2c). In particular, they relaxed part of the formulation to not allow certain grid models such as *switched shunts*. They also used *automatic generation control* modulate the power generation at each generator to ensure the power grid frequency remained at its nominal set point. However, in contrast, our "third stage" power flow solver is built to solve discrete shunts and discrete *transformer tap controls*. We also do not use automatic generation control, but instead enable generators to *ramp* (i.e., change their production) within some limits determined by the base dispatch. These grid models are most consistent with Challenge 2 of the GO competition [ARP19], which updated its grid models from the first competition (and which was ongoing at the time the present work was being done). The full list of relevant differences in grid models is shown in the table below.

	GO Challenge 1 models	CAN∂Y models
Power generation adjustments	Automatic generation control	Generator ramping
Discrete shunts	Not allowed	Allowed
Transformer tap ratios	Fixed	Adjustable (discrete)

As scores and solutions for GO Challenge 2 are not yet available, we compare our solver against solvers from GO Challenge 1, despite the fact that our solver uses grid models from GO Challenge 2. In particular, we score our formulation by using the score weighting criteria given in GO Challenge 1 [ARP19] and compare the against the scores of the top Challenge 1 competitors, which are available at [ARP19]. However, since the grid models used by our solver vs. the Challenge 1 solvers are different, the score comparisons we show in Section 10.5.2 are ultimately approximate. Nonetheless, we believe that the comparisons still indicate that our methodology is competitive for N-1 SCOPF.