Learning and Optimization for Efficient and Optimal Operations in Sustainable Power Systems

Ling Zhang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2024

Reading Committee:

Baosen Zhang, Chair

Daniel Kirschen

Sam Burden

Program Authorized to Offer Degree:
Electrical & Computer Engineering

University of Washington

Abstract

Learning and Optimization for Efficient and Optimal Operations in Sustainable Power Systems

Ling Zhang

Chair of the Supervisory Committee:
Baosen Zhang
Department of Electrical & Computer Engineering

Sustainable energy systems have the potential to significantly reduce climate change and improve economic and social welfare. However, transitioning to renewable energy resources has brought uncertainties into all facets of the decision-making processes across the energy system. For example, the optimal power flow (OPF), a foundational resource allocation problem in power systems, has now to be solved repeatedly for numerous scenarios within tight timeframes to adjust to rapid fluctuations in electricity demands. While machine learning (ML) has emerged as a promising approach to accelerating the computation of optimization problems, standard ML algorithms face challenges in enforcing the constraints of physical systems and providing generalization guarantees. These challenges underscore the necessity for a nuanced design of ML algorithms to ensure efficient and optimal operations in energy systems, especially when faced with high penetration of renewables and significant uncertainties. In this dissertation, we propose machine learning-based algorithms for efficient and optimal operations of power systems under large uncertainty.

More specifically, we develop a flow-based generative approach to model residential load behaviors and generate varied and plentiful future scenarios for residential load demand. Then, to optimize and plan power systems across these diverse scenarios, we design neural network-based solvers to offer solutions orders of magnitude faster than conventional methods. By integrating problem-specific structural insights, we ensure learned solutions satisfy

engineering and operational constraints within power systems. These insights also enhance the data efficiency and generalization capabilities of the learning algorithms. In addition, we apply the proposed algorithmic designs to the fundamental resource allocation problem in power system operations, i.e., optimal power flow, to showcase the effectiveness of these methods. The specific contributions of this dissertation include (i) a flow-based generative approach is proposed to model residential load demand and generate varied load scenarios; (ii) a convex neural network-based algorithm is developed for solving the DC Optimal Power Flow (DCOPF) problem, providing generalization guarantees; (iii) an unsupervised neural network-based algorithm is introduced for solving the two-stage DCOPF problem, ensuring feasibility guarantees; (iv) a Lagrangian-based iterative algorithm is proposed to enhance the solution quality of the AC Optimal Power Flow (ACOPF) by iteratively refining the initial point for local solvers.; (v) using insights from the Lagrangian function, a neural network-based learning algorithm is developed to provide high-quality warm starts for solving the ACOPF; (vi) an analytical method is presented to construct the convex restriction of the feasible set for AC power flows in radial networks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF NOTATIONS

Sets and parameters

$\mathcal{N} = \{1, 2, \cdots, n\}$: Set of buses.

$\mathcal{E} = \{1, 2, \cdots, m\}$: Set of transmission lines.

ℓ_i: Load at bus i (no time considered).

ℓ_t: Load at time t (aggregated).

c_i: Unit generation cost of generator i.

γ_i: Unit cost of scheduled reserve capacity at bus i in the first stage.

q_i^{res}: Unit cost of exceeding scheduled reserve capacity at bus i in the second stage.

g_{ij}: Conductance of the line connecting buses i and j.

b_{ij}: Susceptance of the line connecting buses i and j.

$\overline{V}_i$: Maximum voltage magnitude allowed at bus i.

$\underline{V}_i$: Minimum voltage magnitude allowed at bus i.

$\overline{\delta}_{ij}$: Maximum voltage angle difference allowed between buses i and j.

$\underline{\delta}_{ij}$: Minimum voltage angle difference allowed between buses i and j.

$\overline{p}_i^g$: Maximum active power generation at bus i.

$\overline{q}_i^g$: Maximum reactive power generation at bus i.

$\underline{p}_i^g$: Minimum active power generation at bus i.

$\underline{q}_i^g$: Minimum reactive power generation at bus i.

$\overline{p}_{ij}^f$: Maximum allowed active power flow along the line connecting buses i and j.

$\underline{p}_{ij}^f$: Minimum allowed active power flow along the line connecting buses i and j.

$\overline{q}_{ij}^f$: Maximum allowed reactive power flow along the line connecting buses i and j.

$\underline{q}_{ij}^f$: Minimum allowed reactive power flow along the line connecting buses i and j.

$\mathbf{B} \in \mathbb{R}^{n \times (n-1)}$: Matrix transforming non-slack bus voltage angles to nodal power injec-

tions.

$\mathbf{F} \in \mathbb{R}^{m\times(n-1)}$: Matrix transforming non-slack bus voltage angles to power flows.

$\mathbf{E} \in \mathbb{R}^{m\times(n-1)}$: Matrix transforming non-slack bus voltage angles to voltage angle differences.

$\mathbf{M}^F \in \mathbb{R}^{m\times(n-1)}$: Matrix transforming fundamental flows to all line flows.

$\mathbf{M}^B \in \mathbb{R}^{n\times(n-1)}$: Matrix transforming fundamental flows to nodal power injections.

Decision variables

V_i: Voltage magnitude at bus i.

δ_i: Voltage angle at bus i.

p_i^g: Active power generation at bus i.

q_i^g: Reactive power generation at bus i.

p_i^d: Active power demand at bus i.

q_i^d: Reactive power demand at bus i.

p_i^{inj}: Active nodal power injection at bus i.

q_i^{inj}: Reactive nodal power injection at bus i.

p_{ij}^f: Active power flow along the line connecting buses i and j.

q_{ij}^f: Reactive power flow along the line connecting buses i and j.

p_i^I: Scheduled power at bus i in the first stage.

p_i^R: Recourse power at bus i in the second stage.

$\widehat{r}_i$: Scheduled up reserve capacity at bus i in the first stage.

$\breve{r}_i$: Scheduled down reserve capacity at bus i in the first stage.

$\boldsymbol{\theta}$: Trainable parameters in the neural network.

Functions

$L(\cdot;\boldsymbol{\theta})$: Loss function for training the neural network.

$f_{\boldsymbol{\theta}}(\cdot)$: Functional representation of the neural network.

$\mathcal{L}(\cdot)$: Lagrangian function.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Baosen Zhang, for his guidance and patience over the past six years. Working with him has been the greatest privilege of my life. Without his support and encouragement, this dissertation would not have been possible. I cannot adequately describe how much I have learned from him, both academically and personally. His passion for research, wisdom, and compassionate care for his students will always inspire me to work hard and strive to be like him.

My appreciation also goes to my committee members. I want to thank Prof. Kirschen, Prof. Burden, and Prof. Zhao for their valuable comments. Their expertise and insightful suggestions have greatly improved my dissertation.

The Department of Electrical & Computer Engineering at the University of Washington has been a wonderful home to me. I appreciate all the help and happiness I found here. I am grateful for the tremendous opportunities to learn from so many excellent teachers. Many thanks to my amazing graduate student colleagues. Their intelligence and support have made my Ph.D. journey memorable.

Finally, I would like to thank my parents and brother for always believing in me, being there for me, and encouraging me throughout my doctoral journey.

DEDICATION

To my parents and brother.

Chapter 1
INTRODUCTION

The increasing penetration of intermittent renewable resources, responsive loads, electric cars, and other new technologies have made it more and more challenging to build accurate models for power grid optimization and control. This leads to the popular use of machine learning in power system applications because of the universal approximation capabilities of neural networks. However, despite many advances in this area, most existing machine learning approaches are data-driven and do not consider the underlying physics in power systems. As a result, these approaches often require a large number of high-quality training data and complicated neural network structures. There are also serious concerns that machine learning results are very likely to violate critical physical constraints. All these make machine learning-based approaches less appealing when applied to power systems.

In this dissertation, we develop physics-informed machine learning approaches for power system planning and optimal operation, including scenario forecasting of residential load and faster optimal power flow solvers. By leveraging the unique structure of these problems, we incorporate physics-based information into the design of neural networks to improve data efficiency and ensure that predictions from the neural networks satisfy critical physical constraints.

In Chapter 2, we focus on generating scenario forecasts of future residential load by conditioning on historical realizations. Instead of utilizing point forecasts, we resort to (deep) generative models, flow-based models, to generate new scenarios with variations by learning the true distribution of the future load. Flow-based models are not only eligible for conditioning on continuous-valued vectors but also easy-to-use. It is also worth noting that flow-based models explicitly model the data distribution and maximize the value of the probability density function (PDF) for the modeled distribution. By employing a series of specially-designed reversible transformations, flow-based models are able to calculate the

PDF value very efficiently. Our experiment results also show that flow-based generative models achieve significantly better performance in generating high-quality residential load scenario forecasts given the past observations.

In Chapter 3, we propose a two-step approach for solving DCOPF. Firstly, a neural network is used to learn the *value* (i.e., the optimal cost) of the DCOPF, and its gradient with respect to the load is the locational marginal prices (LMPs). Then the binding constraints are identified based on the LMPs. This process is robust in the sense that if they are approximately correct, then the binding constraints are correctly identified. We provide a formal guarantee to the generalization capability of this method by directly designing the fundamental features of the DCOPF problem into the machine learning algorithm. Specifically, we constrain the neural network architecture to be convex and build KKT conditions into the training process. First, by utilizing the convexity of DCOPF problem, we train an input convex neural network. Second, we construct the training loss based on KKT optimality conditions. By combining these two techniques, the trained model has provable generalization properties, where small training error implies small testing errors. In experiments, our algorithm significantly outperforms other machine learning methods.

In Chapter 4, we overcome the challenge in policy design and solving two-stage DCOPF problems by presenting a neural network (NN)-based architecture that is computationally efficient and also guarantees the feasibility of learned solutions. In particular, our architecture involves two neural networks, one each for the first and second stages. The first neural network learns the mapping from the load forecast to the first-stage decisions. The second neural network approximates the cost-to-go given the net-load realization and the learned first-stage decisions. So, instead of using the affine policy, we offer an NN-based policy to solve the second-stage OPF problem. A technique called the gauge map is incorporated into the learning architecture design to guarantee the learned solutions' feasibility to the network constraints. Namely, we can design policies that are feed forward functions and only output feasible solutions. Simulation results on standard IEEE systems show that, compared to iterative solvers and the widely used affine policy, our proposed method not only learns solutions of good quality but also accelerates the computation by orders of magnitude.

In Chapter 5, we propose a simple algorithm that can effectively escape from strict local

solutions to find better ones. By moving from one solution to another while reducing the cost, we can successively move towards the globally optimal solution. In contrast to algorithms that are launched repeatedly with random initializations, our proposed algorithm is deterministic. And it relies on duality theory to provide better warm starts to existing solvers. First, we solve the ACOPF problem using some solver (e.g., IPOPT [1] or Matpower [2]). From the solution and its associated dual variables, we form a partial Lagrangian by dualizing the power balance equations. We then optimize this partial Lagrangian, which leads to a different solution. Using this second solution as a warm start, we again call the solver for the ACOPF problem. We show that our algorithm can quickly escape from local solutions and find lower cost solutions. This feature holds even for ACOPF problems with disconnected feasible spaces, for example, the 2-bus network shown in Section 5.4.1, which has been traditionally difficult to deal with [3, 4]. For networks with known global solutions (3, 9, 22, 118, 300-bus), we show that our algorithm can find the globally optimal solution in a single iteration, even starting from a strict local solution.

In Chapter 6, we present a machine learning architecture that overcomes the challenge of multiple suboptimal local solutions in the training data set. We first learn a neural network that maps load to the dual variable of the power balance constraints. These dual variables are the locational marginal prices and would be readily available from any modern nonlinear solver. These dual variables are used to form a partial Lagrangian, whose solution we also learn via a neural network. Then we use the predicted solution of the partial Lagrangian as a warm start [5]. Interestingly, this warm starting point tends to be closer to the globally optimal solution of the ACOPF, even if the training data set only has suboptimal solutions or a mixture of global and local solutions. Therefore, by using the learned warm start as an initialization point, we could have better solution quality than directly learning based on load/solution pairs.

In Chapter 7, we focus on convex restrictions of OPF feasible sets in radial networks. We propose to construct the convex restriction in a transformed coordinate space of voltage phase angles. Specifically, we apply a change of variables such that the active and reactive power equations become naturally convex after variable change, hence eliminating the need to approximate the bus power upper bound constraints. The lower bound constraints can

be approximated using the first-order Taylor approximation, which is the best (the least conservative) upper bound one could have for concave functions. We use a 3-bus line network as an example (Figure 7.1) to show that the convex restriction constructed in this way can be a maximal convex subset, meaning it cannot be contained within any other convex subset. As different tangent points can produce different first-order Taylor approximations, thus leading to different convex restricted sets, we introduce an iterative algorithm that progressively refines the locations of tangent points to find the optimal convex restriction, one that contains the optimal solution of the original problem. We test our method on the IEEE 123-bus distribution network with three types of objective functions applied: power loss minimization, generation cost minimization, and state estimation. The simulation results show that the iterative algorithm that builds on the proposed convex restriction method always finds a good feasible solution within at most 10 iterations, even when the traditional methods failed.

Finally, in Chapter 8, we conclude this dissertation and discuss future research directions.

Chapter 2

SCENARIO FORECASTING OF RESIDENTIAL LOAD PROFILES

2.1 Introduction

Distribution systems are becoming more dynamic and more decentralized because of the emergence of new technologies and services. Instead of operating distribution networks as passive systems, utilities start to account for distributed energy resources such as rooftop solar, electric vehicles and demand response programs. Since these resources are stochastic and intermittent, accurate forecasting of residential load becomes important for operators to decide how to integrate distributed energy resources and where to deploy energy storage so as to match customers' demand and make better use of energy [6]. Furthermore, as the energy system is transformed into a more distributed architecture, forecasting residential load for a single or a small number of households becomes necessary for many services and applications. For example, demand response programs require forecasting the residential load of each participating household to make personalized adjustment in the demand and also allow customers to manage individual costs such as peak demand charges [7, 8]. Other applications include stochastic energy management in distribution grids, where forecasting in much smaller aggregation scales-individuals or small groups of households-is considered for voltage stability issues because of the use of plug-in hybrid electric vehicles (PHEVs) [9, 10, 11].

Compared to standard load forecasting used in transmission system operations, residential level forecasting have received less attention until relatively recently. For introductions and surveys on this topic, the readers can refer to [12, 13, 14] and the references within. Despite these advances, residential load forecasting, especially for a single or a small number of households, remains a challenging problem for two reasons. Firstly, individual load naturally exhibits higher volatility compared to a larger aggregation of loads because of the randomness of human behaviors and smaller base loads [15, 16, 17]. This makes achieving

very accurate point forecasts fundamentally difficult and the standard metric of the distance between forecasted and realized values becomes less useful as a figure of merit [18]. Secondly, the increasing deployment of distributed solar, the popularity of PHEVs and the emerging trend of behind-the-meter energy storage bring even more uncertainties to the electricity demand of users. Therefore, methodologies should capture and reflect these uncertainties in the forecasting process.

An important method in load forecasting used to describe the future uncertainties associated with a load is *scenario forecasting* [19]. Different than the conventional deterministic point forecasting approach [20, 21, 22], which generates the most likely forecast for the future load as a single estimate, scenario forecasting provides a wide range of possible realizations of the future that can occur. Scenario forecasting can be more useful compared to probabilistic forecasts by not only informing operators of the uncertainties about the future in the form of prediction intervals or quantile forecasts, but also generating a set of plausible time series for early planning [18]. This also opens more possibilities of generating realistic residential load profiles to compensate for the lack of measured residential load datasets. These datasets are difficult to collect because of changes in occupancy and spotty deployment of smart meters [23, 24]. As a result of insufficient measured data, average load profiles are commonly used in research studies, which may lead to misleading results due to a lack of diversity [25]. In these settings, scenario forecasting provides a promising method for generating artificial residential load profiles that have similar properties to measured data and hence can be used in downstream tasks in power systems.

Previous works on generating scenario forecasts for residential load fall into two main categories. The first category is to make use of the point load forecasts coming from the pre-trained models and then add noise to them [26]. Specifically, the residuals of the point forecasts are modeled with a normal distribution, which is then added back to the original point load forecasts to generate a set of possible scenarios. The other category of methods take advantage of the relationship between the weather and the load, and generate probabilistic forecasts of load based on simulated weather scenarios [27, 28, 29, 30]. For example, a group of weather scenarios are created based on the historical data, and then each generated weather scenario is fed into the point load forecasting model to obtain a different

set of point forecasts for the load. The former category suffers from the fact it creates scenarios centered at the point forecast, which may not capture the diversity in load behaviors, especially if there are multiple modes in the data. The latter category can generate more diverse scenarios, but does not overcome the fundamental issues since it pushes the question to that of how to select a set of good weather profiles. More fundamentally, both methods are based on point forecasts, which are designed to be the deterministic solutions that minimize a distance metric. However, the goal of scenario forecasting is different, being that we want to generate i.i.d. samples of possible future load realizations.

Recently, generative models based on (deep) neural networks have been applied to scenario forecasts generation to overcome the challenges in more traditional methods. The works in [31, 32] use the Generative Adversarial Network (GAN) [33] to generate scenarios for renewable resources (i.e., wind and solar). This method is then extended by [34] to generate scenario forecasts for residential load. Particularly, the work in [34] built the GAN model based on the Auxiliary Classifier GAN (ACGAN) [35] to generate load profiles with specific load patterns. It is worth noting that these generative models are not really forecasting models, since they can only include discrete-valued conditional information (e.g., winter vs. summer days). However, in most practical forecasting applications, the side information to be conditioned on is typically continuous-valued, like the past observations of the residential load.

In this chapter, we focus on generating scenario forecasts of future residential load by conditioning on historical realizations. Instead of utilizing point forecasts, we resort to (deep) generative models, such as GANs and flow-based models, to generate new scenarios with variations by learning the true distribution of the future load. It is interesting to note that the performance of the conditional GAN (CGAN) [36] is not satisfactory for this task because the conditional information is continuous and vector-valued, and the training of GAN models is notoriously unstable because of its two constantly competing components-the generator and the discriminator [37]. In contrast, flow-based models are not only eligible for conditioning on continuous-valued vectors but also easy-to-use. It is also worth noting that, unlike GANs, flow-based models explicitly model the data distribution and maximize the value of the probability density function (PDF) for the modeled distribution. By em-

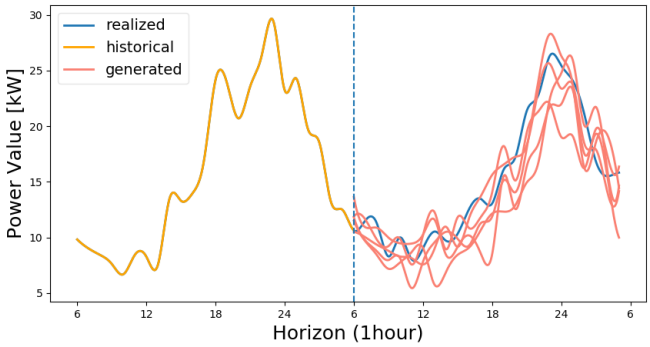


Figure 2.1: Realized residential load versus a group of generated scenario forecasts using our methods. Red curves are generated scenarios, the blue curve is the realized load data and the black curve is the historical load data. Generated scenario forecasts exhibit both accuracy and diversity.

ploying a series of specially-designed reversible transformations, flow-based models are able to calculate the PDF value very efficiently. Our experiment results also show that flow-based generative models achieve significantly better performance in generating high-quality residential load scenario forecasts given the past observations. We show examples of generated daily scenarios using the flow-based model in Fig. 2.1. For these reasons, we adopt flow-based generative models [38, 39, 40] for conditional scenario forecasting in this chapter.

The rest of this chapter is organized as follows: Section 2.2 introduces the flow-based generative models; Section 2.3.2 extends the flow-based models to conditional generative models, and employ the flow-based conditional generative models to conditional scenario forecasting for residential load; simulation results are illustrated and evaluated in Section 2.4.5; and Section 2.5 concludes this chapter.

2.2 Flow-Based Generative Models

In this section, we introduce the flow-based generative models [38, 40, 39]. Unlike the other types of generative models, such as Generative Adversarial Networks (GANs) [33] and Variational Encoders (VAEs) [41], whose training objectives either avoid constructing the PDF of data or utilize a lower bound instead, flow-based generative models are trained

to directly maximize the value of the modeled PDF. We first review the change-of-variable technique and use this technique to formulate the objective function for training flow-based models, then we talk about the architectures adopted in flow-based models that enable the efficient computation of the training objective.

Consider a $D-$dimensional data variable X, with $\mathbf{x}$ as its realization. Denote the true value of the PDF of X at the point $\mathbf{x}$ by $p_X(\mathbf{x})$, and we train a flow-based generative model to estimate this value. Specifically, we first draw a $D-$dimensional latent variable $\mathbf{z}$ from a simple prior distribution p_Z and provide it as the input to the flow-based model. Suppose there exists a bijective mapping $f : \mathbb{R}^D \longrightarrow \mathbb{R}$ such that $f(\mathbf{x}) = \mathbf{z}$ and $f^{-1}(\mathbf{z}) = \mathbf{x}$. Then, according to the change-of-variable formula, the density function of X at the given point $\mathbf{x}$ can be represented by [39]

$$p_X(\mathbf{x}) = p_Z(f(\mathbf{x}))|\det(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^T})| \tag{2.1}$$

$$\log p_X(\mathbf{x}) = \log p_Z(f(\mathbf{x})) + \log|\det(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^T})|. \tag{2.2}$$

Typically, the density function p_Z is chosen to be standard multivariate Gaussian, i.e., $\mathcal{N}(0, \mathbf{I})$, and $\det(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^T})$ is the determinant of the Jacobian matrix of f at $\mathbf{x}$. Since the ground truth bijective mapping f that can map the distribution p_Z to the true PDF of X is unknown, we implement a parameterized bijective function f_θ that can be learned by training the flow-based model. From the change-of-variable formula in (2.1) (or (2.2)), the modeled PDF of X under the mapping of f_θ is given by

$$p_X(\mathbf{x};\theta) = p_Z(f_\theta(\mathbf{x}))|\det(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}^T})| \tag{2.3}$$

$$\log p_X(\mathbf{x};\theta) = \log p_Z(f_\theta(\mathbf{x})) + \log|\det(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}^T})|, \tag{2.4}$$

where the modeled PDF of X, $p_X(\mathbf{x};\theta)$, can be considered as a function of the parameter θ, which is called the likelihood function, and the log of it is called the log-likelihood function. Using the maximum likelihood estimation (MLE) method, we can train the flow-based model to choose the appropriate value of θ such that the likelihood in (2.3) or the log-likelihood in (2.4) is maximized:

$$\max_\theta \log p_X(\mathbf{x};\theta) = \log p_Z(f_\theta(\mathbf{x})) + \log|\det(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}^T})|. \tag{2.5}$$

In flow-based models, the parameterized bijective function f_θ is chosen to be the composition of a sequence of reversible transformations, that is, $f_\theta = f_1 \circ f_2 \circ \cdots \circ f_K$, such that the mapping from $\mathbf{x}$ to $\mathbf{z}$ and the inverse mapping from $\mathbf{z}$ to $\mathbf{x}$ can be represented as follows:

$$\mathbf{x} \xrightarrow{f_1} \mathbf{x}_1 \xrightarrow{f_2} \mathbf{x}_2 \cdots \mathbf{x}_{K-1} \xrightarrow{f_K} \mathbf{x}_K = \mathbf{z} \tag{2.6}$$

$$\mathbf{z} = \mathbf{x}_K \xrightarrow{f_K^{-1}} \mathbf{x}_{K-1} \xrightarrow{f_{K-1}^{-1}} \mathbf{x}_{K-2} \cdots \mathbf{x}_1 \xrightarrow{f_1^{-1}} \mathbf{x}. \tag{2.7}$$

The sequence of reversible transformations in (2.6) is called a normalizing flow [40]. Based on the design of the normalizing flow in (2.6), the log-determinant of the Jacobian matrix $\frac{\partial f_\theta(x)}{\partial x^T}$ can be written as follows by using the chain rule:

$$\log|\det(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}^T})| = \sum_{i=1}^{K} \log|\det(\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}^T})| \tag{2.8}$$

where $\mathbf{x}_0 = \mathbf{x}$, and $\mathbf{x}_K = \mathbf{z}$. To facilitate the computation of the log-determinants of Jacobian in (2.8), each reversible transformation f_i in (2.6) and (2.7) is implemented as an affine coupling layer. Take the affine coupling layer design in Real-valued Non-Volume Preserving (RealNVP) model [39] as an example. Given a $D-$dimensional input $\mathbf{x}$, we can split it into two parts, $\mathbf{x}_{1:d}$ and $\mathbf{x}_{d+1:D}$, where $d < D$. Then the output $\mathbf{y}$ of an affine coupling layer is given by

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d} \tag{2.9}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + \tau(\mathbf{x}_{1:d}) \tag{2.10}$$

where $\odot$ represents element-wise product, and $s(\cdot)$ and $\tau(\cdot)$ are scaling and translating functions, respectively. [1] Following the transformations in (2.9) and (2.10), the Jacobian matrix of $\mathbf{y}$ at $\mathbf{x}$ is a lower-triangular matrix

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}^T} = \begin{bmatrix} \mathbf{I}_d & 0 \\ * & \text{diag}(\exp(s(\mathbf{x}_{1:d}))) \end{bmatrix} \tag{2.11}$$

[1] Note that the other flow-based generative model, Non-linear Independent Component Estimation (NICE) model [38], uses a similar affine coupling layer structure as in (2.9) and (2.10) but without scaling, and the latest generative flow (Glow) model [40] adopts the same affine coupling layer as RealNVP.

and its log-determinant is simply $\text{sum}(s(\mathbf{x}_{1:d}))$. Note that, in (2.11), the operation $\text{diag}(\cdot)$ constructs a diagonal matrix from a vector, and the value of $*$ has no impact on the log-determinant of this Jacobian.

In order to get the series of transformations in (2.6), multiple coupling layers like (2.9) and (2.10) are combined in an alternating way to construct a normalizing flow[39]. As a result, the log-determinant of the Jacobian matrix $\frac{\partial f_\theta(x)}{\partial x^T}$ in (2.4) is just a sum of lower-triangular matrices' log-determinants, which makes the efficient computation of the training objective in (2.5) possible.

2.3 Conditional Scenario Generation

In this section, we first show flow-based generative models can be extended to conditional generative models by providing some side information $\mathbf{c}$ as the conditioning input in the training process. Particularly, different than previous conditional generative models which typically condition on discrete-valued categorical labels [36], flow-based conditional generative models can condition on continuous-valued data, such as time-series observations. Aside from the basic structure of flow-based conditional generative models, we also provide a new structure design for the transformations in the normalizing flow in order to capture as much information as possible from the conditioning input. At the end of this section, we describe how to employ flow-based conditional generative models to scenario forecasting for residential load by considering the historical observations.

2.3.1 Conditional Flows

Considering the data sample $\mathbf{x} \in \mathbb{R}^D$ and the associated side information $\mathbf{x}^c \in \mathbb{R}^{D'}$, we train a flow-based generative model to estimate the value of the conditional PDF of X at the point $\mathbf{x}$ given $\mathbf{x}^c$, i.e., $p_{X|X^c}(\mathbf{x}|\mathbf{x}^c)$. Specifically, we first draw a latent variable $\mathbf{z}$ from distribution p_Z. Then we construct a parameterized bijective function f_θ that takes $\mathbf{x}^c$ as an extra input such that $f_\theta(\mathbf{x};\mathbf{x}^c) = \mathbf{z}$ and $f_\theta^{-1}(\mathbf{z};\mathbf{x}^c) = \mathbf{x}$. Using the change-of-variable formula in (2.1) (or (2.2)), the modeled conditional PDF of X under the mapping of f_θ can

be written as

$$p_{X|X^c}(\mathbf{x}|\mathbf{x}^c) = p_Z(f_\theta(\mathbf{x};\mathbf{x}^c))|\det(\frac{\partial f_\theta(\mathbf{x};\mathbf{x}^c)}{\partial \mathbf{x}^T})| \tag{2.12}$$

$$\log p_{X|X^c}(\mathbf{x}|\mathbf{x}^c) = \log p_Z(f_\theta(\mathbf{x};\mathbf{x}^c)) + \log|\det(\frac{\partial f_\theta(\mathbf{x};\mathbf{x}^c)}{\partial \mathbf{x}^T})|. \tag{2.13}$$

Using the MLE method, we train the flow-based model to maximize the conditional likelihood of X in (2.12) or the conditional log-likelihood of X in (2.13):

$$\max_\theta \log p_{X|X^c}(\mathbf{x}|\mathbf{x}^c;\theta) = \log p_Z(f_\theta(\mathbf{x};\mathbf{x}^c)) + \log|\det(\frac{\partial f_\theta(\mathbf{x};\mathbf{x}^c)}{\partial \mathbf{x}^T})|. \tag{2.14}$$

Suppose we collect N independent identically distributed (i.i.d.) samples $(\mathbf{x}^1,\mathbf{x}^{c,1}),\cdots,(\mathbf{x}^N,\mathbf{x}^{c,N})$ from the ground-truth conditional distribution $p_{X|X^c}(\mathbf{x}|\mathbf{x}^c)$, following the objective function in (2.14), we can train the flow-based model on this dataset through the following optimization:

$$\begin{aligned}\max_\theta \sum_{i=1}^N \log p_{X|X^c}(\mathbf{x}^i|\mathbf{x}^{c,i}) = &\sum_{i=1}^N \log p_Z(f_\theta(\mathbf{x}^i;\mathbf{x}^{c,i})) \\ &+ \sum_{i=1}^N \log|\det(\frac{\partial f_\theta(\mathbf{x}^i;\mathbf{x}^{c,i})}{\partial {\mathbf{x}^i}^T})|.\end{aligned} \tag{2.15}$$

When constructing the parameterized bijective function f_θ for flow-based conditional generative models, we provide $\mathbf{c}$ as an extra input to both the scaling and translating functions for each affine coupling layer in the normalizing flow, that is, given the input $\mathbf{x}$, the output of the affine coupling layer is given by

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d} \tag{2.16}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d},\mathbf{x}^c)) + \tau(\mathbf{x}_{1:d},\mathbf{x}^c). \tag{2.17}$$

We call the flow-based conditional generative models with the basic structure in (2.16) and (2.17) the *vanilla*-flow.

We can see from (2.16) and (2.17), in each affine coupling layer of *vanilla*-flow, only one part of the output is affected by the condition. In order for the output to have more dependencies on the conditioning input, we extend the structure design in *vanilla*-flow to get a new structure. Specifically, the part of the input that remains unchanged in (2.16)

also goes through scaling and translating transformations to reach the output:

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d} \odot \exp(s(\mathbf{x}^c)) + \tau(\mathbf{x}^c) \tag{2.18}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d}, \mathbf{x}^c)) + \tau(\mathbf{x}_{1:d}, \mathbf{x}^c). \tag{2.19}$$

We call the flow-based conditional generative models with this new structure the *reinforced*-flow. It is worth pointing out that, in (2.18), the scaling and translating functions associated with $\mathbf{x}_{1:d}$ only have the condition $\mathbf{c}$ as their inputs. With this design, the Jacobian matrix of the transformation given in (2.18) and (2.19) is still lower-triangular:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}^T} = \begin{bmatrix} \text{diag}(\exp(s(\mathbf{x}^c))) & 0 \\ * & \text{diag}(\exp(s(\mathbf{x}_{1:d}, \mathbf{x}^c))) \end{bmatrix}. \tag{2.20}$$

As a result, the calculation of the log-determinants of Jacobian and hence the training objective in (2.14) for *reinforced*-flow is as efficient as for the *vanilla*-flow.

Now we apply flow-based conditional generative models to scenario forecasting for residential load. Specifically, we focus on generating a set of scenario forecasts for future load from the given historical realizations. Assuming at time t, we have H observations of the previous realized residential load, which are collected in the vector $\boldsymbol{\ell}_{past} = \{\ell_{t-H}, \cdots, \ell_{t-1}\} \in \mathbb{R}^H$. Given this historical data $\boldsymbol{\ell}_{past}$, we generate scenario forecasts for future H' time points, and H' is referred to as the forecasting horizon. Suppose we have access to the realized load over the H' look-ahead time points, which is denoted by $\boldsymbol{\ell}_{future} \in \mathbb{R}^{H'}$, then $(\boldsymbol{\ell}_{past}, \boldsymbol{\ell}_{future})$ can constitute a training sample for training flow-based models, with $\boldsymbol{\ell}_{past}$ as the conditioning input and $\boldsymbol{\ell}_{future}$ as the data input. Note that the data sample $(\boldsymbol{\ell}_{past}, \boldsymbol{\ell}_{future})$ is actually a time series of length $(H + H')$. If we have a long time series of the realized residential load over W time points and $W \gg H + H'$, then we can break down this long time series into small pieces where each piece corresponds to a time-series data sample and overlaps with the following one. To be specific, the $i-$th piece consists of $(\boldsymbol{\ell}^i_{past}, \boldsymbol{\ell}^i_{future})$, and the following $(i+1)-$th piece consists of $(\boldsymbol{\ell}^{i+1}_{past}, \boldsymbol{\ell}^{i+1}_{future})$, where $\boldsymbol{\ell}^{i+1}_{past}$ is exactly $\boldsymbol{\ell}^i_{future}$. In this way, we can construct our training dataset $\{(\boldsymbol{\ell}^i_{past}, \boldsymbol{\ell}^i_{future})\}_{i=1}^N$ for training flow-based models through the optimization in (2.15).

Suppose the learned value of parameter θ through the optimization in (2.15) is denoted by $\hat{\theta}$, and the associated learned bijective function is denoted by $f_{\hat{\theta}}$. Given any available

historical residential load time series $\boldsymbol{\ell}_{past}$ of length H, we can generate scenario forecasts for the following H' time points using the inverse function of $f_{\hat{\theta}}$ as follows

$$\hat{\boldsymbol{\ell}}_{future} = f_{\hat{\theta}}^{-1}(\mathbf{z}; \boldsymbol{\ell}_{past}) \tag{2.21}$$

where $\mathbf{z} \in \mathbb{R}^k$ is any sample taken from the standard multivariate Gaussian distribution. If we can take N samples from the standard multivariate Gaussian distribution, then we can produce N conditional scenarios for the future residential load through (2.21). The architecture of the flow-based model that we use for training and residential load scenario forecasting is given in Fig. 2.2 and Fig. 2.3.

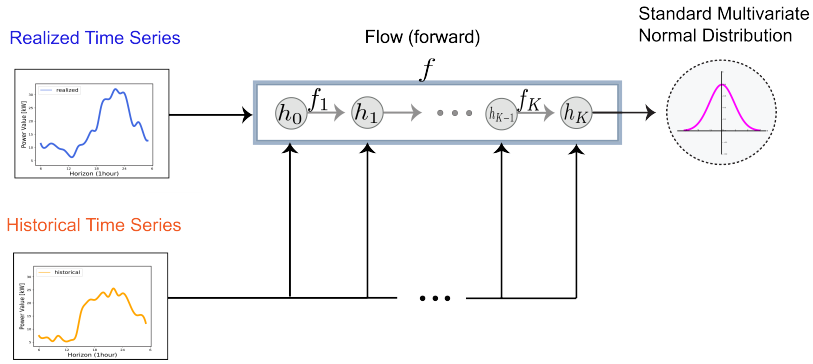


Figure 2.2: Training Phase

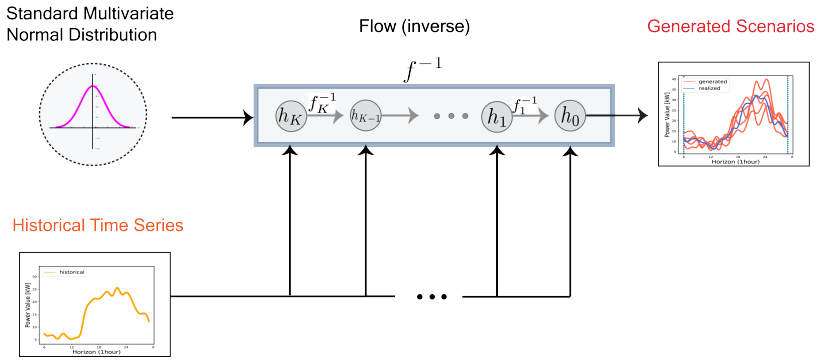


Figure 2.3: Forecasting Phase

2.3.2 Flows with Wasserstein Distance

From Section 2.2 we know the objective function for training flow-based generative models is to maximize the log-likelihood of the training data. It turns out this objective is equivalent to minimizing the Kullback-Leibler (KL) divergence between the true data distribution and the modeled one [42]. To show this, recall the objective function in (2.5), and rewrite it as follows

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{N} \log p_X(\mathbf{x}^i|\theta) \tag{2.22}$$

where $\hat{\theta}$ is the set of trained parameters. Suppose the ground-truth value of the parameter θ is denoted by $\theta^\star$. Since the optimization problem in (2.22) is independent of the ground truth value $\theta^\star$, we can rewrite it as follows

$$\begin{aligned}
\hat{\theta} &= \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log p_X(\mathbf{x}^i|\theta) && (2.23)\\
&= \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log p_X(\mathbf{x}^i|\theta) - \frac{1}{N} \sum_{i=1}^{N} \log p_X(\mathbf{x}^i|\theta^\star) && (2.24)\\
&= \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log \frac{p_X(\mathbf{x}^i|\theta)}{p_X(\mathbf{x}^i|\theta^\star)} && (2.25)\\
&= \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \log \frac{p_X(\mathbf{x}^i|\theta^\star)}{p_X(\mathbf{x}^i|\theta)}. && (2.26)
\end{aligned}$$

The expression in (2.26) is exactly the empirical KL divergence between the true data distribution $p_X(\mathbf{x}|\theta^\star)$ and the modeled data distribution $p_X(\mathbf{x}|\theta)$. That is to say, the goal of the flow-based generative models is to minimize the KL divergence between the true data distribution and the modeled one.

There are well-known advantages and disadvantages in using the KL divergence as the distance measure between two probability distributions [43]. An undesirable effect of using it in scenario generation is that it is asymmetric: in the cases where, for a given point $\mathbf{x}$, the true data distribution $p_X(\mathbf{x}|\theta^\star)$ is non-zero, while the learned distribution $p_X(\mathbf{x}|\theta)$ is close to zero, then the KL divergence can be very large; however, when the true data distribution $p_X(\mathbf{x}|\theta^\star)$ is close to zero at the point $\mathbf{x}$, but the learned distribution $p_X(\mathbf{x}|\theta)$ is non-zero, the KL divergence is small. As a result, by minimizing the KL divergence between the learned

data distribution and the true one, the learned data distribution tend to spread out, which leads to good coverage of the data points that come from the true data distribution, but also tend to create superfluous data points that are not a part of the true distribution. Notably, the learned data distribution can have a relatively larger variance than the true one.

A simple example of this effect can be found in Fig. 2.4. Suppose there is a one-dimensional Gaussian mixture model with two equally weighted components. The first component has mean $\mu_1 = -1.0$ and variance $\sigma_0^2 = 0.1$, while the second has mean $\mu_2 = 1.0$ and variance $\sigma_0^2 = 0.1$. Now suppose we hope to fit a zero-mean Gaussian distribution $Q = \mathcal{N}(0, \sigma^2)$ to this Gaussian mixture model by tuning values of the variance σ^2 to minimize the KL divergence. This is a one dimensional minimization problem and can be solved by simply graphing the KL divergence while varying σ^2. It turns out the optimal σ^2 is around 1.05 and the ground-truth distribution $p_X(x)$ and the optimal distribution $Q^{KL} = \mathcal{N}(0, 1.05)$ are shown in Fig. 2.4. This figure shows that the fitted Gaussian distribution covers both components of the Gaussian mixture model and even has a much larger variance.

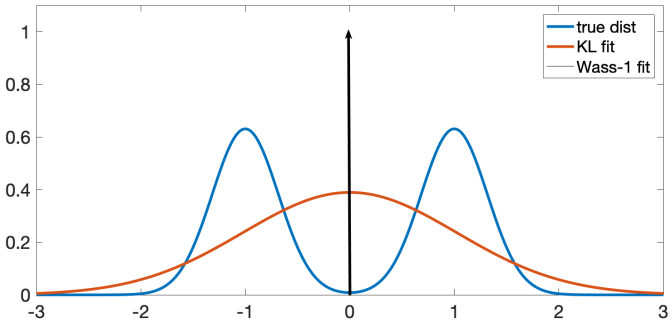


Figure 2.4: An example of fitting a mixed Gaussian distribution with a zero-mean Gaussian distribution using the KL divergence and Wasserstein distance. Under the KL divergence, the fitted distribution tends to be spread out to cover the true distribution, while under the Wasserstein distance the fitted distribution tends to concentrate to minimize the distance between the two components.

Since in many situations the parametrized distribution we learn by minimizing the KL

divergence would not include the true data distribution, it becomes desirable to balance the impact of KL divergence using another distance metric. To this end, we use the Wasserstein distance as a regularizer in the flow objective. This is inspired by the performance of the Wasserstein based generative adversarial networks (WGAN) [37]. The impact of the Wasserstein distance can be seen again in Fig. 2.4. Here we fit the Gaussian mixture model by minimizing the Wasserstein distance between the ground-truth distribution $p_X(x)$ and the fitted distribution $Q = \mathcal{N}(0, \sigma^2)$ in order to decide the optimal value of σ. It turns out that there is a closed-form expression for the Wasserstein-1 distance between two one-dimensional distributions [44]:

$$W_1(P, Q) = \int_0^1 |F^{-1}(z) - G^{-1}(z)| dz, \tag{2.27}$$

where F and G are the cumulative distribution functions (CDFs) of P and Q. Using (2.27), the optimal σ turns out to be close to 0, which is the delta function (point distribution) at 0 as shown in Fig. 2.4. This fit is much more "concentrated" than the fit using KL divergence, but it does not cover the true distribution.

The intuition gained in this toy example carries over to more complex and higher dimensional distributions, which leads to a natural solution of combining the objective of the flow-based generative models (KL divergence) with the Wasserstein distance. This both decreases the variance of the generated scenarios and at the same time generate plausible future realizations that have significantly non-zero probabilities to occur. Specifically, we add a weighted Wasserstein distance metric to the training objective of flow-based generative models:

$$\max_{\theta} p_X(\mathbf{x}|\theta) + \beta W(p_X(\mathbf{x}|\theta^\star, p_X(\mathbf{x}|\theta)) \tag{2.28}$$

We call the flow-based generative models that are trained using the combined objective function in (2.28) $\mathcal{W}$-flows, and the flow-based conditional generative models trained in this way are called *conditional* $\mathcal{W}$-flows, particularly, which include the *vanilla*-$\mathcal{W}$-flow and the *reinforced*-$\mathcal{W}$-flow.

2.4 Simulation Studies

In this section, we study the performance of our proposed flow-based approach in conditional scenario forecasting of residential load. We focus on generating daily scenario forecasts by conditioning on the observed realizations in the previous day with hourly resolution. We first show that our approach can provide accurate forecasts for residential load at different aggregation levels in the form of quantile forecasts and prediction intervals. We also use a practical example in demand response to show that our approach can generate scenarios that cover a wider range of possibilities than the standard method of adding noise to point forecasts, and hence has better performance for planning. We note that Gaussian noise is used for the standard method. Also, scenario forecasting does not generate a central forecast, so we use the median value of a group of generated scenarios for illustrative purposes. Then we quantitatively evaluate our approach to show the scenarios generated using our approach can have better reliability and sharpness than those generated by the standard method.

The experiments in this section are implemented using Pytorch [45] and the latest Glow model [40]. The flow-based model adopted in the simulation is composed of chained 9 blocks of reversible transformations. In the normalizing flow of the adopted flow-based model, the scaling and translating functions that only take the condition as the input are implemented as fully-connected Neural Networks (NNs), while all the other scaling and translating functions are implemented using three-layer 1D-Convolutional Neural Networks (1D-CNNs). Batch-normalization is applied in 1D-CNNs before every layer except the input layer to increase the stability of learning. The activation functions for all scaling transformations are tanh function while rectified linear units (ReLUs) are used as the activation functions in all translating transformations. All models in this section are trained using Adam optimizer [46].

2.4.1 Dataset Description

All experiments in this section use real measurement data from actual homes in the United States. Specifically, hourly measurements of power consumption from households in Austin, Texas are collected and published on Dataport by the Pecan Street Corporation [47].

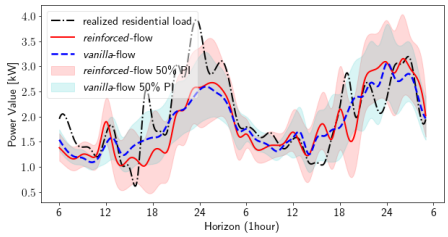
We select the power consumption measurement data of 128 households from 1/1/2013 to 12/31/2017 in the database. We note that, out of the 128 households, only 105 of them are consistent in the dataset. Therefore, we restrict our dataset to these 105 households.[2] The data from 1/1/2013 to 10/1/2017 is used for training and validation, and the last three month's data is used for testing. We conduct our simulation experiments for different aggregation scales, and, particularly, the residential load data from single household, 10 households and 100 households are used. At each aggregation level, we repeat the experiment for 10 independent runs.

2.4.2 Simulation Results

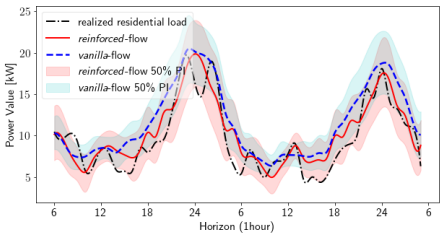
We first show the load scenario forecasting results of our proposed approach for the residential load from a varying number of households. Both structures of the flow-based model, i.e., *reinforced*-flow and *vanilla*-flow, are used for this experiment. We select one 48-hour long sample from the test data at each aggregation level. All the samples are taken from the same month in a year, i.e., October, 2017. These samples are given in Fig. 2.5. In each sample, we plot the median value of the generated scenarios to compare with the realized residential load. We also plot the 50% prediction interval (PI), i.e., the interval between the 25^{th} and the 75^{th} quantiles, as a colored belt to show the confidence level in the generated scenario forecasts.

We can see from Fig. 2.5b and Fig. 2.5c that, for the aggregated residential load of 10 or 100 households, both structures can provide accurate day-ahead forecasts for the coming time and power value of load peak and load valley in the form of median values. At a single household level, the median values of the generated scenario forecasts from both flow structures deviate from the realized load because of the randomness inherent in the dataset. We also note that, when the residential load from 100 households is forecasted, for both flow structures, the 50% prediction interval of the generated scenarios is the narrowest among three aggregation levels, and the realized load is completely covered by the 50% prediction interval. When only a single household is considered, the 50% prediction interval

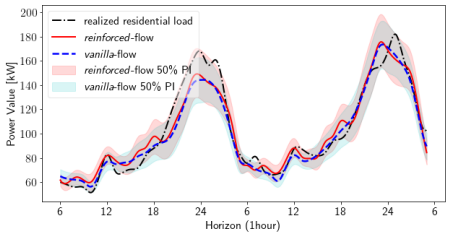
[2]We also note that all the data used in this section can be found on Dataport with each household's ID, location, enrollment date, and withdrawal date recorded.



(a) Single household



(b) 10 households



(c) 100 households

Figure 2.5: Scenario forecasts of residential load from single household (a), 10 households (b), and 100 households (c) are shown. Scenarios generated by the two structures of the flow-based model, i.e., *reinforced*-flow (red) and *vanilla*-flow (blue), are compared.

of the generated scenarios using *reinforced*-flow can cover more parts of the realized load in comparison to that of using *vanilla*-flow. This is because the residential load from a single household has large randomness and is typically hard to be forecasted accurately. Under this circumstance, *reinforced*-flow can have better performance than *vanilla*-flow due to the improved structure design of the affine coupling layer, which makes a stronger coupling relationship between the future load and the historical realizations, as discussed in Section 2.3.1. For the residential load from 10 households, although the randomness is reduced by aggregation, the realized load is still corrupted by a certain degree of noise and show volatility. In this setting, we can see from Fig. 2.5b that *reinforced*-flow also shows better performance than *vanilla*-flow by including more parts of the realized load in its 50%

prediction interval.[3]

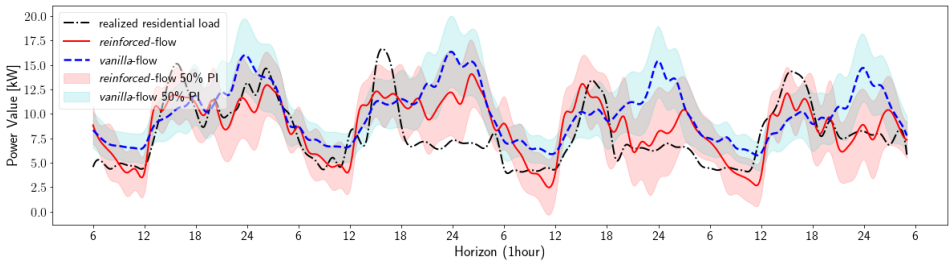
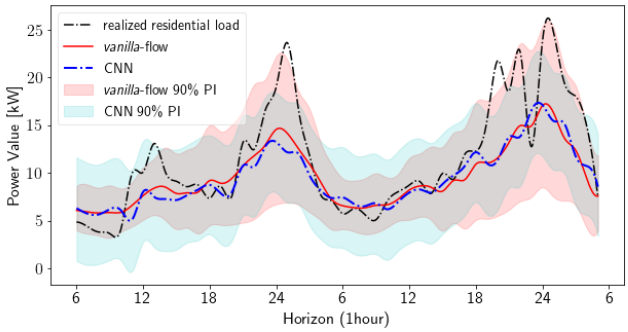


Figure 2.6: An example to show that *reinforced*-flow can generate scenarios that are more closely related to historical observations than *vanilla*-flow. The realized residential load is represented by the black dotted line, the median value of the generated scenarios is plotted as the red solid line for *reinforced*-flow and blue dashed line for *vanilla*-flow, and the colored banded area indicates the 50% prediction interval of generated scenarios. The scenarios generated by *reinforced*-flow are more correlated to the historical realizations and can provide more accurate forecasts even when the realized load goes through sudden changes.

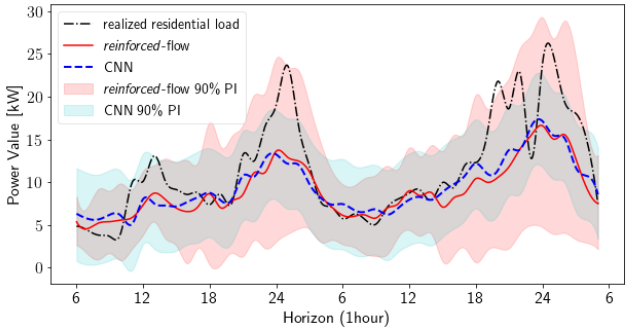
Recall from Section 2.3.1 that we extend the structure design in *vanilla*-flow to a new structure in *reinforced*-flow, where the conditioning input and the output in affine layers have a stronger coupling relationship. To further validate if the new structure in *reinforced*-flow can have better performance in learning more rich information from the condition, we show the forecasting results of both structure designs for a series of four days in Fig. 2.6, where the median values and the 50% prediction intervals of the generated scenarios are plotted. From Fig. 2.6, we can see the change in the pattern of the realized load that the second peak around 24 : 00 becomes flat from the second day. We note that the median value of the scenarios generated by *reinforced*-flow goes through a similar change from the third day to become more close to the realized load. The 50% prediction interval produced by *reinforced*-flow also changes to cover the realized load as much as possible. However,

[3]Since the residential load from a single household is notoriously difficult to be forecasted accurately because of the large randomness, and the forecasting for an aggregation of 100 households is typically an easier task [12], we only use the forecasting results for 10 households as examples in the remaining part of Section 2.4.5 to illustrate the performance of our approach.

the median value and the 50% prediction interval of the scenarios generated by *vanilla*-flow show little changes.



(a) CNN-based baseline and *vanilla*-flow



(b) CNN-based baseline and *reinforced*-flow

Figure 2.7: Scenarios generated by *vanilla*-flow (a) and *reinforced*-flow (b) versus those by adding noise to point forecasts from a CNN for the worst-case planning of the amount of power to purchase.

Next, we consider a practical example in demand response to illustrate the application of our proposed approach in probabilistic load forecasting. We also compare the forecasting performance of our approach against that of the standard method of adding noise to point forecasts. We note that the point forecasts for residential load are derived from a CNN, which is composed of two convolutional layers, followed by a max-pooling layer and a linear output layer. ReLUs are used as the activation functions in convolutional layers. This CNN

takes the 24-dimensional historical residential load as input and produces the point forecasts for the residential load in the next 24 hours.

In this example, utilities want to decide the amount of power to purchase one day ahead to match the residential load in the next day for a group of 10 households. To this end, they use probabilistic load forecasting to know possible residential load in the future. To prevent blackouts or power waste, they target at the 90% prediction interval of the generated scenario forecasts for the worst-case planning of the amount of power to purchase. We select the two days from 11/1/2017 to 11/2/2017 to show the probabilistic load forecasting results of our proposed approach and compare them with the results of the CNN-based standard method. We plot the mean value and the 90% prediction interval of the generated scenarios in Fig. 2.7. Particularly, the CNN-based standard method is compared against *vanilla*-flow in Fig. 2.7a, and compared to *reinforced*-flow in Fig. 2.7b. We can see from Fig. 2.7 that the point forecasts from CNN are very close to the mean value of the generated scenarios using our approach; however, the peak value of the realized residential load is not covered by the 90% prediction interval produced by the CNN-based standard method. By contrast, the realized residential load is almost always included in the 90% prediction interval of the scenarios generated by our approach. That is to say, if the CNN-based standard method is used for probabilistic load forecasting in this example, it is very likely to underestimate the possible residential load in the next day and result in power shortage. For the two days shown in Fig. 2.7, we can see that the bias in the underestimation could be up to 5 kW for a 10-households customer group.

2.4.3 Quantitive Evaluation

In the forecasting literature, two properties have been used to examine the quality of generated scenario forecasts: reliability and sharpness [48, 49]. The reliability of generated scenario forecasts is that they should cover the actual value as much as possible, and the sharpness requires that the generated scenario forecasts should be able to provide a situation-dependent assessment of forecasting uncertainty. For the example of residential load scenario forecasting, it is intuitively expected that the forecasting uncertainty should

not be the same when the power consumption drops to the lowest point and when the peak demand occurs, because the smallest level of load demand is usually the base load and can be more fixed than the peak demand.

To examine the quality of the scenarios generated by our approach, we first evaluate the reliability of the generated scenarios using the Deviation-Coverage Area plot. Particularly, the Deviation-Coverage Area plot gives the amount of deviation from the realized load as a function of the size in the coverage area of quantiles. Specifically, considering a coverage area of size $1-\alpha$, $0 \leqslant \alpha \leqslant 1$, then we calculate the $\alpha/2^{\text{th}}$ and $(1-\alpha/2)^{\text{th}}$ quantiles of the generated scenario forecasts at each time point t. Denote the value of realized residential load at time t by ℓ_t, and the $\alpha/2^{\text{th}}$ and $(1-\alpha/2)^{\text{th}}$ quantiles of the scenario forecasts at time t by $\hat{\ell}_t^{\alpha/2}$ and $\hat{\ell}_t^{1-\alpha/2}$, respectively. Then the amount of deviation from ℓ_t for a coverage area with size $1-\alpha$ can be calculated as follows:

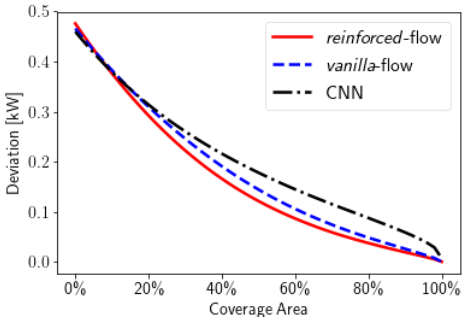
$$dev_t^{1-\alpha} = \begin{cases} 0 & \text{if } \ell_t \in [\hat{\ell}_t^{\alpha/2}, \hat{\ell}_t^{1-\alpha/2}]; \\ \hat{\ell}_t^{\alpha/2} - \ell_t & \text{if } \ell_t < \hat{\ell}_t^{\alpha/2}; \\ \ell_t - \hat{\ell}_t^{1-\alpha/2} & \text{if } \ell_t > \hat{\ell}_t^{1-\alpha/2}. \end{cases} \tag{2.29}$$

Particularly, when $\alpha = 1$, the coverage area has size zero, and both the $\alpha/2^{\text{th}}$ and $(1-\alpha/2)^{\text{th}}$ quantiles become the median values. In this case, the deviation amount for the coverage area with size zero is simply the distance between the realized residential load and the median value of the generated scenarios. After generating the scenario forecasts over T look-ahead time points, we average the amounts of deviation over all T time points for each coverage area size to get the Deviation-Coverage Area plot:

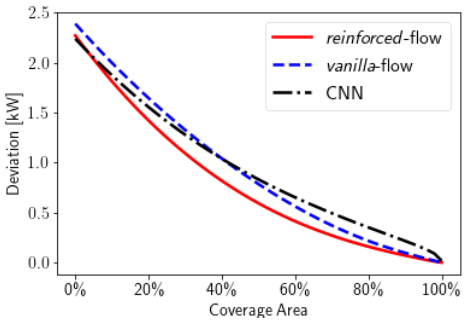
$$Dev^{1-\alpha} = \frac{1}{T}\sum_{t=1}^{T} dev_t^{1-\alpha}. \tag{2.30}$$

Intuitively, the closer the Deviation-Coverage Area plot is to the origin, the higher reliability.

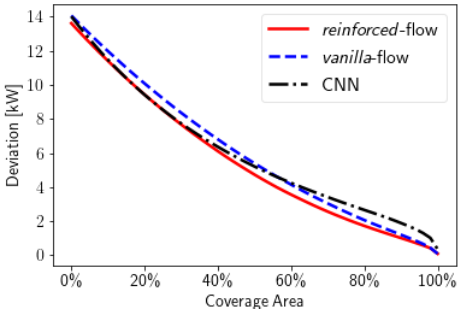
The Deviation-Coverage Area plots of the generated scenarios for the residential load from a varying number of households are given in Fig. 2.8. Particularly, we plot the Deviation-Coverage Area plots for both our approach and the standard method of adding noise to point forecasts from the CNN for comparison. Recall that the closer the Deviation-Coverage Area plot is to the origin, the higher level of reliability. We can see from Fig.



(a) Single household



(b) 10 households

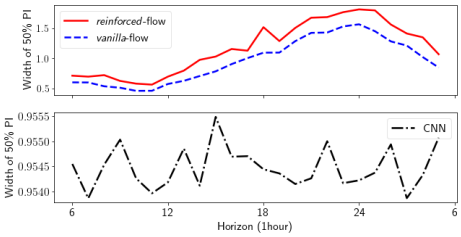


(c) 100 households

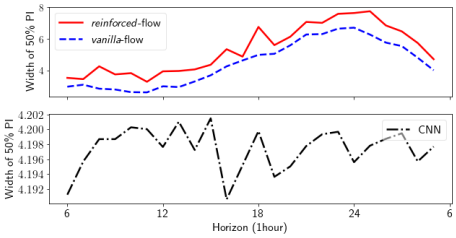
Figure 2.8: Deviation-Coverage Area plots of generated scenario forecasts for residential load of single household (a), 10 households (b), and 100 households (c).

2.8 that the scenarios generated by *reinforced*-flow exhibit the highest level of reliability among all methods at every aggregation level. When a single household is considered, the generated scenarios using *vanilla*-flow can have a higher level of reliability than those using the CNN-based standard method. As the residential load of 10 or 100 households is aggregated, although the point forecasts from the CNN have smaller deviation from the realized residential load than the mean value of the scenarios generated by *vanilla*-flow, the 50% or larger coverage area generated by *vanilla*-flow can be closer to the realized load.

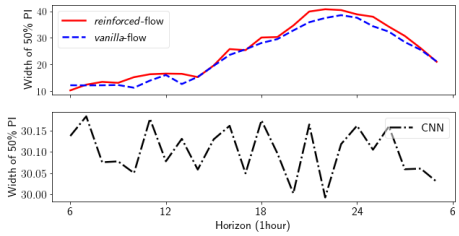
To evaluate the sharpness of the generated scenario forecasts, we calculate the width of the 50% prediction interval, and investigate its variations over 24 hours. The width of the



(a) Single household



(b) 10 households



(c) 100 households

Figure 2.9: The variations of the 50% prediction interval width over 24 hours for the generated scenario forecasts when single household (a), 10 households (b), and 100 households (c) are included. The x-axis is the forecasting horizon at an hourly resolution, and the y-axis is the width of the 50% prediction interval of generated scenarios in the unit of kW.

50% prediction interval for the scenarios generated using our approach is given in Fig. 2.9, and is compared against that using the CNN-based standard method. We can see from Fig. 2.9 that, for all three aggregation levels, the width of the 50% prediction interval of the scenarios generated using our approach has noticeable changes over 24 hours. Particularly, if compared to the realized load in Section 2.4.2, the width can be narrower when the load is relatively small, which represents higher confidence level in the forecasting results, and becomes larger when the load demand reaches the highest point in order for the generated scenarios to cover a wider range of possibilities. By contrast, the prediction interval width of the scenarios generated by the CNN-based standard method almost does not show changes during one day.

2.4.4 Cases Where Our Methods May Fail

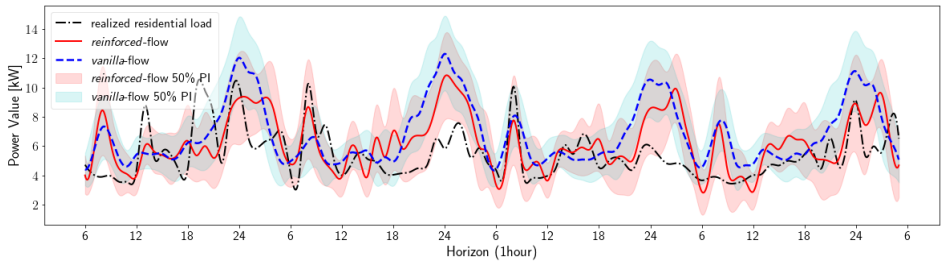


Figure 2.10: A case where the scenarios generated by *reinforced*-flow (red) and *vanilla*-flow (cyan) may fail to provide accurate forecasts. The realized residential load is represented by the black dotted line, the median value of the generated scenarios is plotted as the red solid line for *reinforced*-flow and blue dashed line for *vanilla*-flow, and the colored banded areas indicate the 50% prediction interval of generated scenarios. Starting from the second day, the realized residential load becomes too noisy and can not provide useful information to aid conditional scenario generation. As a result, the flow-based model ignores the noisy historical observations and generate scenarios that are irrelevant to the condition.

In the simulation experiments, we also find there are certain cases where the generated scenarios using our approach may fail to provide accurate forecasts for the future load, and the derived prediction interval may not be able to cover the realized data. To show this, we give an example in Fig. 2.10. We can see from this figure, the realized load becomes noisy and has relatively small values in most of the time. Although one can hardly detect a clear pattern from the realized load in Fig. 2.10, the median value of the generated scenarios using our approach shows a clear pattern that is different than the realized load, and the 50% prediction interval fails to cover the realized load. In this case, the inaccurate scenario forecasting can be explained by the objective function used for training flow-based models. Recall from Section 2.3.1 that the objective of training flow-based conditional generative models is to maximize the conditional log-likelihood on the training set. That is to say, given the historical observations, a flow-based model always generates the scenarios that are most likely to occur under this condition. However, if the historical observations to be

conditioned on are too noisy, then they can not provide useful side information to the flow-based model. Without helpful side information, the flow-based model simply maximizes the log-likelihood instead of conditional log-likelihood. As a result, the scenarios that are likely to occur under normal circumstances are generated by the flow-based model.

2.4.5 Conditional Scenario Generation with Wasserstein Distance

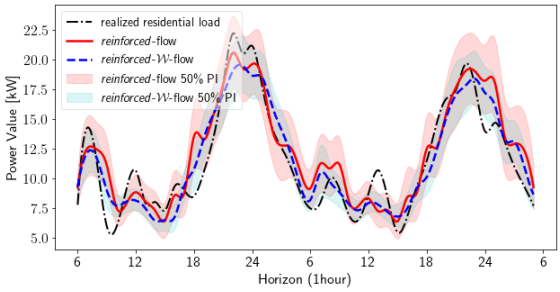
Recall from Section 2.3.2 that we add a weighted Wasserstein distance metric to the training objective of flow-based generative models to balance the large variety of scenarios generated by minimizing the KL divergence and the small variance of those generated by minimizing the Wasserstein distance. In order to validate if the scenarios generated by minimizing both metrics can have smaller variance compared to those generated by only minimizing the KL divergence metric, we use the aggregated residential load data from 10 households, and compare the conditional scenario forecasting results in these two cases. Considering that *reinforced*-flow can have better performance than *vanilla*-flow, as shown in previous simulation results, we take the *reinforced*-flow and its counterpart *reinforced*-$\mathcal{W}$-flow as examples to illustrate the effect of adding a weighted Wasserstein distance metric to the training objective.

Particularly, the Wasserstein distance between the true data distribution and the learned one is calculated through the dual formulation [50]:

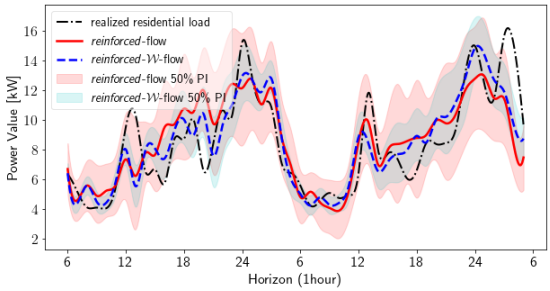
$$\begin{aligned} & W(p_X(\mathbf{x}|\theta^\star), p_X(\mathbf{x}|\theta)) \\ = & \sup_{g:\|g\|_L \leqslant 1} \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x}|\theta^\star)}[g(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x}|\theta)}[g(\mathbf{x})] \end{aligned} \tag{2.31}$$

where g represents any 1-Lipschitz function that maps from $\mathbb{R}^D$ to $\mathbb{R}$, and D is the dimension of $\mathbf{x}$. In our case of conditional scenario generation, the 1-Lipschitz function g is implemented as an 1D-CNN with the condition as an extra input. To enforce the 1-Lipschitz property on g, we utilize the weight clamping technique in [37].

In Fig. 2.11, we show the scenarios generated by *reinforced*-flow and its counterpart *reinforced*-$\mathcal{W}$-flow, and plot the median values and the 50% prediction intervals of the scenarios generated by these two models. Particularly, to validate that *reinforced*-$\mathcal{W}$-flow



(a) A good case of realized residential load



(b) A bad case of realized residential load

Figure 2.11: Scenarios generated by minimizing both KL divergence and Wasserstein distance (*reinforced*-$\mathcal{W}$-flow) versus those generated by only minimizing the KL divergence (*reinforced*-flow). Example (a) represents the case where the realized residential load has less volatility, while (b) represents the case where the realized residential load has larger randomness.

can reduce the variance of the generated scenarios in both cases of large uncertainties and small uncertainties, we have shown two samples in Fig. 2.11, where the residential load in Fig. 2.11b has larger volatility than that in Fig. 2.11a. We can see from Fig. 2.11 that, in both samples, the 50% prediction interval of the scenarios generated by *reinforced*-$\mathcal{W}$-flow is narrower than that in *reinforced*-flow. That is to say, the scenarios generated by *reinforced*-$\mathcal{W}$-flow can have smaller variance and are more centered around the median value. Particularly, the variance of the generated scenarios is reduced more in Fig. 2.11b when the realized load curve is more volatile than in Fig. 2.11a. In each given sample in Fig.

2.11, when the residential load to be forecasted is relatively low, the scenarios generated by *reinforced*-$\mathcal{W}$-flow can have very thin 50% prediction interval. When the residential load comes to the peak value and hence is more unpredictable, the 50% prediction interval of the scenarios generated by *reinforced*-$\mathcal{W}$-flow becomes wider but is still narrower than that in *reinforced*-flow. We note that the median value of the scenarios generated by *reinforced*-$\mathcal{W}$-flow can also provide more accurate forecasts for future residential load than those generated by *reinforced*-flow, especially when the residential load is more volatile as shown in Fig. 2.11b.

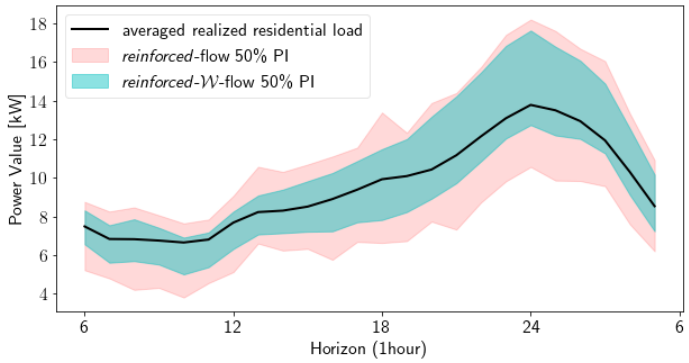


Figure 2.12: The 50% prediction interval of scenarios generated by *reinforced*-$\mathcal{W}$-flow (cyan-colored band) versus that generated by *reinforced*-flow (red-colored band). The black solid line represents the realized residential load. The scenarios generated by *reinforced*-$\mathcal{W}$-flow have narrower 50% prediction interval at all time points compared to those generated by *reinforced*-flow.

To quantitatively investigate the effect of adding a weighted Wasserstein distance metric to the objective function, we calculate the average 50% prediction interval of all generated daily scenarios for both models and plot them in Fig. 2.12. We can see from Fig. 2.12 that, in both models, the 50% prediction interval can completely cover the realized load. However, the 50% prediction interval of the scenarios generated by *reinforced*-$\mathcal{W}$-flow is narrower compared to those generated by *reinforced*-flow at all time points during one day. That is to say, the scenarios generated by minimizing both KL divergence and Wasserstein distance metrics can have smaller variance than those generated by only minimizing KL

divergence. Also, recall from Section 2.4.3 that the width of 50% prediction interval can be used to evaluate the sharpness of generated scenarios. We note that the scenarios generated by *reinforced-$\mathcal{W}$*-flow can also provide situation-dependent confidence levels in forecasting results. Therefore, by adding a weighted Wasserstein distance metric, we can not only reduce the variance of generated scenarios but also keep the sharpness.

2.5 Conclusions

In this chapter, we study the problem of scenario forecasting for a single or a small group of households. Because of the advancement of solar PV, electric vehicles and demand response, accurately forecasting the behavior at the household level becomes an important step in the operation of distribution systems. Since the high variability and small base load make providing accurate point forecasts difficult, we focus on providing a group of scenarios that capture the potential behavior of future load. We adopt the so-called flow-based generation method, where we generate time series representing the future load conditioned on past historical observations. This approach can generate scenarios that are both diverse and follow the true pattern of the load. The simulation results show the flow-based designs significantly outperform existing methods in scenario forecasting for residential load by providing both more reliable and more sharp scenarios.

Chapter 3

A CONVEX NEURAL NETWORK SOLVER FOR DCOPF

3.1 Introduction

The optimal power flow (OPF) problem is a fundamental tool used in the planning and operation of power systems [51, 52, 53]. The OPF problem finds the least cost generator outputs that satisfy the power flow equations and other operational constraints. In this chapter we specifically consider the DCOPF formulation of the OPF problem, which linearizes the power flow equations [54].

The DCOPF problem has been studied extensively for almost half a century and is a workhorse of the power industry. If the generator cost is linear, the DCOPF is a linear program (LP). These LPs can be solved efficiently by a variety of algorithms, which have been implemented in a number of software packages [2, 55]. Today, a DCOPF problem can be solved quickly for fairly large networks [54].

Even though solving a single instance of DCOPF problems is easy, computational challenges are arising because of the increase in renewable resources, since they introduce significant uncertainties into generation and load [56, 57]. To account for these uncertainties, operators often need to repeatedly solve the DCOPF problem for a large number of scenarios [58, 59, 60]. If these scenarios are analyzed close to real-time, then using standard solvers can become too inefficient. For example, suppose a single instance of DCOPF can be solved in 1 second. Then solving a thousand instances would require more than 15 minutes, which would likely be too slow.

Recently, end-to-end neural network architectures have been proposed as surrogates to conventional LP solvers [61, 62, 63, 64, 65, 66, 67]. These neural networks treat load as the input and output the generation values. Since they only require simple function evaluations, they offer orders of magnitude speedup compared to iterative algorithms. Despite the increase in speed, these machine learning approaches lack provable guarantees on their

performances. There are two broad classes of approaches using neural networks for DCOPF. In [61, 62, 63, 64], the neural network is used to directly approximate the functional mapping from load to the optimal generations. In [65, 66, 67], neural networks are used to identify the binding constraints and the solutions are recovered by solving linear systems of equations. Both of these approaches rely on training with a large set of labeled data, then showing the performance of the algorithms through simulations.

A fundamental barrier in adopting these methods in practice is the need to show that small *training error* implies small *testing errors*. That is, we need to show that the method *generalizes*. Using machine learning for DCOPF is most useful when operators are faced with unfamiliar conditions, but there are many examples when machine learning precisely fails in these conditions [68, 69, 70]. The hesitancy in using machine learning (especially deep learning) methods also comes from the perception that they rely on "black-boxes" that are hard to understand [71, 72].

In this chapter we propose a two-step approach for solving DCOPF. Firstly, a neural network is used to learn the *value* (i.e., the optimal cost) of the DCOPF, and its gradient with respect to the load is the locational marginal prices (LMPs). Then the binding constraints are identified based on the LMPs. This process is robust in the sense that if they are approximately correct, and the binding constraints are correctly identified. We provide formal guarantee to the generalization capability of this method by directly designing the fundamental features of the DCOPF problem into the machine learning algorithm. Specifically, we constrain the neural network architecture, and building KKT conditions into the training process.

The first technique we use to guarantee generalization is to constrain the neural network to have an input convex structure, since the cost of the DCOPF is a convex function of the loads. Therefore, we use input convex neural networks (ICNNs) to learn this relationship [73, 74, 75]. The ICNNs represent functions that are convex from input to the output by using ReLU as the activation functions and restricting the weights to be nonnegative. Then the convex structure follows from the composition of convex functions [74]. It turns out that the convex structure of the neural network is key to its generalization capability.

We show that a small training error for a finite number of samples implies that the

error would be small for entire regions of inputs. Intuitively, this means that the gradient of the function (the LMPs) would be roughly correct as long as some points in the input region are sampled during training. This result is in contrast with standard generalization results in the literature, where most are of a statistical nature [76]. Instead, we show that the structure of the neural network is the key, since if convexity is not imposed, we can construct cases with zero training error but arbitrary large testing errors.

The second technique we use is to add KKT conditions to the training process. Perhaps the most direct way to improve generalization is to increase the number of labeled samples. However, for even moderately large power systems, covering the whole load space with labeled data is intractable due to the curse of dimensionality. We overcome this challenge by using that at optimality, the primal and dual variables satisfy the KKT conditions [77]. Interpreting the dual variables as the partial derivatives of the value function with respect to different parameters, the KKT conditions can be written as a set of partial differential equations. We train the ICNN by minimizing the training loss that is based on this set of partial differential equations. This enables us to include a much larger set of inputs without explicitly adding labeled training data.

The challenge in solving DCOPF repeatedly for different loads is similar in spirit to solving linear systems of equations for changing right-hand-side vectors. In this chapter we are trying to solve a parametric optimization problem rather than a parametric linear system [78]. However, unlike LU factorization, it is hard to guarantee that the machine learning methods would always recover the right answer. The approach in [79] can bound the worst case errors for a trained neural network with fixed parameters. But these types of ex post analysis is hard to generalize and do not shed light on why a method may perform better or worse. In this chapter, we show how designing the structure of the neural network can lead to more robust guarantees. A similar observation was made in the context of voltage regulation problems in [80].

This chapter is organized as follows. Section 3.2.2 provides the DCOPF model and Section 3.3.2 describes the solution method. Section 3.4.2 describes the neural network design and the training loss based on KKT conditions, and Section 3.5.3 states and proves the generalization guarantees. Simulations illustrating the results are shown in Section 3.6.2.

Section 3.7 concludes this chapter.

3.2 DCOPF Model

3.2.1 Model

Consider a power system where the n buses are connected by m edges. For each of the bus, we let p_i^g denote the output of the generator located at the bus, and let ℓ_i denote the load consumed at the bus. Let $\mathbf{p}^g = (p_1^g, \ldots, p_n^g)$ and $\ell = (\ell_1, \ldots, \ell_n)$ be the generation and load vectors, respectively. The generation cost vector is denoted as $\mathbf{c} \in \mathbb{R}^n$. We assume that $\mathbf{c}$ is non-negative and has at least one positive component. We assume the system is connected. For notational simplicity, we assume that all buses have generation and load. Without loss of generality, we assume p_i^g is bounded by 0 and $\bar{p}_i^g$.[1] If bus i does not have any generation capability, we set $\bar{p}_i^g = 0$.

The line flows are related to the bus power injections through a linear relationship. Because of Kirchhoff's laws, not all flows in the m lines are independent. In particular, there are only $n-1$ fundamental flows and the rest of $m-n+1$ flows are linear combinations of the fundamental ones [81, 82]. Let $\mathbf{p}^e \in \mathbb{R}^{n-1}$ denote these $n-1$ fundamental flows. More details on line flow modeling is given in Appendix A.1.

The DCOPF problem asks for the least cost generations while satisfying all the loads and flow constraints:

$$J^\star(\ell) = \min_{\mathbf{p}^g, \mathbf{p}^e} \ \mathbf{c}^T \mathbf{p}^g \tag{3.1a}$$

$$\text{s.t. } \mathbf{0} \leqslant \mathbf{p}^g \leqslant \bar{\mathbf{p}}^g \tag{3.1b}$$

$$-\bar{\mathbf{p}}^f \leqslant \mathbf{M}^F \mathbf{p}^e \leqslant \bar{\mathbf{p}}^f \tag{3.1c}$$

$$\mathbf{p}^g + \mathbf{M}^B \mathbf{p}^e = \ell, \tag{3.1d}$$

where the matrix $\mathbf{M}^F \in \mathbb{R}^{m \times (n-1)}$ maps $\mathbf{p}^e$ to the flows on all edges, and the matrix $\mathbf{M}^B \in \mathbb{R}^{n \times (n-1)}$ is the modified incidence matrix that maps the fundamental flows to the

[1] Since the power flow equations are linear, nonzero lower bounds can be shifted to be zero by subtracting a constant from the generation values.

nodal power injections. The value of the optimization problem is denoted as $J^\star(\boldsymbol{\ell})$ and the optimal solution is denoted as $\mathbf{p}^{g\star}(\boldsymbol{\ell})$.

The DCOPF problem in (3.1) is an LP and is readily solved by most optimization packages such as CPLEX or Gurobi [2]. These solvers have been optimized to the point that a single LP can be solved in the order of seconds even for large systems. The challenge comes from the fact that repeatedly solving (3.1) for changing loads can be time-consuming, even if the time it takes for a single instance is small. The formulation in (3.1) can be easily extended to quadratic costs and is given in Appendix A.4.

3.2.2 Example

we present a small example that is used to illustrate many of the points in this chapter. Consider a single-bus system with three generators (with cost \$1/MW, \$2/MW and \$3/MW, respectively) serving a load. The DCOPF problem becomes

$$J^\star(\ell) = \min\ p_1^g + 2p_2^g + 3p_3^g \tag{3.2a}$$

$$\text{s.t. } 0 \leqslant p_i^g \leqslant \overline{p}_i^g, i = 1, 2, 3 \tag{3.2b}$$

$$p_1^g + p_2^g + p_3^g = \ell. \tag{3.2c}$$

Figure 3.1 plots the cost against the load. In an end-to-end approach, the goal is to learn the generations directly. In the next section we will introduce a method that learns the curve $J^\star(\ell)$ and its derivatives.

3.3 Learning Active Constraints

The goal of using machine learning is to avoid resolving (3.1) every time the load changes. A number of algorithms have been proposed to directly learn the functional mapping from $\boldsymbol{\ell}$ to $\mathbf{p}^{g\star}(\boldsymbol{\ell})$ [65, 66, 67]. However, it is difficult to ensure the learned solutions satisfy the constraints in (4.1b) to (3.1d). For the example in Fig. 3.1, each of the generations have upper and lower bounds as well as the load balance constraint (sum of generation is equal to the load). Instead of using end-to-end neural networks, the mapping $J^\star(\boldsymbol{\ell})$ is learned. Then the associated dual variables are obtained from the global dependence on the right-hand-side vector in the LP. With the value of the dual variables, we are able to find out

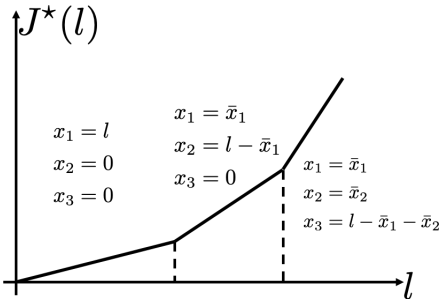


Figure 3.1: The cost curve of a single bus load with three generators. Let x_1 be p_1^g, x_2 be p_2^g, and x_3 be p_3^g. The curve is piecewise linear, convex and increasing, with each piece corresponding to a different generation profile.

a set of active constraints for (3.1). The exact value of $\mathbf{p}^{g\star}(\boldsymbol{\ell})$ is then found by solving a system of linear equations.

3.3.1 Global dependence on the right-hand side vector

The global dependence of the optimal cost function $J^\star(\boldsymbol{\ell})$ on the load vector $\boldsymbol{\ell}$ can be found through standard duality theory [77]. The Lagrangian associated with (3.1) is

$$\mathcal{L}(\mathbf{p}^g, \mathbf{p}^e, \underline{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}}, \underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}, \boldsymbol{\mu}) = \mathbf{c}^T\mathbf{p}^g - \underline{\boldsymbol{\tau}}^T\mathbf{p}^g + \bar{\boldsymbol{\tau}}^T(\mathbf{p}^g - \bar{\mathbf{p}}^g) - \underline{\boldsymbol{\lambda}}^T(\bar{\mathbf{p}}^f + \mathbf{M}^F\mathbf{p}^e) + \bar{\boldsymbol{\lambda}}^T(\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f) + \boldsymbol{\mu}^T(\boldsymbol{\ell} - \mathbf{p}^g - \mathbf{M}^B\mathbf{p}^e), \quad (3.3)$$

where $\underline{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}} \in \mathbb{R}^n$ are the dual variables associated with generator capacity constraints (4.1b), $\underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}} \in \mathbb{R}^m$ are the dual variables associated with flow capacity constraints (3.1c), and $\boldsymbol{\mu} \in \mathbb{R}^n$ are the dual variables associated with equality constraints (3.1d). The dual variables $\boldsymbol{\mu}$ are called the locational marginal prices (LMPs) since they represent the marginal cost of supplying one more unit of power at a bus [83].

The dual problem of (3.1) is

$$\max_{\underline{\boldsymbol{\tau}},\bar{\boldsymbol{\tau}},\underline{\boldsymbol{\lambda}},\bar{\boldsymbol{\lambda}},\boldsymbol{\mu}} \quad \boldsymbol{\mu}^T\boldsymbol{\ell} - \underline{\boldsymbol{\lambda}}^T\bar{\mathbf{p}}^e - \bar{\boldsymbol{\lambda}}^T\bar{\mathbf{p}}^e - \bar{\boldsymbol{\tau}}^T\bar{\mathbf{p}}^g \tag{3.4}$$

$$\text{s.t.} \quad \mathbf{c} - \underline{\boldsymbol{\tau}} + \bar{\boldsymbol{\tau}} - \boldsymbol{\mu} = \mathbf{0} \tag{3.4a}$$

$$-\mathbf{M}^{F^T}\underline{\boldsymbol{\lambda}} + \mathbf{M}^{F^T}\bar{\boldsymbol{\lambda}} - \mathbf{M}^{B^T}\boldsymbol{\mu} = \mathbf{0} \tag{3.4b}$$

$$\underline{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}}, \underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}} \geqslant \mathbf{0}. \tag{3.4c}$$

We assume that the load $\boldsymbol{\ell}$ is feasible. From the value of $\boldsymbol{\mu}$, we can learn the active constraints set for (3.1). To be specific, the optimal solutions for (3.1) are associated with the following active/inactive constraints through the value of $\boldsymbol{\mu}^\star$:

$$p_i^{g\star} = \begin{cases} 0, & \text{if } \mu_i^\star - c_i < 0 \\ \bar{p}_i^g, & \text{if } \mu_i^\star - c_i > 0 \\ (0, \bar{p}_i^g), & \text{otherwise,} \end{cases} \tag{3.5}$$

and

$$p_i^{e\star} = \begin{cases} \bar{p}_i^f, & \text{if } \bar{\lambda}_i^\star - \underline{\lambda}_i^\star > 0 \\ -\bar{p}_i^f, & \text{if } \bar{\lambda}_i - \underline{\lambda}_i^\star < 0 \\ (-\bar{p}_i^f, \bar{p}_i^f), & \text{otherwise,} \end{cases} \tag{3.6}$$

where, given the value of $\boldsymbol{\mu}^\star$, the value of $\bar{\lambda}_i^\star - \underline{\lambda}_i^\star$ can be determined by solving the following optimization problem:

$$\min_{\boldsymbol{\nu}} \ \|\boldsymbol{\nu}\|_1 \tag{3.7}$$
$$\text{s.t. } \mathbf{M}^{F^T} diag(\bar{p}_1^f, \cdots, \bar{p}_m^f)\boldsymbol{\nu} = \mathbf{M}^{B^T}\boldsymbol{\mu},$$

where $diag(\cdot)$ is a diagonal matrix. The value of $\bar{\lambda}_i^\star - \underline{\lambda}_i^\star$ is related to $\boldsymbol{\nu}$ by $\bar{\lambda}_i^\star - \underline{\lambda}_i^\star = \nu_i/\bar{p}_i^f$. Due to space constraints, we skip the detailed derivations. They are straightforward and can be found in the online companion in [67].

It turns out the learning problem becomes much simpler from the dual problem. Instead of directly learning the optimal solutions or the active constraints (neither are continuous in the load), it suffices to learn $J^\star$, which is a scalar function that is continuous in $\boldsymbol{\ell}$. The multipliers μ can be then recovered from the following theorem about the global dependence of the optimal cost $J^\star(\boldsymbol{\ell})$ on load $\boldsymbol{\ell}$:

Theorem 3.1. *A vector $\boldsymbol{\mu}^\star$ is an optimal solution to the dual problem (3.4) if and only if it is a (sub)gradient of the optimal cost $J^\star(\boldsymbol{\ell})$ at the point $\boldsymbol{\ell}$, that is,*

$$\nabla_{\boldsymbol{\ell}} J^\star = \boldsymbol{\mu}^\star. \tag{3.8}$$

The proof of this theorem is standard and can be found, for example, in [84]. The subgradient part of the statement comes from the fact that $J^\star$ is differentiable for almost all $\boldsymbol{\ell}$, but not everywhere. If this is the case, $\boldsymbol{\mu}^\star$ is customarily taken as the (componentwise) largest vector in the set of subgradients.

3.3.2 Solving DCOPF with known marginal prices

Suppose we can learn the cost function $J^\star$. Then $\boldsymbol{\mu}$ (the gradient with respect to the input) can be easily obtained through back propagation. For a nondegenerative LP problem, there would be exactly the same number of constraints as there are variables. Therefore, after using $\boldsymbol{\mu}^\star$ to find the active constraint sets, a linear system of equations is solved to find the optimal solution.

We summarize the algorithm to solving problem (3.1) after the value of $\boldsymbol{\mu}$ is known in *Algorithm 3-1.* This algorithm can offer an order of magnitude speedup compared to iterative solvers, since it only requires solving a sparse linear system of equations. This speed up is comparable to end-to-end regression methods which uses a feed forward neural network to obtain the generation solutions [61] (the detailed computation times are given in [67]). Here we concentrate on the feasibility and generalization properties of our proposed approach.

3.4 Network Architecture and Training Loss Design

The previous algorithm essentially states that if we can learn $\boldsymbol{\mu}^\star$ well, then we can obtain the optimal solution to (3.1). Note that $\boldsymbol{\mu}^\star$ need not to be learned perfectly. Take the middle segment in Fig. 3.1 as an example. As long as the learned $J(\ell)$ has a derivative between 2 and 3 in this segment, we would detect the correct binding constraints. Therefore the key to success is to ensure that the learned $\boldsymbol{\mu}^\star$ always have small error. However, small training error does not guarantee small generalization error. The effectiveness of

Algorithm 3-1: Solving DCOPF with given LMPs
Inputs: $\boldsymbol{\mu}$, $\boldsymbol{\ell}$
Parameters: $\mathbf{M}^B$, $\mathbf{M}^F$, $\mathbf{c}$, $\bar{\mathbf{p}}^f$, $\bar{\mathbf{p}}^g$
1: Given $\boldsymbol{\mu}$, identify active nodal constraints using (3.5)
2: Given $\boldsymbol{\mu}$, identify active flow constraints using (3.6)
3: $\mathsf{EquationSolver}(\mathbf{p}^g + \mathbf{M}^B\mathbf{p}^e = \boldsymbol{\ell}$, active constraints)
Outputs: Optimal solutions $\mathbf{p}^{g\star}$ to (3.1)

Table 3.1: Solving DCOPF for given LMPs.

most machine learning algorithms are demonstrated through simulations, but engineering applications usually require some a priori guarantees, since well-trained models can fail to make reasonable inference on unseen input samples. In this section, we introduce two design features to guarantee the generalization ability of the trained model. By generalization, we mean a neural network that has a small training error in learning $\boldsymbol{\mu}^\star$ should have small testing errors on new samples. To guarantee generalization, we first utilize the convexity of the cost function to train an Input Convex Neural Network (ICNN). Then we leverage the Karush-Kuhn-Tucker (KKT) optimality conditions to design the training loss.

3.4.1 ICNN architecture

A useful result from linear programming that constrain the structure of $J^\star$ is that it is a convex function:

Theorem 3.2. *The optimal cost $J^\star(\boldsymbol{\ell})$ is a convex function of $\boldsymbol{\ell}$ with its domain as the set of all feasible loads. .*

The proof of this theorem is again standard and can be found in [84]. We adopt a special category of deep neural networks (DNNs), called Input Convex Neural Network (ICNN) to leverage this result [74, 73]. The network architecture that we use is shown in Fig. 3.2. The basic construction of ICNNs comes from composition of convex functions. Given two functions g and h, if g is convex and h is convex and nondecreasing, $f = g \circ h$ is convex.

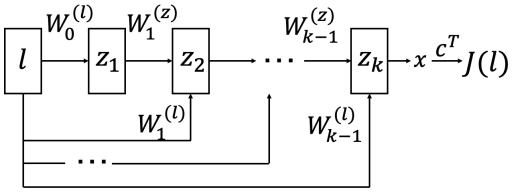


Figure 3.2: The architecture of the trained ICNN. The weights $W_1^{(z)}, \ldots, W_{k-1}^{(z)}$ are restricted to be nonnegative. The pass through links $W_1^{(\boldsymbol{\ell})}, \ldots, W_{k-1}^{(\boldsymbol{\ell})}$ are not sign restricted.

ICNNs satisfy this property by using ReLU as the activation functions and restricting the weights of the network to be nonnegative. This construction is shown to approximate all Lipschitz convex functions arbitrarily closely [74].

Suppose the fully-connected ICNN has $k+1$ layers, i.e., k hidden layers with "passthrough" and one extra linear output layer. The architecture can be written as follows:

$$\mathbf{z}_{i+1} = \sigma(\mathbf{W}_i^{(z)}\mathbf{z}_i + \mathbf{W}_i^{(\ell)}\boldsymbol{\ell} + \mathbf{b}_i), \text{ for } i = 0, \cdots, k-1 \tag{3.9}$$

$$\hat{J} = \mathbf{c}^T(\mathbf{z}_k), \tag{3.10}$$

where $\mathbf{z}_i$ represents the output of the i-th hidden layer, $\mathbf{W}_i^{(z)}$ represents the weight that connects the $i-1$-th hidden layer to the i-th hidden layer, and they are restricted to be nonnegative. The weights $\mathbf{W}_{i-1}^{(\ell)}$ represent the matrices that directly connects the input $\boldsymbol{\ell}$ to the i-th hidden layer. The symbol $\sigma(\cdot)$ represents the ReLU activation function.

We let $\boldsymbol{\theta}$ denote all trainable parameters in (3.9) and (3.10), i.e., $\boldsymbol{\theta} = \{\mathbf{W}_{1:k-1}^{(z)}, \mathbf{W}_{0:k-1}^{(\ell)}, \mathbf{b}_{0:k-1}\}$, and the parameterized function $f_{\boldsymbol{\theta}}(\cdot)$ denote the mapping from $\boldsymbol{\ell}$ to $\hat{J}$. Then (3.9) and (3.10) can be written in a more compact form as $\hat{J} = f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$. The estimated value of $\boldsymbol{\mu}$ can be obtained by taking the derivative of $f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ with respect to $\boldsymbol{\ell}$, and is denoted by $\hat{\boldsymbol{\mu}} = \nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$. We show in Section 3.5.3 that convexity is fundamental to the generalization property of the neural network.

The goal of training the network is to learn the value of $\boldsymbol{\theta}$ which minimizes a specified loss function L, i.e.,

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} L\big(f_{\boldsymbol{\theta}}(\boldsymbol{\ell}), \nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell})\big). \tag{3.11}$$

Next, we illustrate how to construct this loss function.

3.4.2 Capturing KKT conditions

When the load vector $\boldsymbol{\ell}$ changes by certain amounts, the active constraint set and therefore the value of $\boldsymbol{\mu}^\star$ remain unchanged. In fact, we can divide the feasible input space of $\boldsymbol{\ell}$ into a finite number of regions [2]. Each of these region is a convex polytope and corresponds to a different combination of active constraints and a value of $\boldsymbol{\mu}^\star$.

If we have a number of ground-truth values of $(J^\star, \boldsymbol{\mu}^\star)$ for every possible region, then we can train the neural network by minimizing the regression loss. However, the number of possible regions grows quickly with respect to size of the system. For moderately sized system, there maybe a large number of regions and it is unlikely that historical observations would include data in every region. Even if data is generated offline, exhaustively covering all of the regions with labeled data become cumbersome.

If the data set only samples a small number of regions, the model trained by minimizing the regression loss performs poorly for input samples that come from unseen regions. This is not unexpected since there is no data to make prediction for these unseen regions. But in practice the value of using machine learning is to quickly determine what might happen for a large number of loads where some would come from unseen regions. Interestingly, because we are solving a well-defined optimization problem, we can mitigate this data challenge again by looking at duality theory.

We develop an augmented training approach for the neural network $f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ based on

[2]Note that the division of the feasible input space is just for the analysis and illustrative purpose. We do not do the division in the practice and the implementation of our algorithm does not require this division either.

violations of KKT conditions. Recall KKT conditions for the LP in (3.1) is

$$\mathbf{c} - \underline{\boldsymbol{\tau}} + \bar{\boldsymbol{\tau}} - \boldsymbol{\mu} = \mathbf{0} \tag{3.12a}$$

$$-\mathbf{M}^{F^T}\underline{\boldsymbol{\lambda}} + \mathbf{M}^{F^T}\bar{\boldsymbol{\lambda}} - \mathbf{M}^{B^T}\boldsymbol{\mu} = \mathbf{0} \tag{3.12b}$$

$$\mathbf{0} \leqslant \mathbf{p}^g \leqslant \bar{\mathbf{p}}^g \tag{3.12c}$$

$$-\bar{\mathbf{p}}^f \leqslant \mathbf{M}^F\mathbf{p}^e \leqslant \bar{\mathbf{p}}^f \tag{3.12d}$$

$$\mathbf{p}^g + \mathbf{M}^B\mathbf{p}^e = \boldsymbol{\ell} \tag{3.12e}$$

$$\underline{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}}, \underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}} \geqslant \mathbf{0} \tag{3.12f}$$

$$\underline{\tau}_i p_i^g = 0,\ \bar{\tau}_i(p_i^g - \bar{p}_i^g) = 0,\ \forall i \in \{1, \cdots, n\} \tag{3.12g}$$

$$\underline{\lambda}_j(\bar{p}_j^f + M_j^F\mathbf{p}^e) = 0 \tag{3.12h}$$

$$\bar{\lambda}_j(M_j^{F^T}\mathbf{p}^e - \bar{p}_j^f) = 0,\ \forall j \in \{1, \cdots, m\}, \tag{3.12i}$$

where M_j^F is the $j-$th column of matrix $\mathbf{M}^{F^T}$.

Before introducing the loss term related to violations of KKT conditions, we have the following lemma:

Theorem 3.3. *The dual variables $\underline{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}}, \underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}}, \boldsymbol{\mu}$ satisfy the KKT conditions in* (3.12) *if and only if they satisfy equations* (3.13)*:*

$$[\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} = \mathbf{0} \tag{3.13a}$$

$$[\underline{\boldsymbol{\lambda}} - (\mathbf{M}^F\mathbf{p}^e + \bar{\mathbf{p}}^f)]^+ - \underline{\boldsymbol{\lambda}} = \mathbf{0} \tag{3.13b}$$

$$[\bar{\boldsymbol{\tau}} + (\mathbf{p}^g - \bar{\mathbf{p}}^g)]^+ - \bar{\boldsymbol{\tau}} = \mathbf{0} \tag{3.13c}$$

$$[\underline{\boldsymbol{\tau}} - \mathbf{p}^g]^+ - \underline{\boldsymbol{\tau}} = \mathbf{0}. \tag{3.13d}$$

Proof. Here we consider (3.13a) and the rest follow in similar fashions. In particular, we prove the following two conditions are equivalent:

1. $\mathbf{M}^F\mathbf{p}^e \leqslant \bar{\mathbf{p}}^f, \bar{\boldsymbol{\lambda}} \geqslant 0, (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f) \odot \bar{\boldsymbol{\lambda}} = 0$
2. $[\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} = \mathbf{0}$

where $\odot$ represents element-wise multiplication.

First we show that 1) implies 2). If $\mathbf{M}^F\mathbf{p}^e < \bar{\mathbf{p}}^f$, then we must have $\bar{\boldsymbol{\lambda}} = 0$ from the complementary slackness condition $(\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f) \odot \bar{\boldsymbol{\lambda}} = 0$. So

$$[\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} = [\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f]^+ = 0.$$

If $\mathbf{M}^F\mathbf{p}^e = \bar{\mathbf{p}}^f$ and $\bar{\boldsymbol{\lambda}} \geqslant 0$, then we have

$$[\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} = [\bar{\boldsymbol{\lambda}}]^+ - \bar{\boldsymbol{\lambda}} = 0.$$

Next we show 2) implies 1). The right-hand-side implies the following dual feasibility since $[\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} = \mathbf{0}$ gives

$$\bar{\boldsymbol{\lambda}} = [\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f))]^+ \geqslant \mathbf{0}.$$

Suppose the primal feasibility does not hold, i.e., $\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f \geqslant 0$, then

$$[\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} \tag{3.14}$$

$$= \bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f) - \bar{\boldsymbol{\lambda}} \tag{3.15}$$

$$= \mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f \tag{3.16}$$

$$= 0. \tag{3.17}$$

Therefore, we at most have $\mathbf{M}^F\mathbf{p}^e = \bar{\mathbf{p}}^f$ and $\mathbf{M}^F\mathbf{p}^e$ cannot exceeds $\bar{\mathbf{p}}^f$.

For complementary slackness, suppose $\mathbf{M}^F\mathbf{p}^e < \bar{\mathbf{p}}^f$ but $\bar{\boldsymbol{\lambda}} \neq 0$, then we have

$$[\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} \tag{3.18}$$

$$= \begin{cases} \text{either } \bar{\boldsymbol{\lambda}} + (\mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f) - \bar{\boldsymbol{\lambda}} = \mathbf{M}^F\mathbf{p}^e - \bar{\mathbf{p}}^f = 0 \\ \text{or } 0 - \bar{\boldsymbol{\lambda}} = 0, \end{cases} \tag{3.19}$$

which contradicts our assumption. So, if $\mathbf{M}^F\mathbf{p}^e < \bar{\mathbf{p}}^f$, we must have $\bar{\boldsymbol{\lambda}} = 0$. Suppose $\bar{\boldsymbol{\lambda}} > 0$ but $\mathbf{M}^F\mathbf{p}^e < \bar{\mathbf{p}}^f$, we arrive a similar contradiction. ■

Note that dual variables $\underline{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}}, \underline{\boldsymbol{\lambda}}$, and $\bar{\boldsymbol{\lambda}}$ can all be represented in terms of $\boldsymbol{\mu}$. To be specific, based on (3.12a) and (3.12f), we can express the dual solutions $\underline{\boldsymbol{\tau}}$ and $\bar{\boldsymbol{\tau}}$ as follows

$$\underline{\boldsymbol{\tau}}(\boldsymbol{\mu}) = [\boldsymbol{\mu} - \mathbf{c}]^+ - (\boldsymbol{\mu} - \mathbf{c}) \tag{3.20a}$$

$$\bar{\boldsymbol{\tau}}(\boldsymbol{\mu}) = [\boldsymbol{\mu} - \mathbf{c}]^+, \tag{3.20b}$$

where the notation $[a]^+$ represents $[a]^+ = \max\{a, 0\}$. The dual solutions $\underline{\boldsymbol{\lambda}}$ and $\bar{\boldsymbol{\lambda}}$ can be expressed in terms of $\boldsymbol{\mu}$ by solving the $\ell 1$-minimization problem in (3.7), and we denote them by $\underline{\boldsymbol{\lambda}}(\boldsymbol{\mu})$ and $\bar{\boldsymbol{\lambda}}(\boldsymbol{\mu})$, respectively.

Plugging the expressions $\underline{\boldsymbol{\tau}}(\boldsymbol{\mu})$, $\bar{\boldsymbol{\tau}}(\boldsymbol{\mu})$, $\underline{\boldsymbol{\lambda}}(\boldsymbol{\mu})$ and $\bar{\boldsymbol{\lambda}}(\boldsymbol{\mu})$ into (3.13a)-(3.13d), we obtain a system of equations only related to $\boldsymbol{\mu}$. To capture the KKT optimality conditions in (3.12), we define *violation degrees* associated with equations (3.13) as follows

$$\nu_{\bar{\lambda}} = [\bar{\boldsymbol{\lambda}} + (\mathbf{M}^F \mathbf{p}^e - \bar{\mathbf{p}}^f)]^+ - \bar{\boldsymbol{\lambda}} \tag{3.21a}$$

$$\nu_{\underline{\lambda}} = [\underline{\boldsymbol{\lambda}} - (\mathbf{M}^F \mathbf{p}^e + \bar{\mathbf{p}}^f)]^+ - \underline{\boldsymbol{\lambda}} \tag{3.21b}$$

$$\nu_{\bar{\tau}} = [\bar{\boldsymbol{\tau}} + (\mathbf{p}^g - \bar{\mathbf{p}^g})]^+ - \bar{\boldsymbol{\tau}} \tag{3.21c}$$

$$\nu_{\underline{\tau}} = [\underline{\boldsymbol{\tau}} - \mathbf{p}^g]^+ - \underline{\boldsymbol{\tau}} \tag{3.21d}$$

$$\nu_p = \boldsymbol{\ell} - \mathbf{p}^g - \mathbf{M}^B \mathbf{p}^e. \tag{3.21e}$$

Based on *violation degrees* given in (3.13a)-(3.13d), we define the loss term to minimize violations of KKT conditions as follows

$$\begin{aligned} L_K(\theta) = \|\nu_{\bar{\lambda}}(\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}))\|_2^2 + \|\nu_{\bar{\lambda}}(\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}))\|_2^2 \\ + \|\nu_{\bar{\tau}}(\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}))\|_2^2 + \|\nu_{\underline{\tau}}(\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}))\|_2^2 + \|\nu_p\|_2^2. \end{aligned} \tag{3.22}$$

We also define the regression loss as follows

$$L_R = \|f_{\boldsymbol{\theta}}(\boldsymbol{\ell}) - J^\star\| + \|\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}) - \boldsymbol{\mu}^\star\|, \tag{3.23}$$

where the first term is the regression loss defined between $f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ and $J^\star$ over the neural network's outputs, and the second term is the regression loss between $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ and $\boldsymbol{\mu}^\star$ over calculated derivatives.

By combining the the regression loss defined in (3.23) and the KKT-related loss term (3.22), we can express the training loss for our proposed algorithm as follows

$$L(\boldsymbol{\theta}) = L_R(\boldsymbol{\theta}) + L_K(\boldsymbol{\theta}). \tag{3.24}$$

By minimizing the loss function in (3.24), we can get the trained model $f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ and use it to make predictions in an on-line way.

Note that the KKT-related loss term in (3.22) can be calculated for both labelled and unlabelled datasets. For the unlabelled dataset, denoted as $\mathcal{D}_{w/t}$, the data points in it only contain the input load, i.e., $\{\boldsymbol{\ell}^{(1)}, \boldsymbol{\ell}^{(2)}, \cdots, \boldsymbol{\ell}^{(N)}\}$. To calculate the KKT-related loss on $\mathcal{D}_{w/t}$ does not require any ground-truth labels. Since dual variables $\underline{\boldsymbol{\tau}}, \bar{\boldsymbol{\tau}}, \underline{\boldsymbol{\lambda}}$, and $\bar{\boldsymbol{\lambda}}$ can all be represented in terms of $\boldsymbol{\mu}$ by solving the $\ell 1$-minimization problem in (3.7) and using the expressions in (3.20), they can be obtained by learning $\boldsymbol{\mu}$ from the input load. Besides, as shown in the architecture of the ICNN in Fig. 3.2, we can interpret the layer before the output layer as the prediction of the power generation $\mathbf{p}^g$, from which we can get the value of the flow $\mathbf{p}^e$ using the equation (3.12e). Then we do not have to consider the violation degree in (3.21e). For calculating the rest of violation degrees in (3.21), all the information required can be obtained by learning from the load. Therefore, to construct the KKT-related loss on $\mathcal{D}_{w/t}$, we do not ask for ground-truth labels.

We summarize our proposed algorithm for training the ICNN in Table 3.2, and call it *Algorithm 3-2*. Although adding KKT-related loss term makes the loss function complex, we do not observe any numerical issues during the training process. In the simulations, the loss function always decreases and converges to a low level.

Algorithm 3-2: Algorithm for Training the ICNN
Inputs: Samples with labels $\mathcal{D}_w$
Inputs: Samples without labels $\mathcal{D}_{w/t}$
Inputs: Model to be trained $f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$
Parameters: $\mathbf{M}^B$, $\mathbf{M}^F$, $\bar{\mathbf{p}}^f$, $\bar{\mathbf{p}}^g$, $\underline{\mathbf{p}}^g$
1: Calculate regression loss term L_R on $\mathcal{D}_w$ using (3.23)
2: Calculate KKT conditions-related loss term L_K
on $\mathcal{D}_w$ and $\mathcal{D}_{w/t}$ using (3.22)
3: Minimize $L(\boldsymbol{\theta})$ in (3.24) to get the optimal parameter $\hat{\boldsymbol{\theta}}$
Outputs: Trained model $f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\ell})$

Table 3.2: Algorithm for training the ICNN.

3.5 Generalization

In this section we consider the generalization performance of our proposed method. By generalization, we mean the algorithm should perform well on test samples that were not seen during the training process. We adopt the standard method of analysis here: we assume that the neural network can be trained to zero error on the training samples, then we study the errors for testing samples [85, 86, 87]. We use the following definition as a shorthand:

Definition 3.4. *We say a neural network is well-trained if it achieves zero loss on the training data.*

Understanding the generalization properties is important because zero training error does not imply small test error. Consider the example given in Fig. 3.3. Suppose we are fitting a piece-wise linear function but only given two points that lie on a line and the training loss is the regression loss. There are infinitely many functions that pass through these points, implying that they have zero training error. Obviously, many of them can have large testing errors for other points on the line. This example also shows that it is not sufficient to just impose convexity or provide gradient information on their own. There are also infinitely many convex functions passing through the labeled points, and there are infinitely many functions with the right gradients at the given points. To constrain the class of functions to be learned, both convexity and the gradient information are needed.

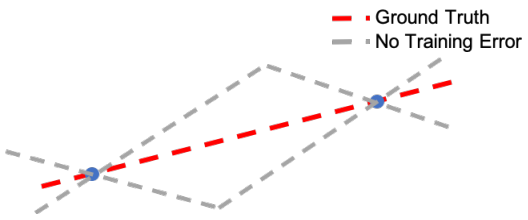


Figure 3.3: Given two points on a line, there are infinitely many piecewise linear functions passing through them. Therefore, small training error (passing through the points) does not necessary imply small generalization error (recovering the line).

Since $J^\star$ is piece-wise linear, we study generalization for two settings. The first is we assume that there are multiple training data within a region where $J^\star$ is linear, and we are

interested in the testing performance of a new input from the same region. This setting is about whether the model is constrained enough to not overfit during training. We provide a positive (and simple) answer in Theorem 3.5.

The second setting is to assume that the test data lies in a region that was unseen during the training process. This question is normally not asked since there is no expectation that learning would be useful for these type of unseen data. However, since there are large number of possible LP regions for DCOPF, it is likely that not all regions would be included in the training data. Therefore, learning algorithms must provide some guarantees on unseen regions. This is especially important as operating conditions change and the historical data are no longer reflective of future scenarios. In Theorem 3.6, we provide a positive answer showing that the gradients of the unseen region are still bounded.

3.5.1 Generalization for A Linear Region

In this part, we study the case where all training samples have the same value for $\boldsymbol{\mu}$. That is, they come from the same LP region. Let us denote the set of training samples as $\mathcal{D}_{trn}$, and $\operatorname{convhull} \mathcal{D}_{trn}$ is the convex hull of set $\mathcal{D}_{trn}$. The following theorem states the performance of the neural network for a new input $\boldsymbol{\ell}^{\text{new}}$ that is not in $\mathcal{D}_{trn}$.

Theorem 3.5. *Given N input loads $\mathcal{D}_{trn} = \{\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N\}$ and assume $\nabla_{\boldsymbol{\ell}} f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\ell}^i) = \boldsymbol{\mu}$ for all $i = 1, \ldots, N$. Assume the ICNN model $g_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ is well-trained on $\mathcal{D}_{trn}$. Then for all points $\boldsymbol{\ell}^{new} \in \operatorname{convhull} \mathcal{D}_{trn}$, $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new}) = \boldsymbol{\mu}$.*

This theorem is useful for LP problems because the gradient of J is piecewise constant over convex polytopic regions. If we are given some points from a region, then a well-trained neural network guarantees that the function is learned correctly for all points within their convex hull. This result implies the correctness of the overall algorithms since they only rely on the gradient (dual variable) information.

Technically, the above theorem says that if the gradients of a convex function are equal at a set of points, then the function is linear on the convex hull of these points. This is not a surprising result, but it does show that by constraining the structure of the neural network and using gradient information, we can generalize to uncountable number of points

(compact regions) by learning from a finite number of training points. The proof of the theorem is given in Appendix A.2.

3.5.2 Generalization with KKT Loss

If training samples may not represent all possible regions, we can construct the KKT conditions augmented loss in (3.22) on a set of unlabeled data points, for which we only have access to the input load values but not the optimal cost and the optimal LMPs. Since we do not ask for labels, we can sample as many data points as we want. This allows us to train with a very large set. We call this set of unlabeled data points the helper set. To construct the KKT loss on the helper set, we calculate the violations of KKT conditions through equations (3.21), where the Lagrange multipliers are determined from the learned $\boldsymbol{\mu}$ by following steps (3.5) and (3.6) in the training process.

By training with the helper set, unseen regions become "seen" in the sense that the outputs from the trained model must satisfy KKT conditions. As long as the model is well-trained, the analysis of generalization is the same as Section 3.5.1. Therefore, the generalization performance of training with helper set can also be guaranteed by Theorem 3.5.

3.5.3 Generalization for Unseen Regions

Here we consider the case where a region is not represented at all by the training data, including both labeled and helper sets. Then if a test sample comes from this region, would our method output anything useful? Methods like classification and dictionary learning cannot make useful predictions since there is no basis to make inferences about unseen regions. The next theorem shows that our approach is still partially successful because the gradient of a test data point is bounded by the gradient of the training points:

Theorem 3.6. *Given N input loads $\mathcal{D}_{trn} = \{\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N\}$, assume the ICNN model $g_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ is well-trained on $\mathcal{D}_{trn}$. Assume that $N \geqslant n+1$ and $\mathcal{D}_{trn}$ does not lie in a lower dimensional subspace in $\mathbb{R}^n$. Then for all points $\boldsymbol{\ell}^{new} \in \operatorname{convhull} \mathcal{D}_{trn}$, $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ is contained in a bounded convex polytope.*

The exact characterization of the polytope is given in the proof in Appendix A.3 and depends on the values of $\{\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N\}$ and $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$. The significance of the theorem lies in that training data are able to constrain the gradient for all points that lies in its convex hull. Intuitively speaking, as long as some surrounding points are included in training, the gradient cannot be "very wrong" even for points coming from LP regions that were not seen during training.

Consider the curve in Fig 3.1. Suppose we learned the two end pieces correctly but there was no training data for the middle piece, as shown in Fig. 3.4. Then by Theorem 3.6, the slope of the middle piece is constrained to be between the slope of the two end pieces. Furthermore, even if the neural network is trained in such a way that there are more than one piece of the middle region, the slopes of all of the pieces are still bounded between the two end pieces. Since Algorithm 3-1 only relies on getting $\boldsymbol{\mu}$ to be in the correct range, the active constraints would be identified correctly for all of these cases.

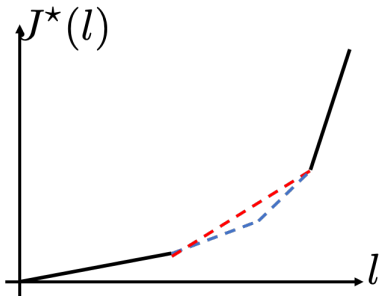


Figure 3.4: Example when the middle region has no data. But as long as the other two regions are well trained (black lines), the slopes in the middle region are bounded (blue and red dashed lines) by Theorem 3.6. Then the active constraint detection (Algorithm 3-1) would still be correct.

Theorem 3.6 formalizes the picture in Fig. 3.4 to higher dimensions, but the geometric intuition remains the same. This theorem also formalizes the empirical observation in [79], where the error of neural network-based OPF is reduced if training points are on the boundary of the feasible region.

Input variations	30%			50%		
Solutions	Optimality	Feasibility	Infeasibility	Optimality	Feasibility	Infeasibility
End-to-End	18.34	21.98	78.02	17.76	46.30	53.70
Our model	**94.93**	**94.93**	**5.07**	**93.08**	**93.08**	**6.92**

Table 3.3: Quality of solutions. We compare the solutions quality of our model to the end-to-end model. The ratios of optimal, feasible and infeasible solutions are listed and the numbers represent percentages. Results under two different input variations are given, i.e., 30% and 50% deviations from the nomial load value. More than 90% of the solutions obtained from our algorithm are optimal, while less than half of the solutions from the end-to-end model are feasible.

3.6 Simulation Studies

In this section, we demonstrate experimental results of using Algorithm 2 for training the ICNN and Algorithm 3-1 for solving DCOPF. We use the IEEE 14-bus system as the benchmark. We first examine the quality of solutions in terms of feasibility and optimality, then we examine the generalization performances. In Section 3.6.2, we show the simulation results of our model in the IEEE 118-bus system.

3.6.1 14-bus System

To generate the training set, we first sample $\boldsymbol{\ell}$ from the uniform distribution. We use two different variations from nominal load values: 30% and 50%. The total size of data samples is 50000 for each setting. Then, for each value of $\boldsymbol{\ell}$, we solve the primal and dual problems using CVXPY [88] powered by CVXOPT [89]. The optimal cost and the dual solutions are recorded. We hold 20% of all data samples as the test set and use the remaining for training. The ICNN we train has 4 hidden layers.

In order to better evaluate the performance of the model trained using our algorithm, we also provide experimental results of the end-to-end learning model, which is by far the most popular in the field of using deep learning for solving DCOPF [61, 62, 63, 64]. We constructed the end-to-end model with an architecture similar to [62], although other architectures do not change the conclusions. To be specific, for the end-to-end model, we

train a 4-layers fully-connected ReLU network by minimizing the regression loss, and use the trained model to directly predict the optimal solution to (3.1).

Both the end-to-end model and our model using Algorithm 2 are implemented on the Tensorflow platform and are trained until the training loss converges. The end-to-end model can be implemented using existing modules in Tensorflow, however, calculating KKT-related loss is specific for solving DCOPF and cannot be implemented using existing modules in Tensorflow. So we need to do some customization. In our experiments, the customized modules take longer time to train than existing functions or modules in Tensorflow. Therefore, the training time of our model for each epoch is longer than the end-to-end model.

Overall performance

To evaluate the quality of solutions obtained by different learning models, we divide solutions into three categories: optimal, feasible, and infeasible solutions. In our model, we feed $\boldsymbol{\ell}$ into the neural network and find the active constraints set. Then we solve a system of linear equations using a standard solver to obtain final solutions. The end-to-end model can directly outputs the solution to $\mathbf{p}^g$. The ratios of optimal, feasible and infeasible solutions obained by different learning models are listed in Table 3.3. As shown in Table 3.3, the optimality ratio of the solutions obtained from our model is higher than 90% under both input variations. In comparison, almost 50% of the solutions obtained from the end-to-end model are infeasible. In terms of computational time, both methods are much faster than iterative solvers in online DCOPF solving.

To examine the solution feasibility in the end-to-end model, we use the output $\mathbf{p}^g$ and the nodal power balance (3.1d) to obtain $\mathbf{p}^e$. The value of feasibility ratio depends on how large the error tolerance is. In this section, 0.3% mismatch is allowed when we calculate feasibility ratios. In Table 3.4, we also list the ratios of solutions that do not satisfy the nodal power balance, the limits on generators' outputs, and the limits on line flows, respectively. From Table 3.4, we can see that more than 98% of the solutions obtained from our model satisfy both generators limits and line flows limits. By contrast, only 24% of the solutions from the end-to-end model satisfy the line flow limits.

Input variations	30%		
Infeasibility	Nodal balance	Generators limits	Lines limits
End-to-End	16.88	0	76.79
Our model	**5.07**	**0**	**1.18**
Input variations	50%		
Infeasibility	Nodal balance	Generators limits	Lines limits
End-to-End	47.45	6.33	6.46
Our model	**6.92**	**0**	**0.79**

Table 3.4: Infeasibility of solutions. We compare our model to the end-to-end model. The ratios of solutions that do not satisfy the nodal power balance, generators' limits and the line limits are listed. 0.3% mismatch is allowed. We give results under two different input variations, i.e., 30% and 50%. In both settings, more than 98% of the solutions obtained from our model satisfy both generators limits and line flows limits, and more than 90% of our solutions satisfy the nodal power balance.

Generalization on Unseen Regions

To evaluate the generalization ability of our model, we create an illustrative example in the 14-bus system. Particularly, we examine the generalization performance on new data points that comes from region without any training samples. To generate the training set for this case, we only change the load values at two buses, but keep the remaining load values fixed. In this way, the space of input loads can be regarded as a two-dimensional plane. When varying the load values at the two buses, we can have four different combinations of active constraints, which correspond to four different values of $\boldsymbol{\mu}^\star$. Therefore, we divide the input load space as four regions, denoted as R_0, R_1, R_2 and R_3. The division of the input space is shown in Fig. 3.5. We take training samples from R_0, R_2, and R_3, and take testing samples from R_1. Let us denote the training set as $\mathcal{D}_{trn}$, and the testing set as $\mathcal{D}_{tst}$.

We use two different training approaches for our model. In the first training approach, we only use $\mathcal{D}_{trn}$ for training and minimize both the regression loss and the KKT-related loss on $\mathcal{D}_{trn}$. For the second training approach, we construct an additional training set,

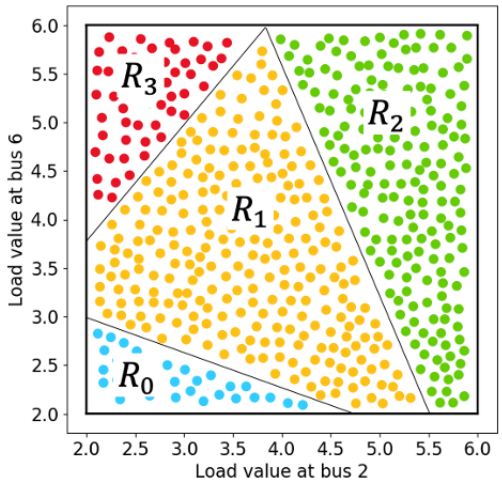


Figure 3.5: Division of the input space. The axes are load values at the two buses. In this example, based on different combinations of active constraints, the input load space can be divided into four regions. We take samples from R_1 as the testing set and samples from surrounding regions R_0, R_2 and R_3 as the training set.

called helper set $\mathcal{D}_{help}$, which only contains the input load, i.e., $\{\boldsymbol{\ell}^{(1)}, \boldsymbol{\ell}^{(2)}, \cdots, \boldsymbol{\ell}^{(N)}\}$, and do not have any ground-truth values. To generate the helper set, we can sample $\boldsymbol{\ell}$ from uniform distributions $\ell_2 \sim \mathsf{Uniform}(2, 6)$ and $\ell_6 \sim \mathsf{Uniform}(2, 6)$. Therefore, $\mathcal{D}_{help}$ contains the testing region R_1. Aside from minimizing (3.24) on $\mathcal{D}_{trn}$, we also minimize the KKT-augmented loss (3.22) on $\mathcal{D}_{help}$. [3] End-to-end model only use labeled samples and is trained on $\mathcal{D}_{trn}$.

We list the ratios of optimal, feasible and infeasible solutions obtained from different learning models in Table 3.5. As we can see, when we use $\mathcal{D}_{help}$ as an additional training set, we can obtain an optimality ratio as high as 96%. Even without $\mathcal{D}_{help}$, more than half of the solutions obtained from our model can achieve optimal values. As a comparison, the end-to-end model fails to make feasible predictions on test samples that come from never seen regions in the training process. The reason that our proposed algorithm outperforms the end-to-end model can be attributed to the KKT-related loss. By minimizing the KKT-

[3] We know from Section 3.4.2 that to calculate the KKT-augmented loss on $\mathcal{D}_{help}$ does not require ground-truth labels.

	Optimality	Feasibility
End-to-end	5.52	8.6
Our model, with $\mathcal{D}_{help}$	**97.24**	**97.24**
Our model, without $\mathcal{D}_{help}$	**62.25**	**72.31**

Table 3.5: Generalization performance on test samples coming from never seen regions. With the helper set, our method is optimal 97% of the time (62% without). The end-to-end model fails to make reasonable predictions.

related loss term, the trained model is able to learn the underlying KKT conditions in all four regions and make better predictions of $\boldsymbol{\mu}^\star$ on $\mathcal{D}_{tst}$.

3.6.2 118-bus System

In this part, we evaluate our model in the IEEE 118-bus system, where 54 generators and 183 edges are located, and compare with the classification approach. To generate the training set, we sample load values $\boldsymbol{\ell}$ from a Gaussian distribution with mean of 1.0 and standard deviation of 0.03. Then, for each value of $\boldsymbol{\ell}$, we solve the primal and dual problems using CVXPY powered by CVXOPT. We solve 60000 samples and split 20% of all data samples for testing and 80% for training. For our model, we train a 5-layer ICNN to learn the mapping from load values to cost function values. For the classification method, a 5-layer fully-connected neural network is constructed to predict the set of active constraints. We use one-hot encoding of different active sets as labels, and use cross-entropy as the loss function. The simulation results show that our model achieves an accuracy of 88.96% in terms of finding optimal solutions. In comparison, the accuracy for the classification task is 39.61%.

3.7 Conclusions

This chapter proposes a new framework to use neural networks for solving DCOPF. By leveraging rich linear programming theories, we prove our framework guarantees general-

ization. First, using the convexity of optimal cost in DCOPF, we constrain the neural network to have an input convex structure. Second, using the KKT optimality conditions, we add violations of KKT conditions to the training loss. In this way, we are able to exploit large amounts of unlabeled data points for training and improve the generalization performance. Our method is evaluated on the IEEE 14-bus and 118-bus. The experimental results demonstrate that our method significantly outperforms existing end-to-end and classification approaches.

Chapter 4

AN EFFICIENT LEARNING-BASED SOLVER FOR TWO-STAGE DC OPTIMAL POWER FLOW WITH FEASIBILITY GUARANTEES

4.1 Introduction

The optimal power flow (OPF) problem is one of the fundamental tools in the operation and planning of power systems [51, 52, 53]. It determines the minimum-cost generator outputs that meet the system demand and satisfy the power flow equations and operational limits on generators, line flows and other devices. Traditionally, the OPF is formulated as a deterministic optimization problem, where a solution is computed for some nominal and fixed demand. However, with significant penetration of renewable energy into the power grid as well as demand response programs, the fluctuation in the demand should be explicitly taken into account [90].

To take uncertainties in the net-load into account,[1] stochastic programming methods are a type of common tools used to to reformulate the OPF as a multi-stage problem [91, 92, 93]. In these problems, decisions are made sequentially at each stage, based on the forecast of the net-load and the fact that additional adjustments can be made in future stages when the uncertainties are better known.

In this chapter, we consider a two-stage stochastic program based on the DC optimal power flow (DCOPF) model. The DC power flow model linearizes the power flow equations and is the workhorse in power industries [54]. The two-stage DCOPF problem is also becoming increasingly popular as a canonical problem that incorporates the impact of uncertainties arising from renewable resources [94, 95]. In more general terms, the two-stage DCOPF problem falls under the category of *two-stage stochastic linear programs with (fixed) recourse* (2S-SLPR) [96].

[1] In this chapter, we use the term net-load to capture both renewable generation in the system [91] and the load.

Like other 2S-SLPR problems, the second stage of the two-stage DCOPF involves an expectation of the uncertain parameters (i.e., the randomness in the net-load) over some probability distribution. In practice, the probability distributions are rarely known and difficult to work with analytically. Therefore, several different approaches have been used to approximate 2S-SLPR problems. Among these, the most popular is the sample average approximation (SAA) [97, 98, 99].

The SAA is a basic Monte Carlo simulation method, which represents the random parameter using a finite set of realizations (scenarios), yielding a (possibly large) deterministic two-stage linear programming problem. Though the SAA approach is easy to implement, directly using it to solving two-stage DCOPF may result in computational challenges. A reason for this is that the SAA method tends to require a large sample set in order to generate a good-quality solution [100, 101, 102], rendering the SAA formulation for two-stage DCOPF into a very large-scale linear program. In some sense, the challenge has moved from generating many high quality samples from a probabilistic forecast to being able to solve an optimization problem using these samples [103, 61, 104]. Secondly, as decisions in power system operations are made in a more online (or corrective) manner [95, 105], OPF problems need to be solved repeatedly in real time. Even though solving single linear programs is easy, solving two-stage DCOPF problems are not [106, 61].

A common approach to reduce the computational burden in solving two-stage DCOPFs is to model the second stage decisions using an affine policy. More specifically, the second-stage (or the recourse) dispatch decision is restricted to be an affine function of the realized net-load and the first-stage decisions [107, 108, 109]. Once the affine policy is determined, the decision-making in the real time is just simple function evaluations. This method has been observed to provide good performance when the net-load variations are small or are restricted to a few possible instances [110, 111, 112]. However, if the variations are large or include many possible values, the affine policy method tends to not perform well. In fact, it may produce decisions that do not even satisfy the constraints in the two-stage optimization problem.

In this chapter, we overcome the challenge in policy design and solving two-stage DCOPF problems by presenting a neural network (NN)-based architecture that is computationally

efficient and also guarantees the feasibility of learned solutions. In particular, our architecture involves two neural networks, one each for the first and second stages. The first neural network learns the mapping from the load forecast to the first-stage decisions. The second neural network approximates the cost-to-go given the net-load realization and the learned first-stage decisions. So, instead of using the affine policy, we offer an NN-based policy to solve the second-stage OPF problem.

Recently, many machine-learning based algorithms have been proposed to accelerate the computational speed for OPF, please see [61, 65, 113, 114, 115] and the references within. However, many of the existing algorithms are not suitable in a two-stage problem. To enhance computational speed, the proxy of the second-stage cannot involve multiple steps (e.g., a neural network followed by a power flow solver. For feasibility, the solutions need to have feasibility guarantees. In this chapter, we use an NN policy that is constructed using a technique called the *gauge map* [116, 117, 118], which allows the output of the NN to be guaranteed to satisfy the DCOPF constraints. Since this policy also involves only function evaluations, it preserves the speed of affine policies. At the same time, a neural network is much more expressive than an affine function, and can provide much better approximations to the true solution.

The rest of this chapter is organized as follows: In Section 4.2.3, we describe the general setup of the two-stage DCOPF problem and the two widely-used formulations of two-stage DCOPF. Section 4.3 presents the proposed learning approach to solving the two-stage DCOPF problem, including the overall architecture design, the training of it and the decision-making procedure. Section 4.4.2 illustrates how to incorporate the gauge map technique into the architecture design to ensure the feasibility of the neural networks' predictions. Section 4.5.2 provides the simulation results and Section 4.6 concludes the chapter.

4.2 Two-stage DCOPF Model

In this section, we provide more details about the formulation of two-stage DCOPF problems. Consider a power network with n buses connected by m transmission lines. Without loss of generality, we assume each bus i has a generator as well as a load, and the load is uncertain. We denote the randomness in the system by $\boldsymbol{\omega} \in \mathbb{R}^n$, which is a random vector,

and the net-load at each bus i is a function of $\boldsymbol{\omega}$, denoted by $\ell_i(\boldsymbol{\omega})$. Note that this notation allows us to capture the fact that the load depend non-trivially on the underlying randomness. The algorithms developed in this chapter is compatible with any scenario-based forecasting algorithms.

In the first stage of a problem, the exact value of $\ell_i(\boldsymbol{\omega})$ is not known. Rather, we assume a forecast is available. Specifically, we adopt a scenario-based probabilistic load forecasting framework in this chapter and assume a set of samples (scenarios) that are representative of $\boldsymbol{\omega}$ are available [18, 119, 34, 30, 120]. It is useful to assume that a nominal load–for example, the mean of $\ell_i(\boldsymbol{\omega})$–is known in the first stage. We denote this nominal load by $\bar{\ell}_i$, and based on the scenario forecasts and $\bar{\ell}_i$, the system operator (SO) chooses a first-stage generation dispatching decision, denoted by p_i^I. Then once the actual demand $\ell_i(\boldsymbol{\omega})$ is realized, a second-stage (recourse) decision p_i^R is determined to balance the power network.

For concreteness, we specifically consider two widely used formulations of the two-stage DCOPF problem, risk-limiting dispatch (RLD) [91] and reserve scheduling [111]. Both are two-stage stochastic linear programs with recourse, and both highlight the structure and difficulty of two-stage problems.

4.2.1 Risk-Limiting Dispatch

The RLD problem seeks to find a first-stage dispatching decision p_i^I at each bus i that minimizes expected total cost in two stages. The second-stage decisions, p_i^R, are made after the net-load is observed.

We assume that the cost of dispatching generation at bus i is $c_i p_i^I$ in the first stage and $q_i[p_i^R]^+$ in the second stage, where c_i and q_i are prices measured in dollars per MW ($/MW) and the notation $[z]^+ = \max\{z, 0\}$ means that only power purchasing ($p_i^R > 0$) incurs a second-stage cost and any excess power ($p_i^R < 0$) can be disposed of for free [91, 121, 122]. The cost minimization problem is:

$$J^\star_{\text{rld}}(\bar{\boldsymbol{\ell}}) \triangleq \min_{\mathbf{p}^I} \quad \mathbf{c}^T\mathbf{p}^I + \mathbb{E}[Q(\boldsymbol{\ell}(\boldsymbol{\omega}) - \mathbf{p}^I; \mathbf{q})|\bar{\boldsymbol{\ell}}] \tag{4.1a}$$

$$\text{s.t.} \quad \mathbf{p}^I \geqslant \mathbf{0}, \tag{4.1b}$$

where the expectation is taken with respect to the probability distribution of $\boldsymbol{\ell}(\boldsymbol{\omega})$ conditioned on $\bar{\boldsymbol{\ell}}$, and $Q(\boldsymbol{\ell}(\boldsymbol{\omega})-\mathbf{p}^I;\mathbf{q})$ is the second-stage cost or cost-to-go. Given the first-stage decision $\mathbf{p}^I = [p_1^I,\cdots,p_n^I]^T$ and a particular realization of $\boldsymbol{\ell}(\boldsymbol{\omega})$, the second-stage cost is given by

$$
\begin{aligned}
Q(\boldsymbol{\ell}(\boldsymbol{\omega})-\mathbf{p}^I;\mathbf{q}) \triangleq \min_{\mathbf{p}^R,\boldsymbol{\delta}^{ns}} \quad & \mathbf{q}^T[\mathbf{p}^R]^+ && \text{(4.2a)}\\
\text{s.t.} \quad & \mathbf{B}\boldsymbol{\delta}^{ns} = \mathbf{p}^R - (\boldsymbol{\ell}(\boldsymbol{\omega})-\mathbf{p}^I) && \text{(4.2b)}\\
& -\bar{\mathbf{p}}^f \leqslant \mathbf{F}\boldsymbol{\delta}^{ns} \leqslant \bar{\mathbf{p}}^f, && \text{(4.2c)}\\
& \underline{\boldsymbol{\delta}} \leqslant \mathbf{E}\boldsymbol{\delta}^{ns} \leqslant \bar{\boldsymbol{\delta}}, && \text{(4.2d)}
\end{aligned}
$$

where (4.2b) is the DC power flow constraints, (4.2c) is the line flow limit constraints and (4.2d) is angle difference limit constraints. Without loss of generality, we assume bus 1 is the reference (slack) node and set its voltage angle to be zero. The notation $\boldsymbol{\delta}^{ns} \in \mathbb{R}^{n-1}$ denotes the voltage angles at non-slack buses, the matrix $\mathbf{B} \in \mathbb{R}^{n\times(n-1)}$ maps $\boldsymbol{\delta}^{ns}$ to the nodal power injections, the matrix $\mathbf{F} \in \mathbb{R}^{m\times(n-1)}$ maps $\boldsymbol{\delta}^{ns}$ to the flows on all edges, and the matrix $\mathbf{E} \in \mathbb{R}^{m\times(n-1)}$ is the incidence matrix of the network graph. See Appendix B.1 for details on constructing $\mathbf{B}$, $\mathbf{F}$ and $\mathbf{E}$.

Note that the second-stage problem (4.2) can be seen as a deterministic DCOPF problem with the demand $\boldsymbol{\ell}(\boldsymbol{\omega})-\mathbf{p}^I$. Since the recourse decision $\mathbf{p}^R$ is not bounded, (4.2) is feasible for any given demand input.

We approximate the expectation in (4.1) using samples. Let $\{\boldsymbol{\omega}^k\}_{k=1}^K$ be a collection of samples of $\boldsymbol{\omega}$, and $\{\boldsymbol{\ell}(\boldsymbol{\omega}^k)\}_{k=1}^K$ be the collection of load realizations. We determine the first-stage decision by solving the following scenario-based problem that is a deterministic

linear program:

$$\widetilde{J}^K_{\rm rld}(\bar{\boldsymbol{\ell}}) \triangleq \min_{\substack{\mathbf{p}^I,\\ \{\mathbf{p}^R(\boldsymbol{\omega}^k),\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k)\}_{k=1}^K}} \quad \mathbf{c}^T\mathbf{p}^I + \frac{1}{K}\sum_{k=1}^K \mathbf{q}^T[\mathbf{p}^R(\boldsymbol{\omega}^k)]^+ \tag{4.3a}$$

$$\text{s.t.} \quad \mathbf{p}^I \geqslant \mathbf{0} \tag{4.3b}$$

$$\mathbf{B}\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k) = \mathbf{p}^R(\boldsymbol{\omega}^k) - (\boldsymbol{\ell}(\boldsymbol{\omega}^k) - \mathbf{p}^I),\ \forall k \tag{4.3c}$$

$$-\bar{\mathbf{p}}^f \leqslant \mathbf{F}\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k) \leqslant \bar{\mathbf{p}}^f, \forall k \tag{4.3d}$$

$$\underline{\boldsymbol{\delta}} \leqslant \mathbf{E}\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k) \leqslant \bar{\boldsymbol{\delta}}, \forall k \tag{4.3e}$$

where the second-stage decisions $\{\mathbf{p}^R(\boldsymbol{\omega}^k), \boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k)\}_{k=1}^K$ are functions of $\boldsymbol{\omega}$ and the constraints (4.3c)-(4.3e) related to the second-stage decisions need to be satisfied for every load realization $\boldsymbol{\ell}(\boldsymbol{\omega}^k)$.

4.2.2 Two-stage DCOPF with Reserve

Sometimes the second-stage recourse decision $\mathbf{p}^R$ cannot be arbitrarily positive or negative. Instead, $\mathbf{p}^R$ is limited by various factors such as generator capacities or real-time (second-stage) electricity market structure. This is captured by a two-stage DCOPF where reserve services are provided to deal with the possible mismatch between the prescheduled generation and the realized load [95, 111].

Specifically, we consider the spinning reserve service in this chapter. In the first stage, in addition to choosing an initial dispatching decision p_i^I at each bus i, the SO also needs to decide the up and down reserve capacities, denoted by $\widehat{r}_i$ and $\breve{r}_i$, respectively. In this way, the first-stage cost at each bus i includes both the cost of dispatching p_i^I, i.e., $c_i p_i^I$, and of providing reserve services that is given by $\gamma_i(\widehat{r}_i + \breve{r}_i)$, where c_i and γ_i are prices measured in \$/MW.

The second-stage recourse decision p_i^R at each bus i is constrained by the reserve capacities, $\widehat{r}_i$ and $\breve{r}_i$. To quantify the amount by which the reserve capacities decided in the first stage might be exceeded, we define the cost of dispatching p_i^R at each bus i as a piecewise-affine function given by $q_i^{\rm res}\Big([p_i^R - \widehat{r}_i]^+ - [p_i^R + \breve{r}_i]^-\Big)$, where $q_i^{\rm res}$ is penalty cost in \$/MW and $[z]^- = \min\{z, 0\}$. This cost function means that there would be no cost for

second-stage dispatching within the reserve amounts that are allocated in the first stage.

The two-stage DCOPF with reserve scheduling can be formulated as the following stochastic program:

$$
\begin{aligned}
& J^{\star}_{\text{res}}(\bar{\boldsymbol{\ell}}) \triangleq \\
\min_{\substack{\mathbf{p}^I, \\ \widehat{\mathbf{r}}, \check{\mathbf{r}}}} \quad & \mathbf{c}^T \mathbf{p}^I + \boldsymbol{\gamma}^T (\widehat{\mathbf{r}} + \check{\mathbf{r}}) + \mathbb{E}[Q(\boldsymbol{\ell}(\boldsymbol{\omega}) - \mathbf{p}^I; \widehat{\mathbf{r}}, \check{\mathbf{r}}, \mathbf{q}^{\text{res}}) | \bar{\boldsymbol{\ell}}] && (4.4a) \\
\text{s.t.} \quad & \mathbf{0} \leqslant \mathbf{p}^I \leqslant \bar{\mathbf{p}}^g && (4.4b) \\
& \mathbf{p}^I + \widehat{\mathbf{r}} \leqslant \bar{\mathbf{p}}^g && (4.4c) \\
& \mathbf{p}^I - \check{\mathbf{r}} \geqslant \mathbf{0} && (4.4d) \\
& \widehat{\mathbf{r}},\ \check{\mathbf{r}} \geqslant \mathbf{0}, && (4.4e)
\end{aligned}
$$

where (4.4c)-(4.4e) constrain the up and down reserve at each bus i to be positive and no larger than the available capacities around p_i^I. Given the first-stage decisions $(\mathbf{p}^I, \widehat{\mathbf{r}}, \check{\mathbf{r}})$ and a particular realization of $\boldsymbol{\ell}(\boldsymbol{\omega})$, the second-stage cost is given by

$$
\begin{aligned}
& Q(\boldsymbol{\ell}(\boldsymbol{\omega}) - \mathbf{p}^I; \widehat{\mathbf{r}}, \check{\mathbf{r}}, \mathbf{q}^{\text{res}}) \triangleq \\
\min_{\mathbf{p}^R, \boldsymbol{\delta}^{ns}} \quad & \mathbf{q}^{\text{res}T} \Big([\mathbf{p}^R - \widehat{\mathbf{r}}]^+ - [\mathbf{p}^R + \check{\mathbf{r}}]^- \Big) && (4.5a) \\
\text{s.t.} \quad & \mathbf{B}\boldsymbol{\delta}^{ns} = \mathbf{p}^R - (\boldsymbol{\ell}(\boldsymbol{\omega}) - \mathbf{p}^I) && (4.5b) \\
& -\bar{\mathbf{p}}^f \leqslant \mathbf{F}\boldsymbol{\delta}^{ns} \leqslant \bar{\mathbf{p}}^f && (4.5c) \\
& \underline{\boldsymbol{\delta}} \leqslant \mathbf{E}\boldsymbol{\delta}^{ns} \leqslant \bar{\boldsymbol{\delta}}, && (4.5d)
\end{aligned}
$$

which can also be seen as a deterministic DCOPF problem with demands $\boldsymbol{\ell}(\boldsymbol{\omega}) - \mathbf{p}^I$ and the cost being the penalty imposed on the generation amount if it exceeds the reserve capacities. This "penalizing deviations" technique is commonly employed by stochastic programmers to promote the feasibility of second-stage problems for any given first-stage decisions [123].

The SAA method solves the following scenario-based problem associated with (4.4):

$$\widetilde{J}^K_{\text{res}}(\bar{\boldsymbol{\ell}}) \triangleq \min_{\substack{\mathbf{p}^I,\widehat{\mathbf{r}},\check{\mathbf{r}},\\ \{\mathbf{p}^R(\boldsymbol{\omega}^k),\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k)\}_{k=1}^K}} \mathbf{c}^T\mathbf{p}^I + \boldsymbol{\gamma}^T(\widehat{\mathbf{r}}+\check{\mathbf{r}}) + \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{q}^{\text{res}T}\left([\mathbf{p}^R(\boldsymbol{\omega}^k)-\widehat{\mathbf{r}}]^+ - [\mathbf{p}^R(\boldsymbol{\omega}^k)+\check{\mathbf{r}}]^-\right)\right) \tag{4.6a}$$

$$\text{s.t.} \quad \mathbf{0} \leqslant \mathbf{p}^I \leqslant \bar{\mathbf{p}}^g \tag{4.6b}$$

$$\mathbf{p}^I + \widehat{\mathbf{r}} \leqslant \bar{\mathbf{p}}^g \tag{4.6c}$$

$$\mathbf{p}^I - \check{\mathbf{r}} \geqslant \mathbf{0} \tag{4.6d}$$

$$\widehat{\mathbf{r}},\ \check{\mathbf{r}} \geqslant \mathbf{0} \tag{4.6e}$$

$$\mathbf{B}\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k) = \mathbf{p}^R(\boldsymbol{\omega}^k) - (\boldsymbol{\ell}(\boldsymbol{\omega}^k) - \mathbf{p}^I), \forall k \tag{4.6f}$$

$$-\bar{\mathbf{p}}^f \leqslant \mathbf{F}\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k) \leqslant \bar{\mathbf{p}}^f, \forall k \tag{4.6g}$$

$$\underline{\boldsymbol{\delta}} \leqslant \mathbf{E}\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k) \leqslant \bar{\boldsymbol{\delta}}, \forall k. \tag{4.6h}$$

4.2.3 Computational Challenges

To have a sample set of load realizations that is representative enough of the true distribution of the random net-load, a large number of realizations are required for even a moderately sized system [124]. Therefore, although (4.3) and (4.6) are linear programs, they are often large-scale problems. In addition, since both the first and second-stage decisions depend on the mean of the scenario forecasts, $\bar{\boldsymbol{\ell}}$, every time the set of scenarios changes, we need to re-solve (4.3) and (4.6). Even if a single instance can be solved efficiently using commercial solvers such as CVXPY [88, 125] and GLPK [126], repeatedly solving large-scale linear programs can still impose considerable computational burdens.

The scale of the problems can grow quickly as the size of the system and the number of scenarios grow. Therefore, an affine policy is often used to approximate (4.2) and (4.5). However, finding a good policy that satisfies the constraints ((4.3c),(4.3d), (4.3e), (4.6f), (4.6g), and (4.6h)) can be difficult. In the next section, we present an NN-based learning architecture to enable more efficient computation.

4.3 Proposed Learning Algorithm

In this section, we present the learning algorithm to solve the scenario-based problems in (4.3) and (4.6). To start with, we rewrite the two-stage problem in a more compact way as follows

$$\widetilde{J}^K(\bar{\boldsymbol{\ell}}) \triangleq \min_{\mathbf{x}} \quad \widetilde{c}(\mathbf{x}) + \widetilde{Q}^K(\mathbf{x}; \{\boldsymbol{\omega}^k\}_{k=1}^K, \widetilde{\mathbf{q}}) \tag{4.7a}$$

$$\text{s.t.} \quad \mathbf{x} \in \mathcal{X} \tag{4.7b}$$

where $\mathbf{x}$ denotes the first-stage decisions, which is $\mathbf{p}^I$ for (4.3) and $(\mathbf{p}^I, \widehat{\mathbf{r}}, \check{\mathbf{r}})$ for (4.6), and the set $\mathcal{X}$ collects all constraints that $\mathbf{x}$ has to satisfy, i.e., (4.3b) or (4.6b)-(4.6e). The notation $\widetilde{c}(\cdot)$ is the generic representation of the first-stage cost and $\widetilde{Q}^K(\mathbf{x}; \{\boldsymbol{\omega}^k\}_{k=1}^K, \widetilde{\mathbf{q}})$ is the estimated second-stage cost based on a collection of scenarios $\{\boldsymbol{\ell}(\boldsymbol{\omega}^k)\}_{k=1}^K$.

Here we use a simple decomposition technique such that (4.7) becomes much easier to work with. To be specific, if the first-stage decision $\mathbf{x}$ is taken as given, then the second-stage cost $\widetilde{Q}^K(\mathbf{x}; \{\boldsymbol{\omega}^k\}_{k=1}^K, \widetilde{\mathbf{q}})$ is separable:

$$\widetilde{Q}^K(\mathbf{x}; \{\boldsymbol{\omega}^k\}_{k=1}^K, \widetilde{\mathbf{q}}) = \frac{1}{K} \sum_{k=1}^K \widetilde{Q}^k(s_\ell(\mathbf{x}; \boldsymbol{\omega}^k); \mathbf{x}, \widetilde{\mathbf{q}})$$

where we use the notation $s_\ell(\mathbf{x}; \boldsymbol{\omega}^k) \triangleq \boldsymbol{\ell}(\boldsymbol{\omega}^k) - \mathbf{p}^I$ to represent the demands that are not balanced by the first stage when the load realization is actually $\boldsymbol{\ell}(\boldsymbol{\omega}^k)$, and $\widetilde{Q}^k(s_\ell(\mathbf{x}; \boldsymbol{\omega}^k); \mathbf{x}, \widetilde{\mathbf{q}})$ is the optimal value of each scenario problem for a particular load realization $\boldsymbol{\ell}(\boldsymbol{\omega}^k)$. Each of these scenario problems can be seen as a deterministic DCOPF problem with demands $s_\ell(\mathbf{x}; \boldsymbol{\omega}^k)$ and an objective function $q^R(\cdot; \mathbf{x}, \widetilde{\mathbf{q}})$ that takes $\mathbf{x}$ as given and $\widetilde{\mathbf{q}}$ as parameters. The deterministic DCOPF problem can be written in the following generic form:

$$\widetilde{Q}(s_\ell(\mathbf{x}; \boldsymbol{\omega}); \mathbf{x}, \widetilde{\mathbf{q}}) \triangleq \min_{\mathbf{p}^R, \boldsymbol{\delta}^{ns}} \quad q^R(\mathbf{p}^R; \mathbf{x}, \widetilde{\mathbf{q}}) \tag{4.8a}$$

$$\text{s.t.} \quad \mathbf{B}\boldsymbol{\delta}^{ns} = \mathbf{p}^R - s_\ell(\mathbf{x}; \boldsymbol{\omega}) \tag{4.8b}$$

$$-\bar{\mathbf{p}}^f \leqslant \mathbf{F}\boldsymbol{\delta}^{ns} \leqslant \bar{\mathbf{p}}^f \tag{4.8c}$$

$$\underline{\boldsymbol{\delta}} \leqslant \mathbf{E}\boldsymbol{\delta}^{ns} \leqslant \bar{\boldsymbol{\delta}}. \tag{4.8d}$$

In a similar fashion, by exploiting the decomposable structure of (4.7), the proposed learning algorithm consists of two subnetworks, denoted by f^I and f^R, respectively. The

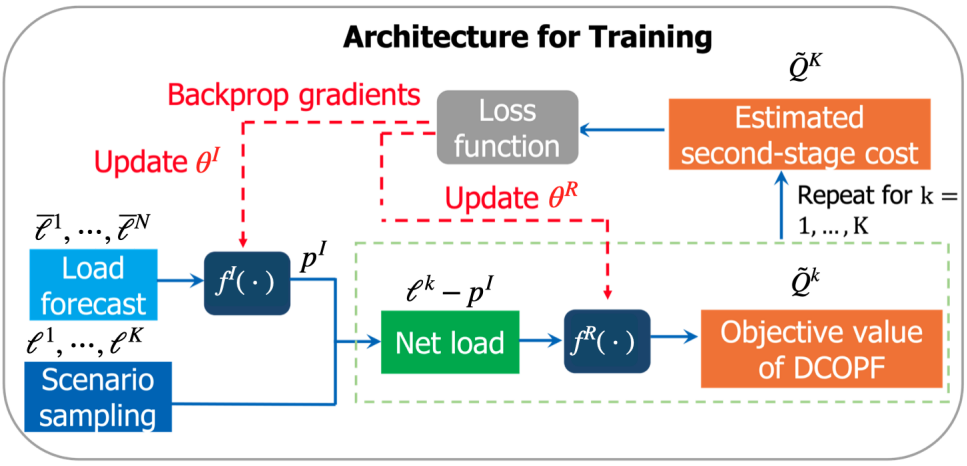


Figure 4.1: The architecture used for training in the proposed algorithm. When making decisions in real time, we only need the network f^I to predict the first-stage decisions from the given load forecast.

first subnetwork f^I learns the mapping from $\bar{\boldsymbol{\ell}}$ to $\mathbf{x}$, i.e., $f^I : \mathbb{R}^n_+ \longmapsto \mathcal{X}$, while the second one learns the mapping from $s_\ell(\mathbf{x}; \boldsymbol{\omega})$ to $\widetilde{Q}(s_\ell; \mathbf{x}, \tilde{\mathbf{q}})$ considering $\mathbf{x}$ as given, i.e., $f^R : s_\ell(\mathbf{x}, \boldsymbol{\omega})|\mathcal{X} \longmapsto [0, +\infty]$, where $s_\ell(\mathbf{x}, \boldsymbol{\omega})$ is the set of all possible mismatches, i.e., $s_\ell(\mathbf{x}, \boldsymbol{\omega}) = \{\boldsymbol{\ell}(\boldsymbol{\omega}) - \mathbf{p}^I | (\mathbf{x}, \boldsymbol{\omega}) \in \mathcal{X} \times \Omega\}$, and Ω is the sample space of $\boldsymbol{\omega}$. Note that the second network f^R takes two inputs. The first input is the unbalanced demand $s_\ell(\mathbf{x}; \boldsymbol{\omega}^k)$ when the load realization is $\boldsymbol{\ell}(\boldsymbol{\omega}^k)$. The second input is the first stage decision $\mathbf{x}$, which is treated as a given value. For the sake of analysis, we introduce an augmented vector $\tilde{s}_\ell$ that includes both inputs, specifically: $\tilde{s}_\ell = (s_\ell(\mathbf{x}; \boldsymbol{\omega}^k), \mathbf{x})$.

The two subnetworks can be implemented using neural networks. Once trained, these neural networks can produce solutions much faster than existing solvers. However, a key question also arises: how to make neural networks satisfy constraints, namely, how to ensure the output from f^I lies within the feasibility set $\mathcal{X}$ and the constraints of the optimization problem in (4.8) are satisfied? Notably, we want to avoid steps such as projecting to the feasible set since these introduce additional optimization problems [127], which somewhat defeats the purpose of learning. This key question will be tackled in the next section, and for the rest of this section, we first treat the two subnetworks as black boxes to provide an

overview of the proposed algorithm.

This algorithm includes a training process and a prediction process. The architecture used for training is shown in Fig. 4.1. When the learning algorithm is used in practice, i.e., in the prediction process, just the network f^I is required to predict the first-stage decisions from a scenario forecast. The reason why we need a second network f^R is that the two networks need to be trained together in order to obtain a network f^I that can predict the solution to (4.7) accurately. We now describe how the two networks in Fig. 4.1 are trained.

Let $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ denote the respective parameters, i.e., the weights and biases, of neural networks f^I and f^R. The goal of training is to learn the optimal values for $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ that minimize a specifically designed loss function. Unlike standard machine learning practices, where the loss function is based on prediction errors using generated ground truth data, our model is trained in an unsupervised manner, which means the training dataset does not incorporate ground truth information. Instead, the parameters of f^I and f^R are updated to minimize the empirical mean of the total cost associated with the two-stage problem.

Concretely, given a batch of training data consisting of N load forecasts, denoted by $\{\overline{\boldsymbol{\ell}}^i\}_{i=1}^N$, the training loss function for our model is formulated as follows:

$$\min_{\boldsymbol{\theta}^I, \boldsymbol{\theta}^R} L(\boldsymbol{\theta}^I, \boldsymbol{\theta}^R) \triangleq \frac{1}{I} \sum_{i=1}^{N} L^i(\boldsymbol{\theta}^I, \boldsymbol{\theta}^R), \tag{4.9}$$

where

$$L^i(\boldsymbol{\theta}^I, \boldsymbol{\theta}^R) \triangleq \widetilde{c}\Big(f^I(\overline{\boldsymbol{\ell}}^i; \boldsymbol{\theta}^I)\Big) + \frac{1}{K} \sum_{k=1}^{K} f^R\Big(\tilde{s}_\ell^{ik}; \boldsymbol{\theta}^R\Big).$$

We use the double superscript in $\tilde{s}_\ell^{ik}$ to represent that, for each instance of $\overline{\boldsymbol{\ell}}^i$, we need to sample an independent set of scenarios $\{\boldsymbol{\omega}^{ik}\}_{k=1}^K$. Note that the loss function in (4.9) represents exactly the average total cost of the two-stage DCOPF problem across the training dataset. In particular, the parameters of the first network f^I are updated by jointly minimizing the first stage cost and the anticipated cost-to-go associated with its predictions. For the second network f^R, its parameters are updated to minimize the recourse cost across a collection of sampled load realizations. This is because, given a first stage decision $\mathbf{x}$, the best decision that f^R can make is actually the minimal cost one.

After formulating the loss function (4.9) during the forward pass, the gradients of this loss function with respect to $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ are calculated through the backward pass. Following that, the stochastic gradient descent (SGD) method is used to minimize the loss function. The formulas for updating $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ using SGD can be found in Appendix B.2.

Once parameters $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ reach a local minimum and the training process terminates, we can use the trained network f^I that is parameterized by the learned $\boldsymbol{\theta}^I$ to predict the first-stage decisions from load forecasts. We summarize our learning algorithm, including the training and the decision-making procedures, in Table 7.1.

In the next section, we show the detailed architecture design of the two networks and answer the key question about how to make them satisfy the constraints in the optimization problems.

4.4 Network Architecture Design

In this section, we show the network design of f^I and f^R that ensures the feasibility of the networks' outputs. Particularly, each network consists of a sequence of neural layers, which are convolutional or fully connected layers with an activation function applied after each layer, then followed by a series of transformations that map the output of the neural layers to a feasible solution.

We first deal with first-stage constraints that must be satisfied by f^I. These constraints, as given in (4.3b) or (4.6b)-(4.6e), describe axis-aligned rectangular regions and are easy to satisfy. We will then deal with the second-stage constraints that f^R must satisfy. To be specific, f^R is learning the optimal value of the second-stage DCOPF problem and must satisfy all the constraints in the optimization problem; otherwise, the estimated second-stage cost may have a large deviation from the true value and mislead the training of f^I. In turn, if the first-stage decisions are poorly made, the resulting second-stage cost can be potentially very high when the uncertainties are realized. In particular, the constraints in the second-stage problem describe a high-dimensional polyhedral set that is dependent on the input data; thus, guaranteeing feasibility requires some more nontrivial techniques.

Proposed Learning Algorithm

Training Procedure

Inputs: Number of iterations T, a minibatch of training data, $\{\bar{\boldsymbol{\ell}}\}_{i=1}^{I}$, sample space Ω of $\boldsymbol{\omega}$

Parameters: $\boldsymbol{\theta}^I$, $\boldsymbol{\theta}^R$

for $t = 1, \cdots, T$ **do**

  Randomly sample $\{\boldsymbol{\omega}^{ik}\}_{i=1:I,k=1:K}$ from Ω.

  Forward pass $f^I\Big(\{\bar{\boldsymbol{\ell}}\}_{i=1}^{I}; \boldsymbol{\theta}^I\Big) \longrightarrow \{\mathbf{x}^i\}_{i=1}^{I}$

  for $k = 1, \cdots, K$ **do**

    Calculate $s_\ell(\mathbf{x}^i, \boldsymbol{\omega}^{ik})$ for $i = 1, \cdots, I$.

    Forward pass $f^R\Big(\{s_\ell(\mathbf{x}^i, \boldsymbol{\omega}^{ik})\}_{i=1}^{I}; \boldsymbol{\theta}^R\Big) \longrightarrow \Big\{\widetilde{Q}^k(s_\ell(\mathbf{x}^i; \boldsymbol{\omega}^{ik}); \mathbf{x}^i, \tilde{\mathbf{q}})\Big\}_{i=1}^{I}$

  end for

  Construct the loss function using (4.9).

  Randomly pick a data point $\bar{\boldsymbol{\ell}}^i$ and calculate the stochastic gradients using (B.1).

  Update $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ using (B.2).

  Check stopping criterion.

end for

Outputs: Trained networks f^I and f^R

Decision Making Procedure

Inputs: Load forecast $\bar{\boldsymbol{\ell}}^{\text{new}}$, trained network f^I

Forward pass $f^I\Big(\bar{\boldsymbol{\ell}}^{\text{new}}; \boldsymbol{\theta}^I\Big) \longrightarrow \mathbf{x}^{\text{new}}$

Outputs: Predicted first-stage decision $\mathbf{x}^{\text{new}}$

Table 4.1: The proposed learning-based algorithm to solve (4.7).

4.4.1 Design of the first-stage network: f^I

The network f^I in the RLD formulation must satisfy the non-negative orthant constraints in (4.3b), which can be guaranteed by using a ReLU activation after the last neural layer,

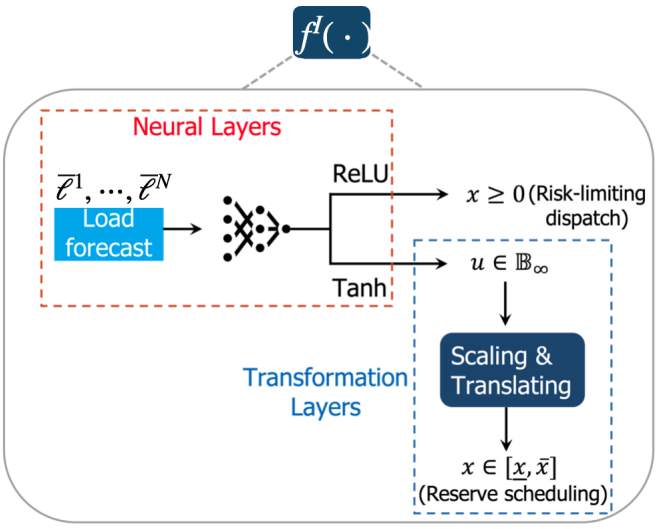


Figure 4.2: In the RLD problem, non-negative orthant constraints can be enforced using ReLU activation in the last neural layer. For the reserve scheduling problem, the Tanh activation is used at the last neural layer and then the hypercubic output is passed through the transformation layers in (4.10) to obtain a feasible solution.

and no additional transformation is needed.[2] For the reserve scheduling problem, we can rewrite the constraints in (4.6b)-(4.6e) in a more compact way as $\mathbf{x} \in [\bar{\mathbf{x}}, \underline{\mathbf{x}}]$, where $\bar{\mathbf{x}} = [\bar{\mathbf{p}}^{gT}, (\bar{\mathbf{p}}^g - \mathbf{p}^I)^T, \mathbf{p}^{IT}]^T$ and $\underline{\mathbf{x}} = [\mathbf{0}^T, \mathbf{0}^T, \mathbf{0}^T]^T$. In this way, the constraints in (4.6b)-(4.6e) can be treated as axis-aligned rectangular constraints.

To enforce such axis-aligned rectangular constraints, we start by using a Tanh activation on the last neural layer and denote its output as $\mathbf{u}$. Since the tanh function has a range between -1 and 1, we have $\mathbf{u} \in \mathbb{B}_\infty$, where $\mathbb{B}_\infty$ is the unit ball with ℓ_∞-norm given by $\mathbb{B}_\infty \triangleq \{\mathbf{z} \in \mathbb{R}^n | -1 \leqslant z_i \leqslant 1, \forall i\}$. Next, we apply the following scaling and translating operations to transform $\mathbf{u}$ to a feasible solution that satisfies (4.6b)-(4.6e):

$$x_i = \frac{1}{2}(u_i + 1)(\bar{x}_i - \underline{x}_i) + \underline{x}_i, \forall i. \tag{4.10}$$

A diagram is provided in Fig. 4.2 to illustrate the separate network architectures of f^I for (4.3) and (4.6).

[2]The ReLU activation function is $\max(x, 0)$.

4.4.2 Network Design of f^R

The network architecture design for f^R is not as straightforward as for f^I because the constraints in (4.8b)-(4.8d) can not be enforced by simply scaling and translating the neural layers' outputs. More importantly, these constraints involve two sets of variables, namely, $\mathbf{p}^R$ and $\boldsymbol{\delta}^{ns}$, which are interconnected by the equality constraints (the power flow equations) in (4.8b). To this end, we can use (4.8b) to express the recourse variable $\mathbf{p}^R$ as an affine function of $\boldsymbol{\delta}^{ns}$. As a result, we can use f^R only for predicting $\boldsymbol{\delta}^{ns}$, and then reconstruct $\mathbf{p}^R$ using the power flow equations. Therefore, the constraints that f^R need to satisfy have been reduced to only (4.8c) and (4.8d), which define a polyhedral space of dimension $n-1$:

$$\mathcal{P}_{feas} = \left\{ \boldsymbol{\delta}^{ns} | -\bar{\mathbf{p}}^f \leqslant \mathbf{F}\boldsymbol{\delta}^{ns} \leqslant \bar{\mathbf{p}}^f, \underline{\boldsymbol{\delta}} \leqslant \mathbf{E}\boldsymbol{\delta}^{ns} \leqslant \bar{\boldsymbol{\delta}} \right\}. \tag{4.11}$$

Next, we describe the architecture design of f^R that transforms the output of neural layers to a point within $\mathcal{P}_{feas}$.

Concretely, we again use a Tanh activation function on the last neural layer and denote the output from it by $\mathbf{u}$, which satisfies $\mathbf{u} \in \mathbb{B}_\infty$ as we have discussed. Then we utilize the *gauge map* technique [116] to fulfill the transformation. Particularly, the gauge map can establish the equivalence between two C-sets using the gauge functions associated with them. The definitions of C-sets and gauge functions are provided in [116], and we present them again here for the sake of clarity. We will also show that, by these definitions, both $\mathbb{B}_\infty$ and $\mathcal{P}_{feas}$ are C-sets.

Definition 4.1 (C-set [128])**.** *A C-set is a convex and compact subset of $\mathbb{R}^n$ including the origin as an interior point.*

By Definition 4.1, the unit hypercube $\mathbb{B}_\infty$ is a C-set. To see that the polyhedral set $\mathcal{P}_{feas}$ also satisfies Definition 4.1, we first note that the origin is already located within $\mathcal{P}_{feas}$, as the origin satisfies the inequalities defining $\mathcal{P}_{feas}$ in (4.11). Regarding the compactness of $\mathcal{P}_{feas}$, we provide the following theorem.

Theorem 4.2. *The polyhedral set $\mathcal{P}_{feas}$ given by* (4.11) *is bounded.*

The proof of Theorem 4.2 is given in Appendix B.3. Together, we can conclude that $\mathcal{P}_{feas}$ is also a C-set. Before describing the gauge transformation between $\mathbb{B}_\infty$ and $\mathcal{P}_{feas}$, we first introduce the concept of the gauge function associated with a C-set.

Definition 4.3 (Gauge function [128]). *The gauge function associated with a C-set $\mathcal{P}$ is a mapping $g_{\mathcal{P}} : \mathbb{R}^n \longmapsto [0, +\infty]$, given by*

$$g_{\mathcal{P}}(\mathbf{z}) = \min\{\lambda : \mathbf{z} \in \lambda\mathcal{P}, \lambda \geqslant 0, \mathbf{z} \in \mathbb{R}^n\}.$$

If C-set $\mathcal{P}$ is a polyhedral set of the form

$$\mathcal{P} = \{\mathbf{z} \in \mathbb{R}^n | \mathbf{A}\mathbf{z} \leqslant \mathbf{b}, \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^n\},$$

then the gauge function associated with it is

$$g_{\mathcal{P}}(\mathbf{z}) = \max_{i=1,\cdots,m} \{\frac{\mathbf{a}_i^T \mathbf{z}}{b_i}\},$$

where $\mathbf{a}_i$ is the i-th row of $\mathbf{A}$ and b_i is the i-th element of $\mathbf{b}$. The proof of Proposition 4.4.2 is provided in Appendix B.4. By using the gauge function defined in Definition 4.3, we can express the gauge map as follows.

Definition 4.4 (gauge map[116]). *The gauge map between any two C-sets $\mathcal{P}$ and $\mathcal{Q}$ is a bijection $G : \mathcal{P} \longmapsto \mathcal{Q}$ given by*

$$G(\mathbf{z}|\mathcal{P}, \mathcal{Q}) = \frac{g_{\mathcal{P}}(\mathbf{z})}{g_{\mathcal{Q}}(\mathbf{z})}\mathbf{z}.$$

According to this definition, for each point $\mathbf{u}$ within $\mathbb{B}_\infty$, it is possible to convert it into a feasible solution within $\mathcal{P}_{feas}$, denoted as $\boldsymbol{\delta}_u^{ns}$, by establishing a one-to-one correspondence (image) between $\mathbb{B}_\infty$ and $\mathcal{P}_{feas}$ through the following gauge map:

$$\boldsymbol{\delta}_u^{ns} := G(\mathbf{u}|\mathbb{B}_\infty, \mathcal{P}_{feas}) = \frac{g_{\mathbb{B}_\infty}(\mathbf{u})}{g_{\mathcal{P}_{feas}}(\mathbf{u})}\mathbf{u}, \tag{4.12}$$

where $g_{\mathbb{B}_\infty}(\mathbf{u})$ and $g_{\mathcal{P}_{feas}}(\mathbf{u})$ can be calculated using Proposition 4.4.2. Particularly, $g_{\mathbb{B}_\infty}(\mathbf{u}) = \|\mathbf{u}\|_\infty$. It is worth noting that the feasibility of the transformed point $\boldsymbol{\delta}_u^{ns}$ is ensured by this definition of gauge map. To this end, we present the following theorem.

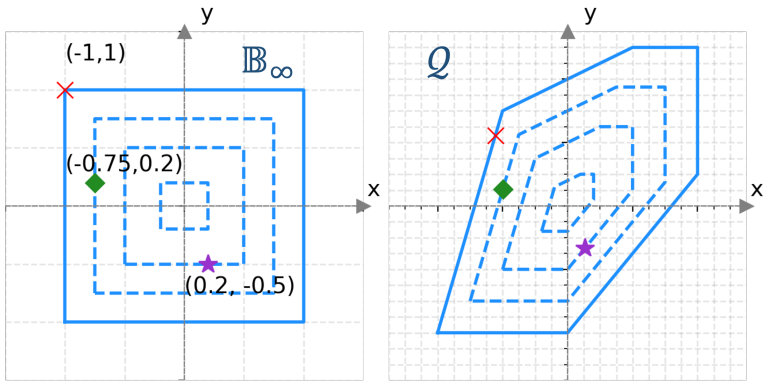


Figure 4.3: An illustrative example of the gauge map from $\mathbb{B}_\infty$ to a polyhedral C-set $\mathcal{Q}$. The 1, $\frac{3}{4}$, $\frac{1}{2}$ and $\frac{1}{5}$ level curves of each set are plotted in blue. For each point in $\mathbb{B}_\infty$, it is transformed to its image (marked using the same color) in $\mathcal{Q}$ with the same level curve.

Theorem 4.5. *Given any point* $\mathbf{u} \in \mathbb{B}_\infty$*, the transformed point* $\boldsymbol{\delta}_u^{ns}$ *by the gauge map in* (4.12) *lies within* $\mathcal{P}_{feas}$*.*

The proof of Theorem 4.5 can be found in Appendix B.5. Moreover, we provide an illustrative example in Fig. 4.3, visualizing the gauge transformation from $\mathbb{B}_\infty$ to a randomly generated polyhedral C-set.

Once a feasible solution of $\boldsymbol{\delta}^{ns}$ is obtained, the values for $\mathbf{p}^R$ and the output of f^R, i.e., the objective value of the deterministic DCOPF in (4.8), can be be easily computed. We summarize the network design of f^R in Fig. 4.4.

Lastly, we discuss the differentiability properties of the function in (4.12) since training the network architecture in Fig. 4.1 requires a backward pass that can calculate the gradients in (B.1). This is a nuanced point since both (4.12) and the layers used in neural networks are not everywhere differentiable. Here, we show that the non-differentiability introduced by the gauge map is no more severe than the non-differentiability that is already present in the neural network activation functions, and the end-to-end policy is differentiable almost everywhere:

Theorem 4.6. *Let* $\mathcal{P}$ *and* $\mathcal{Q}$ *be polyhedral C-sets. Standard automatic differentiation procedures, when applied to the gauge map* $G(\cdot \mid \mathcal{P}, \mathcal{Q})$*, will return the gradient of* $G(\cdot \mid \mathcal{P}, \mathcal{Q})$ *for almost all* $\boldsymbol{z} \in \mathcal{P}$*.*

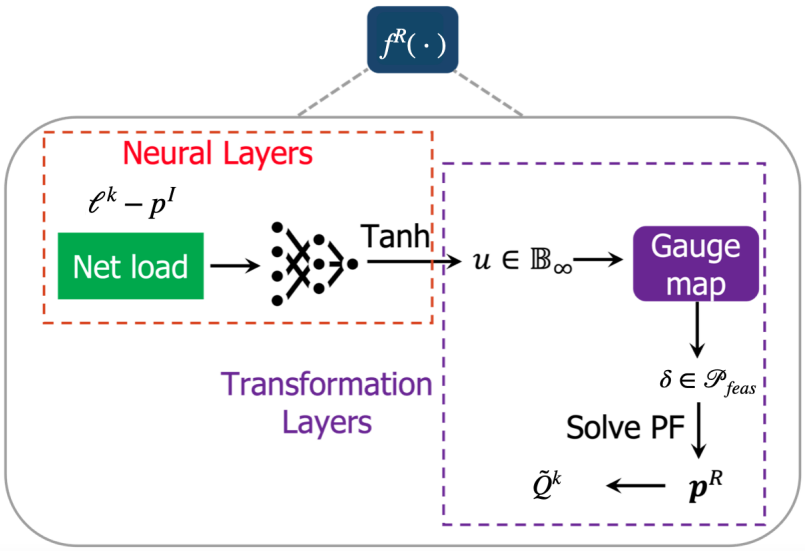


Figure 4.4: The hypercubic output from the neural layers is transformed to a feasible solution for $\boldsymbol{\delta}^{ns}$ by the gauge map. Then the value of the objective function can be easily computed.

Proof. The set $\mathcal{P}$ can be partitioned such that the gauge map is a different analytic function on each region of the partition (excluding the origin). By setting $G(0 \mid \mathcal{P}, \mathcal{Q}) := 0$, we obtain a function for which standard automatic differentiation procedures will compute the gradient of $G(\cdot \mid \mathcal{P}, \mathcal{Q})$ at all $\mathbf{z} \in \mathcal{P}$ except possibly on a set of measure zero [129]. Details are in Appendix B.6. ■

Theorem 4.6 shows that the gauge map is differentiable with respect to the output of the neural layers, and hence enables the computation of backpropagation gradients in (B.1) and the training of the architecture in Fig. 4.1. In the next section, we validate the effectiveness of the proposed learning architecture on a modified IEEE 118-bus system.

4.5 Simulation Studies

In this section, we provide the experimental results of using the proposed algorithm in Table 7.1 to solve two-stage DCOPF problems. Particularly, we consider two application contexts, namely, the risk-limiting dispatch and reserve scheduling problems on two systems, the IEEE 118-bus system [130] and a 2000-bus synthetic grid based on ERCOT [131]. Our algorithm learns the first-stage solutions to the scenario-based problems in (4.3) and (4.6),

respectively. We implement our learning algorithm in Google Colab [132] using Pytorch.

Network architecture: Two convolutional neural networks are constructed, with one representing f^I and the other representing f^R. Each network consists of a 4-layer structure, including 2 convolutional layers followed by 2 fully connected layers. A dropout layer with the rate of 0.5 is used on each of the fully connected layer before the output. The network architectures are trained offline using Adam Optimizer [133] and the default learning rate is adopted. The size of the hidden layer is tuned for each application context and the details can be found in our public code repository.

Data generation: There are two types of data in our algorithm. The first type is the load forecasts. They are inputs to the learning algorithm and comprise the datasets on which we train and test the network architecture. In both application contexts, the training dataset consists of 50000 load forecasts and testing dataset of 100. [3] The second type of data is the load realizations that are used to solve the scenario-based problems (estimating the expected second-stage cost) or to evaluate the solution quality through ex-post *out-of-sample* simulations [109]. During training, after experimenting with various load realization sampling sizes, we found that using 20 load realizations led to a more notable reduction in the training loss over a fixed time period. Therefore, we opt to sample 20 load realizations independently for each batch to approximate the expected second-stage cost. For testing, the first stage solutions obtained using different methods are evaluated on a shared set of load realizations. To ensure a precise evaluation of the solution quality, we use 500 load realizations for out-of-sample simulations.

Both types of data are generated using the Gaussian distribution but with different choices of the mean and standard deviation. When generating load forecasts, we use the base load of the system as the mean and set the standard deviation to be 10% of it. To solve the two-stage problem for a given load forecast, we sample load realizations from a Gaussian distribution centered around this forecast, with a standard deviation of 5% of this

[3] The reason for limiting our testing to a small dataset in these experiments is not because of our learning algorithm, rather it was due to the high computational cost of generating benchmark solutions using CVXPY. As we will see, obtaining 100 benchmark solutions for the reserve scheduling problem in a 118-bus network took approximately 5 hours. By contrast, since our model directly optimizes the objective function, we do not need ground truth data for training, so we can have a large training set.

forecasted load value.

Baseline solvers: In both application contexts, we call the solver from CVXPY [88] to solve the scenario-based problems in (4.3) and (4.6) on the testing dataset to establish a benchmark. We also compare the solutions produced by our method to that by using the affine policy method, which is a widely applied approximation policy to make the two-stage stochastic programs tractable [134]. Specifically, the affine policy approximates the recourse decision $\mathbf{p}^R$ by representing it as an affine function of the uncertain parameter. This allows it to solve a simplified version of the original problem and save solving time. Further information on how to implement the affine policies for each specific application can be found in Appendix B.7.

Evaluation procedure: For each testing instance, we apply various methods to produce their respective first-stage solutions. To evaluate the quality of a specific first-stage solution, we compute the total cost arising from it by aggregating both the first-stage cost and the corresponding cost-to-go. The cost-to-go associated with a given first-stage solution is derived by solving the second-stage problem defined in (4.8) over a common set of 500 load realizations. We then compare the performance of different methods based on their resulting average total costs and solving times on the testing dataset.

4.5.1 Application I: Risk-Limiting Dispatch

The results of using different methods to solve the risk-limiting dispatch problem in (4.3) on the 118-bus system and the 2000-bus system are provided in Table 4.2 and Table 4.3, respectively. The average total costs of different methods are compared against the cost values produced by CVXPY. In both cases, our learning method is faster than applying CVXPY solver by 6 orders of magnitude while the difference in average total cost is less than 2%. In comparison, using the affine policy reduces the average running time by half, however, it also performs 20% to 100% worse. This is because the affine policy has bad generalization when applied to never-seen instances of load forecasts, and tend to resort to load shedding in the second stage, leading to high costs.

Application I: Risk-limiting dispatch on 118-bus system		
Methods	Increase in Total cost (average, %)	Solving Time (average, seconds)
CVXPY	-	24
Proposed	**0.77**	$\mathbf{1 \cdot 10^{-7}}$
Affine policy	99.4	1.2

Table 4.2: Performances of our method and the affine policy method to solve the risk-limiting dispatch problem in (4.3) on the 118-bus system. The resulting total costs and solving times are averaged out over 100 test instances and compared against the benchmark solutions obtained from CVXPY.

Application I: Risk-limiting dispatch on 2000-bus system		
Methods	Increase in Total cost (average, %)	Solving Time (average, seconds)
CVXPY	-	71
Proposed	**1.5**	$\mathbf{1.2 \cdot 10^{-4}}$
Affine policy	19.6	3.6

Table 4.3: Compare the performance of using our method and the affine policy method to solve risk-limiting dispatch on the 2000-bus system. The resulting total costs and solving times are averaged out over 1000 test instances and compared against the benchmark solutions obtained from CVXPY.

4.5.2 Application II: Reserve Scheduling

We summarize the results of using different methods to solve the reserve scheduling problem in (4.6) on the 118-bus system in Table 4.4. All reported total costs are normalized in reference to the cost values obtained from CVXPY. Compared to the risk-limiting dispatch problem, the reserve scheduling problem has more decision variables and constraints, which make it more challenging to solve. It takes minutes to solve a single instance in CVXPY. By using an affine policy to approximate the recourse dispatch decision, the average running time per instance can be reduced by an order of magnitude, but the average total cost also

increases by an order of magnitude due to poor generalization. In contrast, our learning method not only learns to provide good solution quality (within 10% of the benchmark produced by CVXPY solver) but is also able to speed up the computation by 4 orders of magnitude.

Application II: Reserve scheduling on 118-bus system		
Methods	Total cost (average, %)	Solving Time (average, minutes)
CVXPY	100	3.210
Proposed	**110**	$\mathbf{10^{-4}}$
Affine policy	1813	0.343

Table 4.4: Compare the performance of using our method and the affine policy method to solve the reserve scheduling problem in (4.6) on the 118-bus system. The resulting total costs and solving times are averaged out over 100 test instances and compared against the benchmark solutions obtained from CVXPY.

Next, we compare our method with the widely used K-means scenario reduction method. Specifically, we reduce a collection of scenarios (load realizations) into two reduced sets: one with 5 scenarios and the other with 2 scenarios. To generate load realizations, we use a Gaussian distribution centered around the the input forecast, with standard deviation of 50% of this forecast.

We assess the performance of our method and the K-means scenario reduction approach across different experimental configurations by comparing their average total costs and solving times across 100 test instances against the benchmark solutions obtained using CVXPY. The results are detailed in Table 4.5. Particularly, all cost values are normalized relative to CVXPY's solutions, and we report the average increase in total cost within the table.

As we can see, our method achieves the smallest increase in total cost while requiring the least amount of time across both datasets with small and large load variations. Notably, in order to achieve a solving time comparable to ours, the scenario reduction method needs

to shrink the scenario set by a factor of 10, i.e, to 2 scenarios. However, this considerable reduction significantly diminishes its performance when contrasted with both our method and the case using a reduced set of 5 scenarios. On the other hand, although using a factor of 10, corresponding to 5 scenarios, gives improved performance for the K-means scenario reduction method compared to the use of 2 scenarios, the resulting average total cost remains higher than our method's, and the solving speed is considerably slower, being 100 times slower than ours.

Comparions with Scenario Reduction		
Methods	Increase in Total cost (average, %)	Solving Time (average, seconds)
CVXPY	-	29
Proposed	**2.8**	$\mathbf{6 \cdot 10^{-4}}$
2 scenarios	6.6	$3 \cdot 10^{-2}$
5 scenarios	4.4	$6 \cdot 10^{-2}$

Table 4.5: Compare the performance of using our method and the K-means scenario reduction method for the 118-bus system. The resulting average total costs and solving times across 100 test instances are compared against the benchmark solutions obtained from CVXPY.

4.6 Conclusions

This chapter presents a learning algorithm to solve two-stage DCOPF problems efficiently. The algorithm uses two neural networks, one for each stage, to make the dispatch decisions. The gauge map technique is built into the network architecture design so that the constraints in two-stage DCOPF problems can be guaranteed to be satisfied. This allows the proposed model to be trained in an unsupervised way without the need for generating ground truth data. The numerical results on the IEEE 118-bus and 2000-bus systems validate the effectiveness of the proposed algorithm, showing that it can speed up computation by orders of magnitude compared to the commercial solver while still learning

high-quality solutions. A direction of future work is to generalize our learning algorithm to solve non-convex programs, for example, using the AC optimal power flow model.

Chapter 5

IMPROVING SOLUTION QUALITY FOR AC OPTIMAL POWER FLOW PROBLEMS FROM A LAGRANGIAN PERSPECTIVES

5.1 Introduction

The optimal power flow (OPF) problem is a fundamental resource allocation problem in power system operations that minimizes the cost of power generation while satisfying demand. The ACOPF formulation of the problem uses nonlinear power flow equations, resulting in nonlinear and nonconvex optimization problems [135, 3, 136]. The consequence of the nonconvexity of ACOPF we study in this chapter is the presence of multiple solutions.

Most ACOPF problems are solved via variations of nonlinear optimization algorithms, including Newton-Raphson, sequential programming, interior points and others (see [3, 137, 138] and the references within). These algorithms are in general only able to certify whether a solution is locally optimal, that is, they satisfy first order and/or second order optimality conditions. Because of the existence of multiple local solutions, it is often difficult to find the globally optimal one and the system loses efficiency.

The existence of multiple local solutions of the OPF problem has been well-known for several decades [139, 140, 141]. Despite this, a common assumption is that OPF problems tend to have a single "practical" solution that is globally optimal, and therefore the fact that multiple solutions can exist do not impact day-to-day operations [142, 143]. However, an increasingly large body of work have pointed to that multiple solutions do occur under reasonable conditions and cannot be easily ruled out [4, 144, 3]. For example, during day-to-day operations, small changes in the load could lead to a solver jumping between solutions that are far apart from each other [145, 146]. In addition, [4] shows how modifications of the standard IEEE benchmarks can lead to each having more than one local solution. Statistical studies in [147, 148] show that there are more solutions than previously thought in many systems.

An open question in the field is to develop algorithms that can find globally optimal solutions, or at least improve upon local ones. In addition to lowering the operating cost, understanding and distinguishing between locally and globally optimal solutions can lead to important theoretical discoveries about the ACOPF problem. Consequently, several classes of algorithms have been developed. For example, holomorphic embedding has been used in [149, 150], but are slow and require very high numerical precision. Genetic algorithms can escape a local minimum, but are random in nature and require repeated trial and error [151, 152]. Robust optimal power flow can alleviate convergence issues, but may not improve on the quality of the solution [153]. Compared to meshed networks, radial networks are less complicated and there have been advances in the solving techniques for radial networks to avoid being kept at strictly local solutions. A common method is to convexify the nonlinear and nonconvex power flow equations [154, 155, 156, 157]. However, these relaxations can have difficulties when there are lower bounds on reactive or active power and the feasibility region of the problem becomes disconnected.

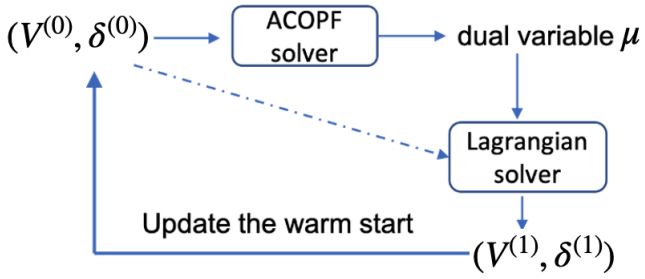


Figure 5.1: Outline of our algorithm. We form partial Lagragians at local optima and solve it to provide better warm starting points. Note that the partial Lagrangian is solved using the same initialization as in the first call of the solver.

In this chapter, we propose a simple algorithm that can effectively escape from strict local solutions to find better ones. By moving from one solution to another while reducing the cost, we can successively move towards the globally optimal solution. In contrast to algorithms that are launched repeatedly with random initializations, our proposed algorithm is deterministic. And it relies on duality theory to provide better warm starts to existing

solvers.

Our process is outlined in Fig. 5.1. First, we solve the ACOPF problem using some solver (e.g., IPOPT [1] or Matpower [2]). From the solution and its associated dual variables, we form a partial Lagrangian by dualizing the power balance equations. We then optimize this partial Lagrangian, which leads to a different solution. Using this second solution as a warm start, we again call the solver for the ACOPF problem. Interestingly, this iterative process moves from higher cost to lower cost ones.

The key reason for the success of the algorithm is that the geometry of the partial Lagrangian is much more "friendly" than the geometry of the original ACOPF problem. We provide both theoretical analysis on tree networks and simulation results on standard 9-bus, 22-bus, 39-bus, 118-bus and 300-bus meshed networks and also on the IEEE 141-bus radial network. We show that our algorithm can quickly escape from local solutions and find lower cost solutions. This feature holds even for ACOPF problems with disconnected feasible spaces, for example, the 2-bus network shown in Section 5.4.1, which has been traditionally difficult to deal with [3, 4]. For networks with known global solutions (3, 9, 22, 118, 300-bus), we show that our algorithm can find the globally optimal solution in a single iteration, even starting from a strict local solution.

Our approach can be seen as a way to provide good warm starts to nonlinear optimization solvers. DCOPF is commonly used, although it can fail to find good starting points as shown by our simulations as well as existing results [158]. More sophisticated approaches either randomizes (stochastic gradient in [145] and load fluctuations in [146]) or uses a previous solution as the starting point [159]. The former tends to be time consuming, while the latter tends to lead to system being stuck in a strict local solution [4].

5.2 Model and Problem Formulation

Consider a power system network of n buses and m lines. For bus i, let V_i denote its voltage magnitude, δ_i its angle, p_i^g and q_i^g the active and reactive output of the generator and p_i^d and q_i^d the active and reactive load. The admittance between i and j is $g_{ij} - jb_{ij}$, the active power and reactive power flow from i to j is p_{ij}^f and q_{ij}^f and δ_{ij} is used as a shorthand for $\delta_i - \delta_j$.

The ACOPF problem is to minimize the cost of active power generations while satisfying a set of constraints [4]. Out of the feasible solutions, we focus on two classes: local solutions and global solutions. Local solutions are all the solutions that satisfy local optimality conditions, for example, the KKT conditions or second order ones [160]. Out of this set, the solutions with the lowest cost are called the global ones. We sometimes refer to the local solutions that are not global as strict local solutions. We assume both strict local solutions and the global solutions are regular and the KKT conditions always hold at these solutions [160]. Therefore, there always exist unique Lagrangian multipliers and we can use them to form the partial Lagrangian.

Specifically, the ACOPF problem is given by the following optimization problem:

$$\min_{\mathbf{V},\boldsymbol{\delta}} \textstyle\sum_i c_i(p_i^g) \tag{5.1a}$$

$$\text{s.t. } p_i^g = p_i^d + \textstyle\sum_{j:(i,j)\in\mathcal{E}} p_{ij}^f \tag{5.1b}$$

$$q_i^g = q_i^d + \textstyle\sum_{j:(i,j)\in\mathcal{E}} q_{ij}^f \tag{5.1c}$$

$$p_{ij}^f = V_i^2 g_{ij} - V_i V_j (g_{ij}\cos(\delta_{ij}) - b_{ij}\sin(\delta_{ij})) \tag{5.1d}$$

$$q_{ij}^f = V_i^2 \hat{b}_{ij} - V_i V_j (b_{ij}\cos(\delta_{ij}) + g_{ij}\sin(\delta_{ij})) \tag{5.1e}$$

$$\underline{V}_i \leqslant V_i \leqslant \bar{V}_i \tag{5.1f}$$

$$\underline{p}_i^g \leqslant p_i^g \leqslant \bar{p}_i^g \tag{5.1g}$$

$$\underline{q}_i^g \leqslant q_i^g \leqslant \bar{q}_i^g \tag{5.1h}$$

$$(p_{ij}^f)^2 + (q_{ij}^f)^2 \leqslant (S_{ij}^{\max})^2 \tag{5.1i}$$

where $\hat{b}_{ij} = b_{ij} + 0.5 b_{ij}^C$ and b_{ij}^C is the line charging susceptance. The constraints (5.1b) and (5.1c) enforce power balance, (5.1d) and (5.1e) are the AC power flow equations, (5.1f) limits the bus voltage magnitudes, (5.1g) and (5.1h) represent the active and reactive limits and (5.1i) are the line flow limits. We assume the cost at each bus i, i.e., $c_i(\cdot)$, is increasing. Other than that, the cost can be linear, quadratic or other functions. We assume problem (5.1) is feasible in this chapter.

Over the years, many nonlinear programming (NLP) solvers have been developed for the ACOPF problem, and their speed and efficiency have improved dramatically (e.g., see [135] and the references within). However, NLP solvers are typically only able to return local

solutions. Since a local solution is not necessarily global, we propose an iterative approach to improve the solution quality by alternatively solving (5.1) and a partial Lagrangian. Any NLP solver can be used, and we use IPOPT [1] in this chapter.

5.3 Proposed Iterative Algorithm

Our algorithm starts with a call to a NLP solver with some initial guess, denoted by $\boldsymbol{\delta}_{\text{init}}$, $\mathbf{V}_{\text{init}}$. For example, this can be the standard flat start with voltage magnitudes being 1 p.u. and angles set to 0. Then we assume the solver returns a feasible solution. Of course, we don't know whether this solution is globally optimal or not. At this solution, we record the dual variables associated with the power balance equations (5.1b) and (5.1c), denoted as $\bar{\boldsymbol{\mu}}^P$ and $\bar{\boldsymbol{\mu}}^Q$. Using these dual variables, we form the following partial Lagrangian by dualizing the power balance equations:

$$\begin{aligned}\mathcal{L}(\mathbf{V}, \boldsymbol{\delta}, \boldsymbol{\mu}^P, \boldsymbol{\mu}^Q) = &\sum_i c_i p_i^g + \sum_i \mu_i^P (p_i^d + \sum_{j:(i,j)\in\mathcal{E}} p_{ij}^f - p_i^g) \\ &+ \sum_i \mu_i^Q (q_i^d + \sum_{j:(i,j)\in\mathcal{E}} q_{ij}^f - q_i^g).\end{aligned}$$

We then minimize the partial Lagrangian by solving

$$\begin{aligned}\min_{\mathbf{V},\boldsymbol{\delta}} \ & \mathcal{L}(\mathbf{V}, \boldsymbol{\delta}, \boldsymbol{\mu}^P, \boldsymbol{\mu}^Q) \\ \text{s.t. } & (5.1d) - (5.1i).\end{aligned} \tag{5.3}$$

The problem in (5.3) can be solved using any NLP solver. Since the partial Lagrangian has less constraints than the primal problem, its feasible solution space is larger. Therefore, the problem in (5.3) is feasible if the original ACOPF problem is feasible.

We solve the problem in (5.3) starting from the same initial point $(\mathbf{V}_{\text{init}}, \boldsymbol{\delta}_{\text{init}})$ that was used to solve the original primal problem in (5.1). Denote this solution to (5.3) by $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$. Note $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$ will not be the same as $(\mathbf{V}_{\text{init}}, \boldsymbol{\delta}_{\text{init}})$ since they come from different problems. Then we start the NLP solver again to solve (5.1) but with the initial point $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$. This process can be repeated until the solutions stop changing or up to a predefined number of iterations.

The key observation in this chapter is that the solution $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$ found by solving the partial Lagrangian is often a much better starting point than the original choice of $(\mathbf{V}_{\text{init}}, \boldsymbol{\delta}_{\text{init}})$. Therefore, by repeating these steps, we can iteratively improve the solution quality (i.e., reducing the cost). The algorithm is summarized below as Algorithm 5.1. We illustrate the intuition behind this algorithm in the next section using 2-bus and 3-bus networks. Formal proofs are given in Section 5.5.2, and simulations results for larger IEEE benchmarks are presented in Section 5.6.7.

Algorithm 5.1: Solving ACOPF iteratively

Inputs: $\boldsymbol{\delta}_{\text{init}}^{(i)}$, $\mathbf{V}_{\text{init}}^{(i)}$, $i = 0$

1: At i-th iteration: Initialized at $\boldsymbol{\delta}_{\text{init}}^{(i)}$, $\mathbf{V}_{\text{init}}^{(i)}$:

2: Call NLP solver for (5.1), record $(\bar{\boldsymbol{\mu}}_{(i)}^{P}, \bar{\boldsymbol{\mu}}_{(i)}^{Q})$.

3: Given $(\bar{\boldsymbol{\mu}}_{(i)}^{P}, \bar{\boldsymbol{\mu}}_{(i)}^{Q})$, call solver for the partial Lagrangian in (5.3), record the solutions as $(\bar{\boldsymbol{\delta}}^{(i)}, \bar{\mathbf{V}}^{(i)})$.

4: Call IPOPT for (5.1) initialized at $(\bar{\boldsymbol{\delta}}^{(i)}, \bar{\mathbf{V}}^{(i)})$, record solutions $(\hat{\boldsymbol{\delta}}^{(i)}, \hat{\mathbf{V}}^{(i)})$.

5: If the solution from line 4 does not reduce the cost, terminate the algorithm.

6: Otherwise, update initial points: $\boldsymbol{\delta}_{\text{init}}^{(i+1)} = \hat{\boldsymbol{\delta}}^{(i)}$, $\mathbf{V}_{\text{init}}^{(i+1)} = \hat{\mathbf{V}}^{(i)}$.

7: Repeat 1-6 until the maximum number of iterations is reached.

Table 5.1: Solving ACOPF iteratively.

In terms of computational overhead, each iteration of Algorithm 5.1 solves an ACOPF problem twice and an OPF-like problem (minimizing the partial Lagrangian) once. In practice, we observe that the cost is reduced after every iteration and the global solution can be reached in a small number of iterations (for the cases where the global solution is known). Therefore, in contrast to algorithms that resolve the ACOPF problem from a large

number of random initialization points [145], Algorithm 5.1 is much more computationally efficient.

5.4 Geometry and Intuition

In this section, we study the geometry of the ACOPF problem to shed some light on why Algorithm 5.1 might be successful. We find that the main reason is that the optmization landscape of the partial Lagrangian is much "better" than the landscape of the original problem. To illustrate this geometric property, we use the 2-bus and 3-bus networks as examples. The formal proofs are provided in Section 5.5.2.

5.4.1 2-bus network

In this part, we consider a 2-bus network. For simplicity, we ignore the reactive power and set both voltage magnitudes to 1 p.u.. Suppose bus 1 is a generator and also the reference (slack) bus with an increasing cost function $c(\cdot)$, and bus 2 is the load bus with angle $-\delta$. The line admittance is $g - jb$. Given a load of l at bus 2 and ignoring all constraints except for the load balancing one, the ACOPF in (5.1) becomes

$$\min_{\delta} \; c(g - g\cos(\delta) + b\sin(\delta)) \tag{5.4a}$$

$$\text{s.t. } \ell + g - g\cos(\delta) - b\sin(\delta) = 0. \tag{5.4b}$$

This is an example of an OPF with a disconnected feasible space, since there are two discrete solutions to (5.4b) and we are asking for the one with lower cost.

To see how a NLP solver would approach this problem, we adopt a common practice [160, 146] and form a penalized version of (5.4). [1] The penalized unconstrained problem is given by

$$\begin{aligned} \mathcal{L}_\rho = & c(g - g\cos(\delta) + b\sin(\delta)) \\ & + \rho/2(\ell + g - g\cos(\delta) - b\sin(\delta))^2, \end{aligned} \tag{5.5}$$

[1] We use quadratic penalties only as an analysis method that allows us to obtain cleaner theoretical results. This approach is standard in convergence analysis of nonlinear programming, for example, see [160] (Chap. 1), [146], and [161]. All simulation and numerical results in this chapter are obtained using state-of-the-art solvers rather than quadratic penalty method.

where ρ is a penalty parameter. For large enough ρ, the solutions of (5.5) would coincide with those of (5.4) [160]. The function $\mathcal{L}_\rho$ is plotted in Fig. 5.2 (green line). We can see that there are two local minimas, with the left one being global. The strict local minimum (the right one) satisfies both first and second order optimality conditions. Therefore, if we initialize an NLP solver with a bad starting point, it would be stuck at the strict local solution. For this example, if the initial point is to the left of the maximum of the green curve, a solver would converge to the left solution; and if the initial point is to the right, a solver would find the right (suboptimal) solution. Hence, a flat start would lead to the global solution. However, for larger systems, flat starts are often not successful (e.g., see the 22-bus system in Section 5.6.7). Therefore, this 2-bus example is useful as it illustrates the geometry of the optimization landscape.

Now suppose μ is the multiplier corresponding to the equality constraint (5.4b) at *the strict local solution*. The partial Lagrangian of (5.4) by dualizing (5.4b) is:

$$\begin{aligned}\mathcal{L}_\mu =& c(g - g\cos(\delta) + b\sin(\delta)) \\ &+ \mu(\ell + g - g\cos(\delta) - b\sin(\delta)).\end{aligned} \tag{5.6}$$

Since the sinusoidal functions are periodic with period 2π, let us consider the range $\delta \in [-\pi, \pi]$. It is interesting now to compare the solution of $\mathcal{L}_\mu$ and the original problem in (5.4) (or equivalently, $\mathcal{L}_\rho$). The blue curve in Fig. 5.2 plots $\mathcal{L}_\mu$. We observe two interesting facts. The first is that unlike $\mathcal{L}_\rho$, $\mathcal{L}_\mu$ in this 2-bus network only has a single minimum. Therefore, no matter where we initialize the NLP solver for $\mathcal{L}_\mu$, we would reach this minimum. The second fact is that the minimum of $\mathcal{L}_\mu$ is close to the global minimum of $\mathcal{L}_\rho$. Therefore, if we start a NLP solver for the ACOPF at the solution of $\mathcal{L}_\mu$, we would reach the global solution. Interestingly, we are using the multiplier at the strict local solution. So even if a solution is not global, it is still very useful, since by solving $\mathcal{L}_\mu$ as an intermediate step, we would not be stuck at the strict local solution. We prove that this procedure is guaranteed to work for tree networks in the next section.

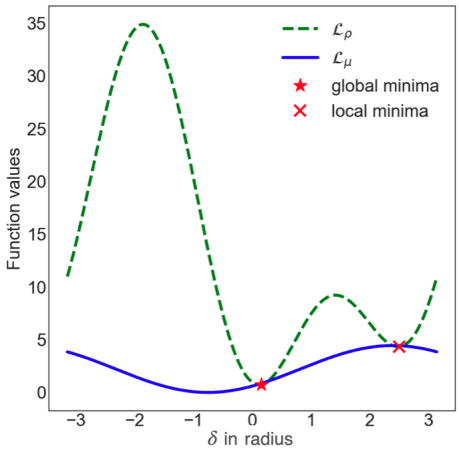


Figure 5.2: Geometry of the penalized objective functions $\mathcal{L}_\rho$ and the partial Lagrangian $\mathcal{L}_\mu$. The line admittance is $1 - j4$ and the penalty parameter is 2.

5.4.2 3-bus network

Now, let us show that the intuitions built in the 2-bus example still carry over into the 3-bus network. We again ignore the reactive power and set all voltage magnitudes to 1 p.u. to optimize over the angles. Suppose bus 1 is a generator and also the reference bus with an increasing cost function $c(\cdot)$, while bus 2 and bus 3 are load buses with angles $-\delta_2$ and $-\delta_3$, respectively. The load at bus 2 is ℓ_2 and at bus 3 is ℓ_3. Then the ACOPF in (5.1) can be simplified to

$$\min_{\delta_2,\delta_3} c\Big(\sum_{j=2,3} g_{1j} - g_{1j}\cos(\delta_j) + b_{1j}\sin(\delta_j)\Big) \tag{5.7a}$$

s.t.

$$\ell_2 + \sum_{j=1,3} \left(g_{2j} - g_{2j}\cos(\delta_{2j}) - b_{2j}\sin(\delta_{2j})\right) = 0 \tag{5.7b}$$

$$\ell_3 + \sum_{j=1,2} \left(g_{3j} - g_{3j}\cos(\delta_{3j}) - b_{3j}\sin(\delta_{3j})\right) = 0. \tag{5.7c}$$

As in the 2-bus case, to understand how a NLP solver may approach (5.7), we form its

penalized version:

$$\begin{aligned}\mathcal{L}_\rho = c(&\sum_{j=2,3} g_{1j} - g_{1j}\cos(\delta_j) + b_{1j}\sin(\delta_j)) \\ &+ \frac{\rho}{2}\left(\ell_2 + \sum_{j=1,3}\left(g_{2j} - g_{2j}\cos(\delta_{2j}) - b_{2j}\sin(\delta_{2j})\right)\right)^2 \\ &+ \frac{\rho}{2}\left(\ell_3 + \sum_{j=1,2}\left(g_{3j} - g_{3j}\cos(\delta_{3j}) - b_{3j}\sin(\delta_{3j})\right)\right)^2.\end{aligned} \tag{5.8}$$

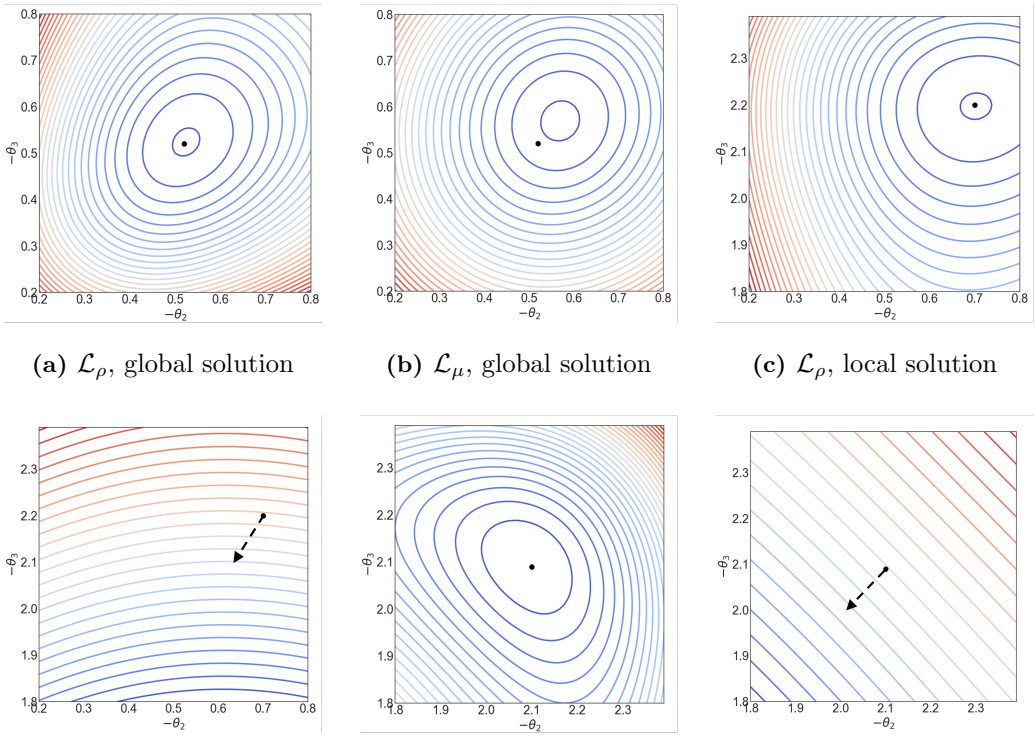
It turns out that there are four local solutions (one of which is global) for (5.8).[2] All of these solutions satisfy both first order and second order conditions and they are listed in Table 5.2. At these solutions, the gradients $\nabla\mathcal{L}_\rho$ are zero and the Hessians $\nabla^2\mathcal{L}_\rho$ are positive definite. This makes $\mathcal{L}_\rho$ look like valleys (convex) at all of the minimas. So it can be hard for an NLP solver to get out of being trapped at a strict local minima. The level sets around the solutions of $\mathcal{L}_\rho$ are plotted on the left of Fig. 5.3, where there is little difference between the local and the global minima.

Solution	Bus 2	Bus 3	Hessian matrix of $\mathcal{L}_\mu$
1st (global)	$\angle 0.52$	$\angle 0.52$	Positive definite
2nd	$\angle 0.7$	$\angle 2.2$	Indefinite
3rd	$\angle 2.2$	$\angle 0.7$	Indefinite
4th	$\angle 2.09$	$\angle 2.09$	Negative definite

Table 5.2: The four solutions to problem (5.7) through grid search. The Hessian of $\mathcal{L}_\rho$ is positive definite at all solutions. The definiteness of the Hessian of $\mathcal{L}_\mu$ is listed. The parameters are $g_{12} - jb_{12} = g_{13} - jb_{13} = 1 - j4$ and $g_{23} - jb_{23} = 0.1 - j0.4$.

Now we show that a partial Lagrangian behaves qualitatively differently. Suppose that we choose a strict local solution of (5.7). Let the multipliers corresponding to the equality constraints (5.7b) and (5.7c) be μ_1 and μ_2, respectively. The partial Lagrangian for (5.7)

[2]They are found via a grid search, i.e., we finely discretize the space and exhaustively check all points.



(a) $\mathcal{L}_\rho$, global solution **(b)** $\mathcal{L}_\mu$, global solution **(c)** $\mathcal{L}_\rho$, local solution

(d) $\mathcal{L}_\mu$, local solution **(e)** $\mathcal{L}_\rho$, local solution **(f)** $\mathcal{L}_\mu$, local solution

Figure 5.3: The contour plot of $\mathcal{L}_\rho$ and $\mathcal{L}_\mu$ nearby the 1st, 2nd and 4th solution. The Hessian matrix of $\mathcal{L}_\mu$ is positive definite in (b), indefinite in (d), and negative definite in (f). The black arrows in (d) and (f) indicates the descent directions of the function value.

is:

$$\mathcal{L}_\mu = c(\sum_{j=2,3} g_{1j} - g_{1j}\cos(\delta_j) + b_{1j}\sin(\delta_j)) \tag{5.9a}$$

$$+ \mu_1(\ell_2 + \sum_{j=1,3} (g_{2j} - g_{2j}\cos(\delta_{2j}) - b_{2j}\sin(\delta_{2j}))) \tag{5.9b}$$

$$+ \mu_2(\ell_3 + \sum_{j=1,2} (g_{3j} - g_{3j}\cos(\delta_{3j}) - b_{3j}\sin(\delta_{3j}))). \tag{5.9c}$$

In contrast to the penalized problem, there is only a single solution for $\mathcal{L}_\mu$ which satisfies both the first order and second order optimality conditions. It is close to the global solution of $\mathcal{L}_\rho$ (the black dot in Fig. 5.3(b)), even though the multipliers used in forming $\mathcal{L}_\mu$ are from a strict local solution.

If we look at the Hessian of $\mathcal{L}_\mu$, we see that the Hessian is either negative definite or indefinite at the strict local solutions of $\mathcal{L}_\rho$ (the definiteness of the Hessians for $\mathcal{L}_\mu$ at the local solutions of $\mathcal{L}_\rho$ are listed in Table. 5.2). If the Hessian is not positive semidefinite, then there is always a direction to lower the objective value of a function. For example, these descent directions are shown as dotted arrows in Fig. 5.3(d) and Fig. 5.3(f).

All together, Fig. 5.3 shows how Algorithm 5.1 would get around the strict local solutions in $\mathcal{L}_\rho$. Suppose we solve the Lagrangian from a point around the global minimum $\mathbf{x}^\star$. Since $\nabla^2\mathcal{L}_\mu(\mathbf{x}^\star)$ is positive definite, this means the starting point is at a valley of the Lagrangian surface. So solving the Lagrangian would return the global solution. Now let us use a point around the local solution, say $\bar{\mathbf{x}}$, as an initial point to solve the Lagrangian. As shown in Fig. 5.3(d)) and 5.3(f)), $\nabla^2\mathcal{L}_\mu(\bar{\mathbf{x}})$ is negative definite or indefinite, so the surface of the Lagrangian is concave down or has a saddle. Then we can find at least one descent direction to get out of being trapped at the current point.

5.5 Analysis of Algorithm 5.1

In this section, we provide a rigorous analysis of Algorithm 5.1. As a first step, we focus our attention on systems with a tree topology and ignore the reactive power. For mesh networks with both active and reactive power flows, we provide detailed simulation studies in Section 5.6.7 and show how the intuition from tree networks applies. Formal proofs for meshed systems is an important part of our future work.

We first consider a tree network with fixed voltage magnitudes and show that the minimizer of the Lagrangian falls into the attraction basin of the global minimum of the ACOPF problem, which generalizes the observations in Section 5.4.1. Then we optimize over both voltage magnitudes and angles for a 2-bus network, and look at the Hessian matrix of the Lagrangian as we do in Section 5.4.2. We prove that the Hessian matrix of the Lagrangian is positive definite at the global minimum and negative definite or indefinite at the local minimum.

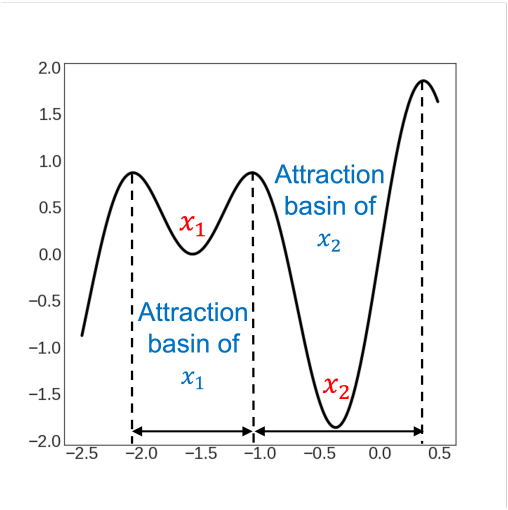


Figure 5.4: Illustrative figure of Definition 5.1. There are two local minimums, i.e., x_1 and x_2. The attraction basins of each of them are marked as intervals and annotated with blue fonts.

5.5.1 Fixed voltage magnitudes

In this part, we consider a tree network with fixed voltage magnitudes. We note that the NLP solver we use in this chapter, IPOPT, uses a barrier function to solve a sequence of unconstrained optimization problems using a mixture of gradient descent and Newton-type methods (with many different ways of tuning stepsizes). Without loss of generality, we assume that the NLP solver runs either a gradient descent or a Newton-type algorithm. For either algorithm, there is a theorem called the Capture Theorem (see [160], Prop. 1.2.4 for gradient-like algorithms and Prop. 1.4.1 for Newton-type algorithms) saying that once the algorithm enters the region of attraction around a local minimum it has to go to this local minimum. This means starting the solver from an initial point in the region of attraction of a solution would return this solution. Formally, the region of attraction of a solution is defined as follows [160]:

Definition 5.1. *Let* $\mathbf{x}^\star$ *be an unconstrained local minimum to* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. *Assume there exists a set* $\mathcal{X}$ *such that* $f(\mathbf{x})$ *is continuously differentiable on* $\mathcal{X}$ *and* $\mathbf{x}^\star \in \mathcal{X}$. *For every point* $\mathbf{x} \neq \mathbf{x}^\star$ *and* $\mathbf{x} \in \mathcal{X}$, *if the following inequality holds, then* $\mathcal{X}$ *is a subset of the region*

of attraction around $\mathbf{x}^\star$*:*

$$\nabla f(\mathbf{x})^T(\mathbf{x}^\star - \mathbf{x}) < 0,\ \forall \mathbf{x} \neq \mathbf{x}^\star, \mathbf{x} \in \mathcal{X}, \tag{5.10}$$

where $\nabla f(\mathbf{x})$ *represents the gradient of* $f(\cdot)$ *at the point* $\mathbf{x}$*.*

An illustrative figure of Definition 5.1 is given in Fig. 5.4. Intuitively, the inequality in (5.10) implies that the direction where the function value decreases (the descent direction) is aligned with the negative gradient. Now we give the following theorem about the performance of Algorithm 5.1 for a tree network with fixed voltage magnitudes:

Theorem 5.2. *Consider a n-bus radial network and keep the voltage magnitudes fixed to optimize over voltage angles. If the NLP solver gets stuck at a strictly local solution when it starts from an initialization point, then starting Algorithm 5.1 from the same initialization point is able to escape from this strictly local solution.*

Proof. Since Algorithm 5.1 uses the solution of the partial Lagrangian as a new initialization point to solve the primal problem again, we prove Theorem 5.2 by showing that the minimizer of the partial Lagrangian falls into the region of attraction around the global minimum of the ACOPF problem. As we assumed at the beginning of this part, the NLP solver is running either a gradient descent or a Newton-type algorithm. Both of the two follow the Capture Theorem in [160]. As a result, starting the solver from the minimizer of the Lagrangian would be able to find the global minimum.

We do the proof by induction starting with a 2-bus network. The ACOPF problem for the 2-bus network is given in (5.4) and its Lagrangian is given in (5.6). We first study the solutions to (5.4) by looking at the equality constraint $h(\delta) = \ell + g - g\cos(\delta) - b\sin(\delta) = 0$. Its gradient can be written as

$$h'(\delta) = g\cos(\delta)(\tan(\delta) - b/g).$$

Suppose $\delta \in (-\pi/2, 3\pi/2)$, then the gradient h' is zero at $\delta = \tan^{-1}(b/g)$. We also have

$$h'(\delta) < 0, \forall\ \delta \in (-\frac{\pi}{2}, \tan^{-1}(b/g)) \tag{5.11a}$$

$$h'(\delta) > 0, \forall\ \delta \in (\tan^{-1}(b/g), \tan^{-1}(b/g) + \pi). \tag{5.11b}$$

This implies that $\delta = \tan^{-1}(b/g)$ is a minima of $h(\delta)$. Since for a feasible problem, the solution to $h(\delta) = 0$ must exist within $(-\pi/2, 3\pi/2)$, then by the intermediate value theorem, there are two solutions to (5.4), which satisfy the following inequalities:

$$-\pi/2 < \delta^\star < \tan^{-1}(b/g) < \bar{\delta} < 3\pi/2, \tag{5.12}$$

where $\delta^\star$ is the global minimum and $\bar{\delta}$ is the local minimum (see Appendix C.1 for more details).

Now we use (5.10) to show that the interval $(-\pi/2, \tan^{-1}(b/g))$ is a subset of the attraction region of $\delta^\star$. For a sufficiently large penalty, the globally optimal solution $\delta^\star$ can be very close to the global minimum of the unconstrained penalized problem in (5.5). Therefore, this is equivalent to showing

$$\mathcal{L}'_\rho(\delta)^T(\delta^\star - \delta) < 0, \forall\ \delta \in (-\frac{\pi}{2}, \tan^{-1}(b/g)). \tag{5.13}$$

As ρ is sufficiently large, the sign of $\mathcal{L}'_\rho(\delta)$ is dominated by the gradient of the second term in (5.5), i.e.,

$$\begin{aligned}\mathcal{L}'_\rho(\delta) \approx& \rho(\ell + g - g\cos(\delta) - b\sin(\delta))(g\sin(\delta) - b\cos(\delta)) \\ =& \rho h(\delta)h'(\delta).\end{aligned}$$

For any $\delta \in (-\pi/2, \tan^{-1}(b/g))$, we have $h'(\delta) < 0$ from (5.11a), which means the function $h(\delta)$ is decreasing on the interval
$(-\pi/2, \tan^{-1}(b/g))$. Also, the global minimum $\delta^\star$ must satisfy $h(\delta^\star) = 0$. Therefore we have

$$\begin{aligned}h(\delta) >& 0, \forall\ \delta \in (-\pi/2, \delta^\star) \\ h(\delta) <& 0, \forall\ \delta \in (\delta^\star, \tan^{-1}(b/g)).\end{aligned}$$

Then the inequality in (5.13) follows from above. Therefore, by Definition 5.1, the interval $(-\pi/2, \tan^{-1}(b/g))$ is a subset of the attraction region of $\delta^\star$.

To obtain the minimizer of $\mathcal{L}_\mu$, we write out the optimality condition of (5.6) for the primal-dual optimal solution $(\hat{\delta}, \hat{\mu})$:

$$(c' + \hat{\mu})g\sin(\hat{\delta}) + (c' - \hat{\mu})b\cos(\hat{\delta}) = 0, \tag{5.16}$$

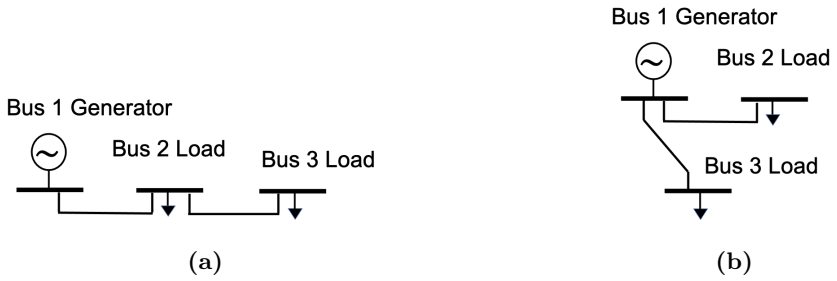


Figure 5.5: The two types of three bus networks with the tree structure.

where c' is a shorthand for $c'(g - g\cos(\hat{\delta}) + b\sin(\hat{\delta}))$ and is the gradient of the cost function. Suppose $\hat{\delta} \in (-\pi/2, 3\pi/2)$, then $\hat{\delta}$ solves

$$\tan^{-1}(\frac{\hat{\mu} - c'}{\hat{\mu} + c'} b/g) + k\pi,\ k = 0, 1, \tag{5.17}$$

where the smaller value is the minimum of $\mathcal{L}_\mu$ and the larger one is the maximum (see Appendix C.2 for more details). Let $\hat{\delta}$ be the minimum, which satisfies $-\pi/2 < \hat{\delta} < \tan^{-1}(b/g)$. Since the interval $(-\pi/2, \tan^{-1}(b/g))$ is a subset of the attraction region of $\delta^\star$, no matter what initial point we start Algorithm 5.1 from, solving the Lagrangian gives us a solution lying in the attraction region around the global minimum. Following from the Capture Theorem in [160], Algorithm 5.1 is able to get out of a strict local solution and reach the global minimum.

Now we induct from 2-bus to 3-bus networks. There are two types of tree topology for a 3-bus network, which are shown in Fig. 5.5. Since the topology in Fig. 5.5(b) is equivalent to two 2-bus networks, we focus on the 3-bus branch in Fig. 5.5(a), where bus 1 is the reference bus.

The ACOPF problem for Fig. 5.5(a) can be written as follows:

$$\min_{\delta_{12}, \delta_{23}} \ c(p_1^g) \tag{5.18a}$$

$$\text{s.t. } \ell_2 + p_{21}^f + p_{23}^f = 0 \tag{5.18b}$$

$$\ell_3 + p_{32}^f = 0. \tag{5.18c}$$

Note that only bus 1 generates power and to deliver the power to bus 3 the power has to

be delivered to bus 2 first. Since the cost function $c(\cdot)$ is increasing, given the load at bus 2 and bus 3, minimizing the power generation cost in (5.1) is equivalent to minimizing the power transfer cost on both lines. Therefore, (5.18) can be decomposed into two parts, and each part is nothing but solving the ACOPF for a 2-bus network with voltage fixed and ignoring reactive power, i.e., the formulation in (5.4). To make this clear, we first rewrite (5.18) as follows:

$$\min_{\delta_{12},\delta_{23}} \; c_1(p_{12}^f) + c_2(p_{23}^f) \tag{5.19a}$$

$$\text{s.t. } \ell_2 + p_{21}^f + p_{23}^f = 0 \tag{5.19b}$$

$$\ell_3 + p_{32}^f = 0. \tag{5.19c}$$

With ℓ_2 fixed, for every given δ_{23}, we can always pick some δ_{12} to satisfy (5.19b). Therefore, we can regard δ_{23} as the optimization variable first. Then the problem (5.19) is reduced to

$$\begin{aligned} &\min_{\delta_{23}} \tilde{c}(p_{23}^f) \\ &\text{s.t. } \ell_3 + p_{32}^f = 0 \end{aligned}$$

where $\tilde{c}(\cdot)$ is some increasing cost function that takes into account the effect of δ_{23} on p_{12}^f. This problem has exactly the same formulation as the 2-bus network in (5.4). As we proved for the 2-bus network, if we start Algorithm 5.1 from a point where δ_{23} is at the local minimum, then we still can get out of this local minimum.

Now we optimize δ_{12} for a given δ_{23}, then we can add p_{23}^f to ℓ_2. Therefore, (5.19) is reduced to

$$\begin{aligned} &\min_{\delta_{12}} c_1(p_{12}^f) \\ &\text{s.t. } \tilde{\ell}_2 + p_{21}^f = 0 \end{aligned}$$

where $\tilde{\ell}_2 = \ell_2 + p_{23}^f$. This problem also has the same formulation as the 2-bus network in (5.4). Therefore, if the initial point is a local minimum for δ_{12}, Algorithm 5.1 still can get out of this local minimum.

Let us assume Theorem 5.2 holds for a $(n-1)$-bus radial network and consider a n-bus radial network. Similar to the proof for a 3-bus network, we can reduce the ACOPF problem for a n-bus network to the case of $(n-1)$-bus. So by induction, Theorem 5.2 holds for the n-bus radial network. ■

5.5.2 Optimizing both voltage magnitudes and angles

In this part, we optimize both voltage magnitudes and angles for a 2-bus network. For simplicity, we ignore the reactive power. Suppose bus 1 is a generator and also the reference (slack) bus with linear cost \$1/MW, and bus 2 is the load bus with load l. The ACOPF in (5.1) can be simplified as

$$\min_{\delta,V_1,V_2} gV_1^2 - V_1V_2(g\cos(\delta) - b\sin(\delta)) \tag{5.22a}$$

$$s.t. \ell + V_2^2 g - V_1V_2(g\cos(\delta) + b\sin(\delta)) = 0 \tag{5.22b}$$

$$\underline{V} \leqslant V_1, V_2 \leqslant \bar{V}. \tag{5.22c}$$

Let us collect all the variables into the vector $\mathbf{x} = (\delta, V_1, V_2)$. We denote the objective function by $f(\mathbf{x})$, and the equality constraint (5.22b) by $h(\mathbf{x}) = 0$. The following theorem gives the property of the Hessian matrix of the Lagrangian.

Theorem 5.3. *Denote the global solution of* (5.22) *as* $\mathbf{x}^\star$ *and the local solution as* $\bar{\mathbf{x}}$*. Then the Hessian matrix of a Lagrangian of* (5.22)*, formed with multipliers at any of the local solutions, is positive definite at* $\mathbf{x}^\star$ *and negative definite or indefinite at* $\bar{\mathbf{x}}$*.*

Proof. To study the solution to (5.22), we look at the equality constraint (5.22b) directly. Its gradient with respect to δ can be written as

$$\partial h/\partial\delta = g\sin(\delta) - b\cos(\delta) = g\cos\delta(\tan(\delta) - b/g).$$

Suppose $\delta \in (-\pi/2, 3\pi/2)$, then $\partial h/\partial\delta$ is zero at $\tan^{-1}(b/g) + k\pi$, $k = 0,\ 1$, where the smaller value is located at the global minimum and the larger value is at the local minimum. Denote the global minimum as $\delta^\star$ and at the local minimum as $\bar{\delta}$. They satisfy (see Appendix C.1 for the details):

$$-\pi/2 < \delta^\star < \tan^{-1}(b/g) < \bar{\delta} < 3\pi/2. \tag{5.23}$$

In Appendix C.3, we show that at least one of V_1 and V_2 need to be binding at a constraint, but both voltages cannot be binding at the same time. This allows us to consider the cases where V_1 is binding or V_2 is binding separately.

First, suppose V_1 is inactive and V_2 is binding. In this case, V_2 is a constant and the Lagrangian of (5.22) can be written as

$$\mathcal{L}_{\lambda,\boldsymbol{\mu}} = f(\mathbf{x}) + \mu h(\mathbf{x}) + \bar{\lambda}_1 \left(V_1 - V_{\max}\right) + \underline{\lambda}_1 \left(-V_1 + V_{\min}\right).$$

The multipliers are associated with some local solution, and μ is the Lagrange multiplier related to the equality constraint, and $\bar{\lambda}_1$ and $\underline{\lambda}_1$ are the multipliers related to the inequality constraints of V_1.

Denote the Hessian matrix of $\mathcal{L}_{\lambda,\boldsymbol{\mu}}$ as $\nabla^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}(\mathbf{x})$. To determine its definiteness, we write out all leading principal minors at a solution $\tilde{\mathbf{x}}$ (see Appendix C.4 for the details):

$$\Delta_1(\tilde{\mathbf{x}}) = \tilde{V}_1 \tilde{V}_2 \frac{-2gb}{g\cos(\tilde{\delta})(\tan(\tilde{\delta}) - b/g)} \tag{5.24a}$$

$$\Delta_2(\tilde{\mathbf{x}}) = 2g\Delta_1(\tilde{\mathbf{x}}). \tag{5.24b}$$

Following from the inequalities in (5.23), both leading principal minors in (5.24) are positive at the global minimum and negative at the local minimum. This means the Hessian matrix at $\mathbf{x}^\star$ is positive definite. In contrast, the Hessian matrix at $\bar{\mathbf{x}}$ is negative definite.

Now we suppose V_2 is inactive and V_1 is binding. In this case, V_1 is a constant and the Lagrangian is:

$$\mathcal{L}_{\lambda,\boldsymbol{\mu}} = f(\mathbf{x}) + \mu h(\mathbf{x}) + \bar{\lambda}_2 \left(V_2 - V_{\max}\right) + \underline{\lambda}_2 \left(-V_2 + V_{\min}\right).$$

where the multipliers are associated with some local solution. Let us denote the Hessian matrix of the Lagrangian as $\tilde{\nabla}^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}(\mathbf{x})$. Its leading principal minors at a feasible solution $\tilde{\mathbf{x}}$ are (see Appendix C.4 for the details):

$$\tilde{\Delta}_1(\tilde{\mathbf{x}}) = \tilde{V}_1 \tilde{V}_2 \frac{-2gb}{g\cos(\tilde{\delta})(\tan(\tilde{\delta}) - b/g)} \tag{5.25a}$$

$$\tilde{\Delta}_2(\tilde{\mathbf{x}}) = 2g\tilde{\mu}\Delta_1(\tilde{\mathbf{x}}). \tag{5.25b}$$

Since the multiplier μ represents the marginal price of consuming each additional unit of load, it is positive at the global minimum. This means $\Delta_2(\tilde{\mathbf{x}})$ has the same sign as $\Delta_1(\tilde{\mathbf{x}})$.

For the global minimum $\mathbf{x}^\star$, $\Delta_1(\tilde{\mathbf{x}})$ is positive from (5.23), hence both leading principal minors in (5.25) are positive and the Hessian matrix is positive definite at $\mathbf{x}^\star$. In contrast, at the local minimum $\bar{\mathbf{x}}$, $\Delta_1(\tilde{\mathbf{x}})$ is negative following from (5.23). Then the Hessian matrix is either negative definite or indefinite at $\bar{\mathbf{x}}$. ■

The simulation results in the next section do not need to make any of the assumptions in Theorem 5.2 and 5.3. They are about mesh networks with all constraints included. Therefore, we suspect the theory can be made much stronger and would extend to larger meshed networks. However, analyzing these cases is challenging and is a future direction for us.

5.6 Simulation Studies

In this section we report the simulation results to validate the effectiveness of our algorithm. The NLP solver used here is IPOPT [1] and the convergence tolerance is set to 0.0001. It returns a feasible solution, which may or may not be a global optimum. We test our algorithm on IEEE meshed networks with 3, 9, 22, 39, 118 and 300 buses, and also on the IEEE radial network with 141 buses. For the 3-bus, 9-bus, 22-bus, 118-bus and 300-bus networks, the local and global solutions are known and listed in [4, 162]. We use the strict local solutions as starting points for the solver to demonstrate the ability of Algorithm 5.1 of getting out of local solutions. For the 39-bus and 141-bus networks, we do an exhaustive search by discretizing each variable within their bounds to find the global solution. The simulation results show that for the 3, 9, 22, 118, 300 and 141-bus networks, Algorithm 5.1 finds the globally optimal solution in 1 iteration. For the 39-bus networks, it takes at most 3 iterations for Algorithm 5.1 to obtain the optimal solution.

5.6.1 3-Bus Mesh Network

The 3-bus network we use is shown in Fig. 5.5a and the voltage bounds are $[0.95, 1.05]$. Two solutions exist and they are listed in Table 5.3. This was an example used in [162] to show that multiple reasonably looking local solutions can exist, and contrary to conventional wisdom, the higher voltage one is the suboptimal one (although the cost difference is small).

If we start the nonlinear solver from an initial point near the second solution, then the solver cannot get out of the attraction basin and always returns the second solution. In contrast, if we launch Algorithm 5.1 using the second solution as a starting point, then the algorithm converges to the first solution (the global solution) after one iteration. Although the cost difference is small between the two solutions, larger networks will have bigger cost differences.

	Bus 1	Bus 2	Bus 3	Cost
Solution 1	$0.95\angle 0$	$0.95\angle -0.48$	$0.98\angle -0.53$	1
Solution 2	$0.95\angle 0$	$1.01\angle -0.46$	$1.05\angle -0.51$	1.0021

Table 5.3: The two local solutions for the 3-bus network in Fig. 5.5a. The cost is normalized to 1 for the global solution.

5.6.2 9-Bus Mesh Network

In the 9-bus network, there are 3 generators (bus 1, 2 and 3) and 9 transmission lines. The voltage bounds are $[0.9, 1.1]$. Four solutions exist. The cost of the worst local solution is 38% more than the cost at the global solution. We also find that the solutions at generators 2 and 3 and load buses 6, 7, and 8 are important to improve the cost. The power transfer along the lines between these buses tend to get stuck at a suboptimal solution, which leads to a cost more than 30% higher than the lowest one. For the nonlinear solver, we need to relaunch it using different initial points in order for these five nodes to get around the attraction basin. This requires many trials. In contrast, Algorithm 5.1 only requires one iteration to achieve the global solution, even starting from the local solution with the highest cost.

5.6.3 22-Bus Mesh Network

In the 22-bus network, the buses are connected in a loop. There are 11 generators and 22 transmission lines. The voltage bounds are $[0.95, 1.05]$. There exist two solutions, and the

	Bus 2	Bus 7	Bus 12	Bus 17	Bus 22	cost
Solution 1	$1.0285\angle -0.045$	$1.05\angle 0$	$1.0285\angle -0.045$	$1.05\angle 0$	$1.0285\angle -0.045$	1
Solution 2	$0.95\angle -0.339$	$1.0145\angle 4.57$	$0.95\angle 3.089$	$1.0145\angle 1.714$	$0.95\angle 0.233$	1.306

Table 5.4: The two solutions for the 22-bus network. We pick five buses and show their voltage and angles. The costs at the two solutions are normalized such that the globally optimal cost is 1.

cost of the local solution is 30% higher than that of the global solution. The two solutions are quite different. We pick 5 buses that are evenly spaced and list their solutions in Table 5.4. Since the two solutions are very different, it is hard for a nonlinear solver to get around the local solution.

Particularly, if we initialize the solver with a flat start, we obtain the strict local solution. Furthermore, we generate 100 random points uniformly at random within the bounds of each variable. If these points are used to initiate the nonlinear solver, the local solution is always obtained and the global one cannot be reached. In comparison, Algorithm 5.1 can achieve the global solution after one iteration regardless of the initial point. This is an example where using random search is very computationally inefficient, and our deterministic algorithm turns out to be much more successful.

5.6.4 39-Bus Mesh Network

In the 39 bus network, there are 10 generators and 46 transmission lines. The voltage bounds are $[0.95, 1.05]$. Unlike the previous smaller networks, the number and the cost of the solutions are not previously known for this network. Therefore we conducted an exhaustive search to find the global solution. To evaluate the effectiveness of Algorithm 5.1, we choose 600 random points within the bounds of each variable using the uniform distribution. Then we start Algorithm 5.1 with these random points to observe the improvement of the solution quality.

In Fig. 5.6, we plot the fraction of global solutions in the set of all 600 results after each iteration. The x-axis represents the number of iterations that Algorithm 5.1 is ran, and y-axis represents the percentage of globally optimal solutions after each iteration. When

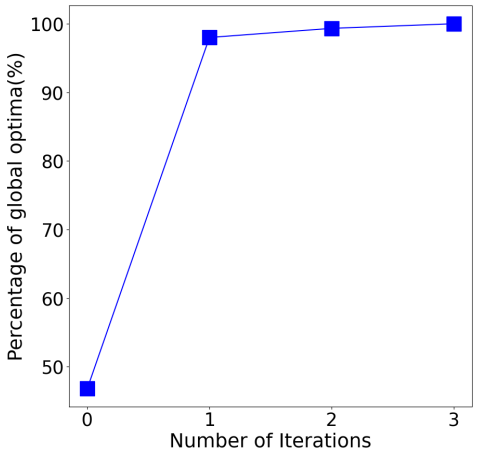


Figure 5.6: Percentage of globally optimal solutions for the 39-bus network after each iteration. Using a set of random starting points, 47% of them leads to the global optima after a direct call to IPOPT. The fraction of global optimal solutions increases to 98%, 99.93% and 100% after running one, two and three iterations of Algorithm 5.1 , respectively.

we make a direct call to the solver, less than half of the solutions are globally optimal. One application of Algorithm 5.1 increases the percentage of globally optimal solutions to 98%. After two iterations, only four cases are not globally optimal. When we run Algorithm 5.1 for three iterations, all solutions are globally optimal.

We also calculate the average cost of the 600 solutions after each iteration of Algorithm 5.1 and plot the result in Fig. 5.7. The x-axis is the number of iterations of running Algorithm 5.1 , and y-axis represents the average cost of 600 solutions, which is normalized using the optimal cost as the factor. After a direct call to the solver, the average cost is 30% higher than the optimal cost. As Algorithm 5.1 is ran, the average cost decreases quickly. After one iteration, the average cost is only 1.5% more than the globally optimal cost, and after three iterations all solution are at the global optimum.

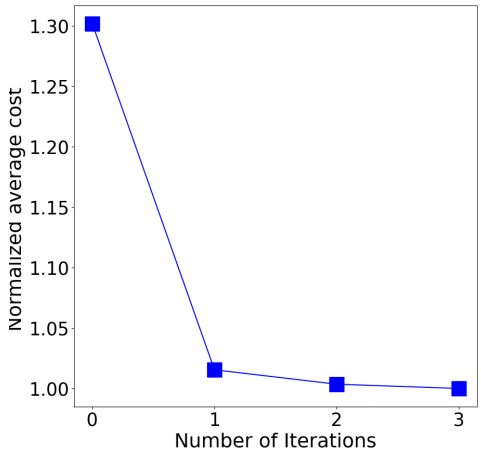


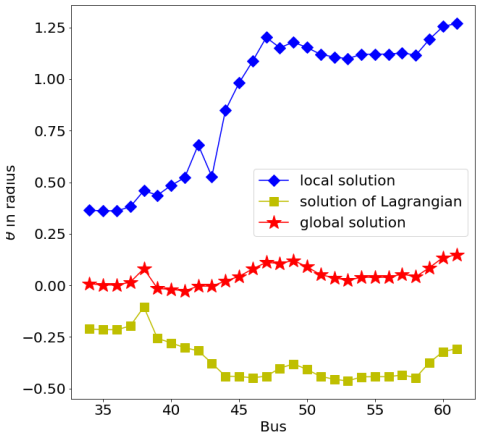
Figure 5.7: The average cost of all 600 solutions for the 39-bus network after each iteration. The cost is normalized such that the optimal cost is 1. After a direct call to IPOPT, the average cost is 30% higher than the optimal cost. Then the average cost reduces to 1.5%, 0.4% higher than the optimal value after running one and two iterations of Algorithm 5.1 , respectively. After three iterations, the average cost is exactly the optimal cost.

5.6.5 118-Bus Mesh Network

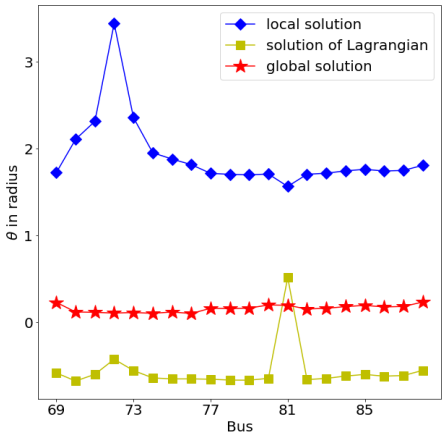
The topology of the 118-bus meshed network can be found in [4]. There are 54 generators and 186 transmission lines. When the voltage bounds are $[0.94, 1.06]$ and power generation bounds are scaled by 4, two locally optimal solutions are known and listed in [4]. The strictly local minima has a cost that is 28.8% higher than the global minimum.

To show how the proposed algorithm can escape from the strictly local minima by using the solution to the partial Lagrangian as a warm start, we choose two subsets of buses from the 118-bus meshed network and plot the angle of the voltage solutions for each subset. The angle solutions for the subset composed of buses from 34 to 61 are plotted in Fig. 5.8a, and for the subset composed of buses from 69 to 88 plotted in Fig. 5.8b.

When the NLP solver reaches the strictly local solution (marked as blue diamonds) from some initialization point, it terminates at this solution. To escape from this suboptimal solution, we call the solver to solve the partial Lagrangian starting from the same initialization



(a) Angles of solutions for buses from 35 to 61.



(b) Angles of solutions for buses from 69 to 88.

Demonstration of how the proposed algorithm can escape from the strictly local minima for the 118-bus meshed system. Use two subsets of buses to show the relationship between the local minimum, the solution of the Lagrangian and the global minimum solution (Figures (a) and (b)).The solution to the partial Lagrangian (yellow squares) provides a good warm start for the solver to escape from the strictly local minima. Using the yellow squares as initialization points, the NLP solver is able to find the globally optimal solutions (red stars).

point and get the solution marked as yellow square. We use this solution as the new warm start to solve the primal problem again, then we can obtain the global minima (marked as red stars) with a single run of Algorithm 5.1 . As is shown in Fig. 5.8a and Fig. 5.8b, the solution to the partial Lagrangian jumps quite far away from the local solution but stay close to the global solution. Therefore, it provides a good initialization point for the NLP solver to escape from the strictly local minima and get to the global minima.

5.6.6 300-Bus Mesh Network

The topology of the 300-bus meshed network can also be found in [4]. There are 69 generators and 411 transmission lines. When the voltage bounds are $[0.93, 1.07]$ and the reactive

	Bus 1	Bus 30	Bus 50	Bus 80	Bus 121	relative cost
PG(MW) at solution 1	3.2578	5.7791	7.9975	5.0351	9.768	1
PG(MW) at solution 2	3.664	4.7109	8.2127	3.8811	13.8361	1.115
PG(MW) at solution 3	4.3365	5.1223	7.039	4.9725	9.5491	1.026

Table 5.5: The active power generation at a subset of generator buses at the three optimal solutions for the 141-bus radial network. The costs are normalized such that the global optimal cost is 1.

power generation lower bounds are tightened to -100 Mvar for all generator buses, two locally optimal solutions are known and listed in [4]. The strictly local minimum has a cost that is 0.62% higher than the global minimum. For the NLP solver to escape from the strictly local solution, only a single run of Algorithm 5.1 is needed.

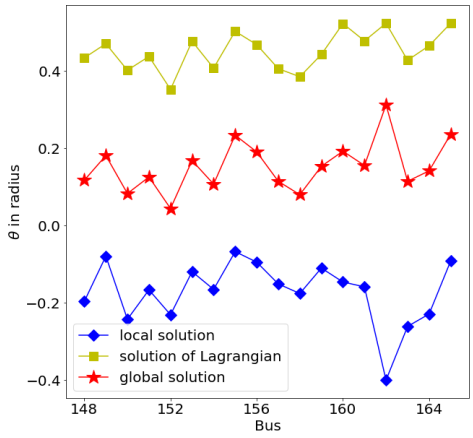


Figure 5.8: Demonstration of how the proposed algorithm can escape from the strictly local minima for the 300-bus meshed system and plot of angle solutions for the subset of buses from 148 to 166. The solution to the partial Lagrangian (yellow squares) jump far away from the strictly local minima (blue diamonds). As a result, starting from the yellow squares, the NLP solver is able to escape from the strictly local solution and find the globally optimal solution (red stars).

To show how the proposed algorithm can escape from the strictly local minimum by using the solution to partial Lagrangian as an initialization point, we choose a subset of

buses from 148 to 166 and plot the angles of their voltage solutions in Fig. 5.8. As Fig. 5.8 shows, the solution to the partial Lagrangian (marked as yellow squares) jumps far away from the strictly local solution and thus act as a good warm start for the NLP solver to escape from the strictly local minima and find the global minima.

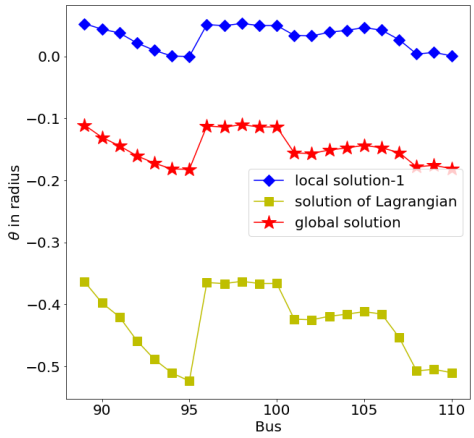
5.6.7 141-Bus Radial Network

The topology of the 141-bus radial network can be found in [163]. There are 9 generators and 140 transmission lines. When the voltage bounds are $[0.83, 1.17]$ and the reactive power generation lower bounds are tightened for buses 2, 7 to 5 Mvar and bus 5 to 15 Mvar, 3 locally optimal solutions are found. The two strictly local minima has costs that is 11.5% and 2.6% higher than the global minimum, respectively. The active power generation at a subset of the generator buses for each of the solutions are shown in Table 5.5.

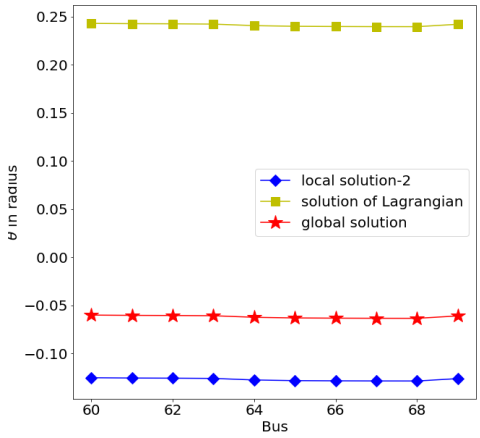
To test the performance of our algorithm and demonstrate how it escapes from local minima, we randomly generate 1000 initial points within the bounds of each variable using a uniform distribution, and call the NLP solver to solve the primal AC OPF problem starting from these initial points.

To demonstrate how the solution of partial Lagrangian can be a good warm start, we choose a subset of buses from the 141-bus radial network and plot the angle of the voltage solutions. First, we show how the proposed algorithm escapes from the first local minimum. In Fig. 5.9a, the NLP solver reaches local solution-1 (marked as blue diamonds) from an initialization point. The NLP solver terminates at this solution. To escape from this suboptimal solution, we call the solver to solve the partial Lagrangian starting from the same initialization point and get the solution marked as yellow square. We use this solution as a new warm start to solve the primal problem again, then we can obtain the global solution (marked as red stars) with only a single run of Algorithm 5.1 . As Fig 5.9a shows, the solution to the partial Lagrangian jumps quite far from the local minimum and provides a good starting point for the NLP solver to reach the global minimum.

In Fig. 5.9b, we show how our proposed algorithm escapes from the second local minimum solution. For an initialization point where the NLP solver reaches local solution-2,



(a) Angles of solutions for buses from 89 to 110.



(b) Angles of solutions for buses from 60 to 69.

Demonstration of how the proposed algorithm can escape from the strictly local minima for the 141-bus radial system. Specifically, Figure (a) shows how the solutions escape from the first local minimum and (b) shows how they escape from the second minimum. Using our proposed algorithm, the solution of the partial Lagrangian is able to escape the local minimum (blue diamonds). Using these (yellow squares) as initialization points, the NLP solver is able to find the globally optimal solution (red stars).

we solve the partial Lagrangian. Initializing the NLP solver from the solution of the partial Lagrangian allows us to reach the global solution, and we show the voltage angles of different solutions for buses 60 to 69 in Fig. 5.9b.[3]

5.7 Conclusions

In this chapter, we propose a simple algorithm to iteratively improve the solution quality of ACOPF problems. First, we solve the ACOPF problem using an existing nonlinear solver. From the solution and its associated dual variables, we construct a partial Lagrangian

[3]The reason why we choose different subsets is that the local and global solutions in the subset composed of buses 89 to 110 are very similar. To better illustrate the effectiveness of our method, we choose the subset composed of buses 60 to 69, where the local and global solutions are quite different.

by dualizing the power balance equations. Optimizing this partial Lagrangian leads to a new solution. With this solution as an initial point, we again call the solver for the ACOPF problem. By repeating these steps, we can iteratively improve the solution quality, escaping from local solutions to find better ones. We illustrate the intuition behind our algorithm using 2 and 3-bus networks, which shows that the partial Lagrangian has a flatter optimization landscape compared to the original primal problem. We prove that the algorithm is guaranteed to work in tree networks. Theoretical analysis for more general networks is an important part of our future work. We validate the effectiveness of our algorithm on standard 9-bus, 22-bus, 39-bus, 118-bus and 300-bus mesh networks and also on the IEEE 141-bus radial network. Regardless of the initial points, our algorithm always finds the global optimum within at most three iterations.

Chapter 6

LEARNING TO SOLVE THE AC OPTIMAL POWER FLOW VIA A LAGRANGIAN APPROACH

6.1 Introduction

In this chapter, we present a machine learning architecture that overcomes the challenge of multiple suboptimal local solutions in the training data set. Instead of focusing on the load/solution pairs, we can think of these machine learning methods to be producing a good *warm start* for a solver, and learning a good warm start would offer significant computational speedups and improvement on solution quality [160].

We first learn a neural network that maps load to the dual variable of the power balance constraints. These dual variables are the locational marginal prices and would be readily available from any modern nonlinear solver. These dual variables are used to form a partial Lagrangian, whose solution we also learn via a neural network. Then we use the predicted solution of the partial Lagrangian as a warm start [5]. Interestingly, this warm starting point tends to be closer to the globally optimal solution of the ACOPF, even if the training data set only has suboptimal solutions or a mixture of global and local solutions. Therefore, by using the learned warm start as an initialization point, we could have better solution quality than directly learning based on load/solution pairs.

We show that our duality-based approach outperforms existing approaches on modified IEEE 22-, 39-, and 118-bus networks [4]. The difference is especially significant if the training data set contains some strictly suboptimal solutions.

6.2 Problem Formulation

6.2.1 ACOPF Formulation

Consider a power system network of n buses and m lines. For bus i, let V_i denote its voltage magnitude, δ_i its angle, p_i^g and q_i^g the active and reactive output of the generator and p_i^d

and q_i^d the active and reactive load. The admittance between i and j is $g_{ij} - jb_{ij}$, the active power and reactive power flow from i to j is p_{ij}^f and q_{ij}^f and δ_{ij} is used as a shorthand for $\delta_i - \delta_j$.

The ACOPF problem is [4]:

$$\min_{\mathbf{V},\boldsymbol{\delta}} \textstyle\sum_i c_i(p_i^g) \tag{6.1a}$$

$$\text{s.t. } p_i^g = p_i^d + \textstyle\sum_{j:(i,j)\in\mathcal{E}} p_{ij}^f \tag{6.1b}$$

$$q_i^g = q_i^d + \textstyle\sum_{j:(i,j)\in\mathcal{E}} q_{ij}^f \tag{6.1c}$$

$$p_{ij}^f = V_i^2 g_{ij} - V_i V_j (g_{ij}\cos(\delta_{ij}) - b_{ij}\sin(\delta_{ij})) \tag{6.1d}$$

$$q_{ij}^f = V_i^2 \hat{b}_{ij} - V_i V_j (b_{ij}\cos(\delta_{ij}) + g_{ij}\sin(\delta_{ij})) \tag{6.1e}$$

$$\underline{V}_i \leqslant V_i \leqslant \bar{V}_i \tag{6.1f}$$

$$\underline{p}_i^g \leqslant p_i^g \leqslant \bar{p}_i^g \tag{6.1g}$$

$$\underline{q}_i^g \leqslant q_i^g \leqslant \bar{q}_i^g \tag{6.1h}$$

$$(p_{ij}^f)^2 + (q_{ij}^f)^2 \leqslant (S_{ij}^{\max})^2 \tag{6.1i}$$

where $\hat{b}_{ij} = b_{ij} + 0.5 b_{ij}^C$ and b_{ij}^C is the line charging susceptance. The constraints (6.1b) and (6.1c) enforce power balance, (6.1d) and (6.1e) are the AC power flow equations, (6.1f) limits the bus voltage magnitudes, (6.1g) and (6.1h) represent the active and reactive limits and 6.1i are the line flow limits.

The problem in 6.1 is nonconvex and can have multiple local solutions. More precisely, local solutions are all the solutions that satisfy local optimality conditions, for example, the KKT conditions or second order ones [160]. Out of this set, the solutions with the lowest cost are called the global ones. We sometimes refer to the local solutions that are not global as strict local solutions.

6.2.2 Challenges of Using Learning for ACOPF

Machine learning is often applied to the optimization problem in 6.1 by viewing it as the mapping from the demands $\mathbf{p}^d$ and $\mathbf{q}^d$ to the solutions $\mathbf{V}$ and $\boldsymbol{\delta}$, with the goal of finding a proxy function to this mapping. Training data is generated by solving 6.1 for a number of

demands and collecting the corresponding solutions. Several different learning architectures have been proposed, including direct regression [164], using sensitivity information [165], and emulation of an iterative solver [166]. The constraints in 6.1b to 6.1i are nontrivial to enforce, and an additional call of power flow or optimal power flow on a smaller problem are often used [166, 167].

However, existing learning methods face a common challenge, stemming from the fact that the ACOPF problem in 6.1 is nonconvex and may have multiple solutions [4, 144, 3]. A number of recent works have shown that for reasonable operation conditions, there could be multiple solutions that differ significantly in cost, but all have practical values (e.g., with voltages being all close to 1 p.u.) [139, 140, 141, 147].

Many nonlinear programming (NLP) solvers have been developed for the ACOPF problem, and their speed and efficiency have improved dramatically (e.g., see [135] and the references within). But NLP solvers are typically only able to return one of the feasible solutions, and there is generally no way to tell whether these feasible solutions are globally optimal. Therefore, using the solutions returned by NLP solvers as ground truth data for training may fundamentally limit the performance of the learning algorithm.

The solution returned by an NLP solver is also sensitive to a variety of factors, and small changes in demand can lead to a large change in the solution. For example, small changes in demand can lead to the solution switching between global and local. Therefore, a data set created directly using the solutions returned by NLP solvers may very well consist of a mixture of local and global solutions, which tend to be very confusing for a learning algorithm.

6.3 Algorithm

In this section, we describe our learning approach to find more optimal solutions to ACOPF problems using neural networks, even when the training data set contains a mixture of local and global solutions.

6.3.1 Lagrangian-based Approach

In chapter 5, we gave an iterative approach to improve the solution quality by alternatively solving (6.1) and its partial Lagrangian. The partial Lagrangian for (6.1) is formed by dualizing the active and reactive power balance constraints 6.1b and 6.1c. Suppose the Lagrangian multipliers associated with 6.1b and 6.1c are μ_i^P and μ_i^Q, respectively, then the partial Lagrangian for 6.1 is:

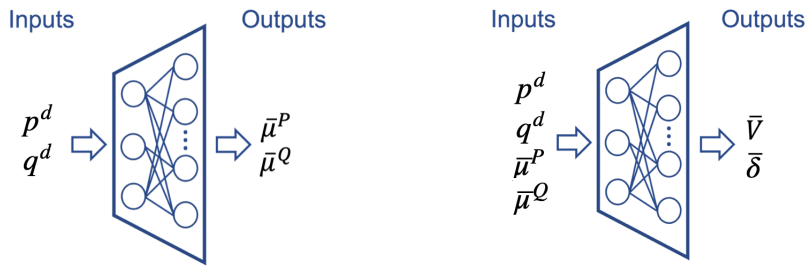
$$\begin{aligned}\mathcal{L}(\mathbf{V}, \boldsymbol{\delta}, \boldsymbol{\mu}^P, \boldsymbol{\mu}^Q) = &\sum_i c_i p_i^g + \sum_i \mu_i^P (p_i^d + \sum_{j:(i,j)\in\mathcal{E}} p_{ij}^f - p_i^g) \\ &+ \sum_i \mu_i^Q (q_i^d + \sum_{j:(i,j)\in\mathcal{E}} q_{ij}^f - q_i^g). \qquad (6.2a) \\ \text{s.t. } &(6.1d) - (6.1i).\end{aligned}$$

It turns out the solution of the partial Lagrangian in (6.2) tends to be close to the global optimal solution of the original ACOPF in (6.1). This is true even if the multipliers μ^P and μ^Q are the ones associated with the strict local solutions (see [5] for more details). Therefore, the solution of (6.2) serves as a good warm start point for an ACOPF solver.

In this chapter, we train two neural networks to replace explicitly solving (6.1) and (6.2). The first one predicts the dual variables of the active and reactive power balance constraints from the load, and the second one to predicts the solution of (6.2) from the load and the predicted multipliers using the first neural network. The output of the second neural network is used as a warm start point for an ACOPF solver. Since this starting point is close to the global solution, the ACOPF solver is solved much faster and is more likely to find the global solution than a solver with a flat or random start.

6.3.2 Training of Neural Networks

The architectures of the two neural networks are shown in Fig. 6.1. The first neural network takes the active and reactive load demands $(\boldsymbol{p}^d, \boldsymbol{q}^d)$ as the input, and the output is the predicted multipliers, denoted by $(\bar{\boldsymbol{\mu}}_P, \bar{\boldsymbol{\mu}}_Q)$. Let $\mathbf{x}$ be the collection of voltage magnitudes and angles, $\boldsymbol{\mu}$ be the collection of $(\boldsymbol{\mu}_P, \boldsymbol{\mu}_Q)$, and $(\mathbf{x}^i, \boldsymbol{\mu}^i)$ be the i-th pair of data in the training set, then the first neural network, denoted as f_{opf} and parameterized by $\boldsymbol{\theta}_{opf}$, is



(a) Learn the dual variables of the power balance constraints.

(b) Predict the solutions of the partial Lagrangian.

Figure 6.1: The two neural networks to be trained.

trained by minimizing:

$$\min_{\boldsymbol{\theta}_{opf}} \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\mu}^i - f_{opf}(\mathbf{x}^i; \boldsymbol{\theta}_{opf}))^2. \tag{6.3}$$

The second neural network emulates solving the partial Lagrangian in (6.2). The input for the second neural network is the active and reactive load demands $(\boldsymbol{p}^d, \boldsymbol{q}^d)$, as well as the associated dual variable solutions $(\boldsymbol{\mu}^P, \boldsymbol{\mu}^Q)$. The output of the neural network is the solution to (6.2), denoted by $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$. Let $\mathbf{z}$ be the collection of inputs for the second neural network, $\mathbf{y}$ be the collection of outputs, and $(\mathbf{z}^i, \mathbf{y}^i)$ be the i'th pair of data in the training set, then the second network, denoted by f_{dual} with weights $\boldsymbol{\theta}_{dual}$ is trained by minimizing

$$\min_{\boldsymbol{\theta}_{dual}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y}^i - f_{dual}(\mathbf{z}^i; \boldsymbol{\theta}_{dual}))^2. \tag{6.4}$$

6.3.3 Making Predictions in Real-time

After training, we use the process in Fig. 6.2 to make predictions. We use the first trained neural network to predict the dual variables $(\bar{\boldsymbol{\mu}}^P, \bar{\boldsymbol{\mu}}^Q)$ from the input load. Then from the predicted dual variable solutions, we use the second trained neural network to predict the solutions $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$ of the Lagrangian. Then we call the NLP solver to solve (6.1) using $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$ as the initial point. This learning algorithm is summarized below as Algorithm 6.1.

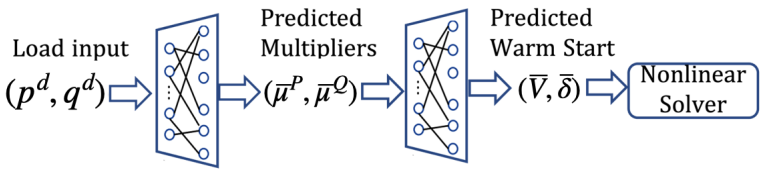


Figure 6.2: Outline of the solution process.

Algorithm 6.1: Solving ACOPF using learning

Inputs: $(\mathbf{p}^d, \mathbf{q}^d)$

First trained neural network to predict multipliers:

$$f_{opf}(\mathbf{p}^d, \mathbf{q}^d; \boldsymbol{\theta}_{opf}) \longrightarrow (\bar{\boldsymbol{\mu}}^P, \bar{\boldsymbol{\mu}}^Q)$$

Second trained neural network to predict solutions to (6.2):

$$f_{dual}(\mathbf{p}^d, \mathbf{q}^d, \bar{\boldsymbol{\mu}}^P, \bar{\boldsymbol{\mu}}^Q; \boldsymbol{\theta}_{dual}) \longrightarrow (\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$$

Call NLP solver for (6.1) initialized at $(\bar{\mathbf{V}}, \bar{\boldsymbol{\delta}})$;

Outputs: Solutions $(\hat{\mathbf{V}}, \hat{\boldsymbol{\delta}})$ to (6.1).

Table 6.1: Algorithm for solving ACOPF using learning.

In our approach, even when the training data set contains suboptimal solutions, the solution of the Lagrangian would be a good warm start that makes the NLP solver get around being trapped at strictly local solutions. The reason is that the partial Lagrangian in (6.2) has "nice" geometry: It has minimums that are near the globally optimal solution of the ACOPF problem, even if it is formed with the multipliers obtained at local optimal solutions of the ACOPF problem [5].

Also, our algorithm is robust in the sense that the solutions to partial Lagrangian are not sensitive to the variations in $\bar{\boldsymbol{\mu}}$. That is, even if the multipliers $\bar{\boldsymbol{\mu}}$ associated with different local solutions are different, the resulting solutions to partial Lagrangian do not change much. This also means we do not ask the predictions of the neural network to be very accurate. Therefore, the training set for our algorithm need not be very large. We sketch the geometric intuitions behind our algorithm in Section 6.4. We validate our

algorithm in standard and modified IEEE benchmark systems, and report simulation results in Section 6.5.3.

6.4 Geometry and Intuition

In this section, we use a 2-bus network as an example to shed some light on why Algorithm 6.1 might learn more globally optimal solutions, even when the training data consists of a lot of local solutions. In the 2-bus network, for simplicity, we ignore the reactive power and set both voltage magnitudes to be 1 per unit. Suppose bus 1 is a generator and also is the reference bus with an increasing cost function $c(\cdot)$, and bus 2 is the load bus with angle $-\delta$. The line admittance is $g - jb$. Given a load of ℓ at bus 2 and ignoring all constraints except for the load balancing one, the ACOPF in (5.1) becomes

$$\min_{\delta} \; c(g - g\cos(\delta) + b\sin(\delta)) \tag{6.5a}$$

$$\text{s.t. } \ell + g - g\cos(\delta) - b\sin(\delta) = 0. \tag{6.5b}$$

This is an example of an OPF with a disconnected feasible space, since there are two distinct solutions to (6.5b) and we are asking for the lower cost one.

To see how an NLP solver would approach this problem, we adopt the common practice in nonlinear programming and form a penalized version of (6.5) [160, 146]. The penalized unconstrained problem is given by

$$\begin{aligned} \mathcal{L}_\rho = & c(g - g\cos(\delta) + b\sin(\delta)) \\ & + \frac{\rho}{2}(\ell + g - g\cos(\delta) - b\sin(\delta))^2, \end{aligned} \tag{6.6}$$

where ρ is a penalty parameter. For large enough ρ, the solutions of (6.6) would be very close to those of (6.5) [160]. The function $\mathcal{L}_\rho$ is plotted in Fig. 6.3. We can see that there are two local minima, with the left one being global. However, both minima satisfy first- and second-order optimality conditions. Therefore, if an NLP solver is initialized with a poor starting point, it would be stuck at the strict local solution.

Now suppose μ is the multiplier corresponding to the equality constraint (6.5b) at *the*

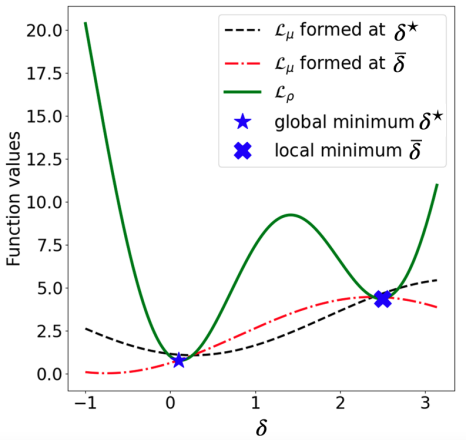


Figure 6.3: Geometry of the penalized objective function $\mathcal{L}_\rho$ and the partial Lagrangian $\mathcal{L}_\mu$. The line admittance is $g - jb$ and the penalty parameter is 2. The red curve is the partial Lagrangian formed at the strict local solution and the black curve is formed at the global solution.

strict local solution. The partial Lagrangian of (6.5) by dualizing (6.5b) is:

$$\begin{aligned}\mathcal{L}_\mu =& c(g - g\cos(\delta) + b\sin(\delta)) \\ &+ \mu(\ell + g - g\cos(\delta) - b\sin(\delta)).\end{aligned} \tag{6.7}$$

Since the sinusoidal functions are periodic with period 2π, let us consider the range $\delta \in [-\pi, \pi]$. It is interesting now to compare the solution of $\mathcal{L}_\mu$ to the original problem in (6.5) (or equivalently, $\mathcal{L}_\rho$). The red curve in Fig. 6.3 plots $\mathcal{L}_\mu$ at the local minimum and the black curve at the global minimum. We can observe an interesting fact that the minimum of $\mathcal{L}_\mu$ is close to the global minimum of $\mathcal{L}_\rho$, even when the multiplier at the strict local solution is used.

It is instructive to think about a training data set with both local and global solutions for the same load, and compare the learned warm starts using direct regression and Algorithm 6.1. Suppose a regression method is used to minimize the distance between a predicted solution and the solutions in the training set. Since a mixture of local and global solutions are used in training, the learned neural network would make a prediction that is the average of the two solutions, as shown in Fig. 6.4. But it may be closer to the local solution rather than the global one. Preprocessing the training data may alleviate some of these issues, but

that is likely to be cumbersome and removes some of the appeals of using machine learning.

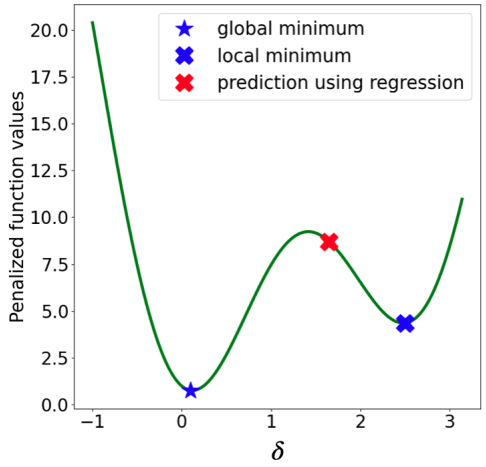


Figure 6.4: When the training set has both local and global solutions, the predicted warm start using direct regression is marked as red, which is closer to the strict local minimum. A solver initialized with this warm start would converge to the strict local minimum.

When Algorithm 6.1 is used, we first predict the multipliers from the load. Since the training set is a mix of local and global solutions, the predicted multiplier would be some point lying between the locally and globally optimal values. The predicted multiplier is plotted in Fig. 6.5a, where we adopt a linear cost function for $c(\cdot)$ in (6.7) and set the cost coefficient to be one, i.e., $c(x) = x$. Then we predict the solution of $\mathcal{L}_\mu$ from the predicted multiplier. For a given multiplier $\bar{\mu}$, the solution to $\mathcal{L}_\mu$ is found through the optimality condition of (6.7):

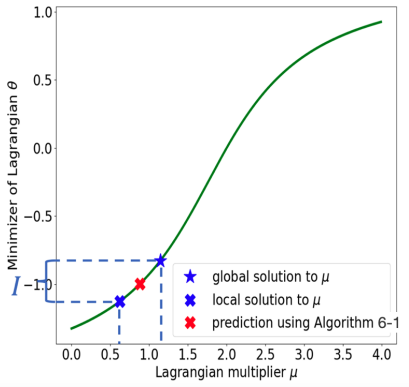
$$(c' + \bar{\mu})g\sin(\bar{\delta}) + (c' - \bar{\mu})b\cos(\bar{\delta}) = 0, \tag{6.8}$$

where c' is a shorthand for the derivative $c'(g - g\cos(\bar{\delta}) + b\sin(\bar{\delta}))$. By varying the multipliers, we can represent the mapping from the multipliers to the solutions of $\mathcal{L}_\mu$ as follows:

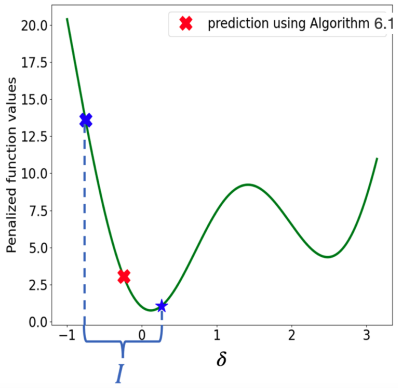
$$\bar{\delta} = \tan^{-1}\left(\frac{\bar{\mu} - c'}{\bar{\mu} + c'} b/g\right), \tag{6.9}$$

which is plotted in Fig. 6.5a. The set of solutions of $\mathcal{L}_\mu$ that is mapped from the multipliers varying between the locally and globally optimal values is denoted by set I. The predicted solution of $\mathcal{L}_\mu$ would lie in set I. We also plot set I in Fig. 6.5b. We can see that every

point in set I is close to the global minimum of $\mathcal{L}_\rho$. More precisely, every point is in the basin of attraction of the global solution. This means if we use the predicted solution of $\mathcal{L}_\mu$ as a warm start, the solver would converge to the global minimum.



(a) The solutions of the partial Lagrangian as a function of the predicted multipliers. When the training data set is a mix of local and global solutions, the predicted multiplier and the associated solution of the partial Lagrangian are marked as red. The set I corresponds to the set of all possible solutions.



(b) The learned warm start point using Algorithm 6.1 is marked as red, which is close to the global minimum of the ACOPF problem. If this predicted warm start is used, the solver would converge to the global solution.

Figure 6.5: Use Algorithm 6.1 to learn a warm start point for the ACOPF solver.

In the next section, we test Algorithm 6.1 on IEEE benchmark systems and show the intuition developed in this section is true for much larger and more complex problems.

6.5 Simulation Results

In this section, we demonstrate the simulation results of using Algorithm 6.1 to predict solutions to the ACOPF problem. We test our algorithm on IEEE networks with 22, 39, and 118 buses. The full specifications for these networks can be found in [4] and [162]. The popular solver IPOPT [1] is used to generate training samples for each network. For a comparison baseline, we use the method in [166], where a deep neural network is trained to

learn the mapping from load to optimal generation values by minimizing the loss between the learned and ground-truth values. Then power flow equations are solved to recover and ensure the feasibility of the overall ACOPF solutions.

Our method can obtain globally optimal solutions even when the training data only contains local solutions and using the warm starts learned by our algorithm can speed up the computation time of solving ACOPF problems using IPOPT.

We use fully-connected neural networks with 2 hidden layers for both Algorithm 6.1 and the baseline method. For the baseline method, the activation function of the neural network is sigmoid for all layers. For Algorithm 6.1 , we use ReLU as an activation function except for the output layer, where a linear activation function is used. All neural network models are implemented using the Tensorflow software library.

6.5.1 22-bus Network

In the 22-bus network, there exist two solutions for a given load, where the cost of the local solution is 30% higher than that of the global solution. We generate the training data by varying the load around the nominal value. For each given load, we solve the ACOPF problem using IPOPT to obtain both solutions (this is done by using a number of random initial points). Then we construct 5 different training sets by adjusting the proportion of strictly local solutions in the data. There are 4000 training samples. We use 90% of them for training and 10% for testing.

The generation costs of the obtained solutions using both methods on different training sets are reported in Fig. 6.6, where the generation costs are represented proportionally to the globally optimal cost. In Fig. 6.6, as the proportion of strictly local solutions in the training set increases, the predicted cost using the baseline method also increases and is larger than the globally optimal cost on every training set. In contrast, Algorithm 6.1 is able to obtain the global solution, even when the training set is comprised only of strictly local solutions. This implies that Algorithm 6.1 is not sensitive to the quality of the training set, and local solutions can also be useful. This observation carries over to larger networks.

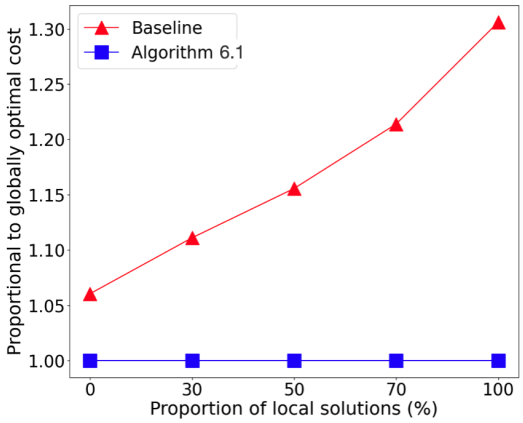


Figure 6.6: Normalized generation costs of the obtained solutions using Algorithm 6.1 and baseline algorithm for the 22-bus network as the proportion of strictly local solutions in the training set increases. Algorithm 6.1 is not sensitive to the quality of training data and is able to obtain the global solution, where the performance of the baseline learning method degrades as the training data quality degrades.

6.5.2 39-bus Network

A key benefit of using machine learning for ACOPF is to speed up computation. Here, we take the IEEE 39-bus network and compare the solution speed of using Algorithm 6.1 to that of directly using IPOPT on random (Gaussian) initial points. We evaluate the computation time on Macbook Pro with Intel Core i5 8259U CPU @ 2.30GHz.

We call IPOPT with both initializations for 200 instances and report the computation time for each instance in Fig. 6.7. The computation time of using the learned warm starts given by Algorithm 6.1 is plotted as the blue line, which is much faster than the random initialization (red) almost for every instance. Note that the neural networks used in Algorithm 6.1 are feed-forward functions, and their evaluation time (sub-milliseconds) is negligible for the comparison in Fig. 6.7.

6.5.3 118-bus Network

For the 118-bus network, there exist three solutions for a given load. The worst cost is 39% higher than the globally optimal cost. We generate the training data by varying the load

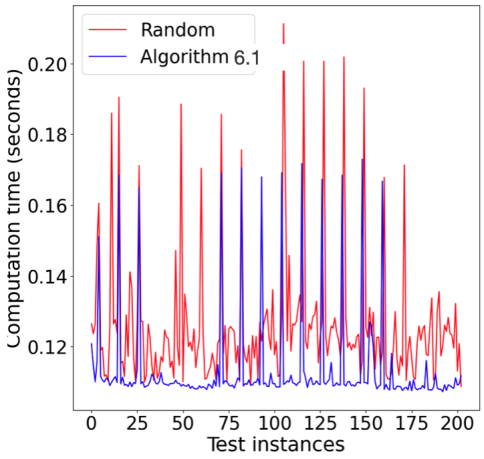


Figure 6.7: Computation time of calling IPOPT to solve the ACOPF problem in 39-bus network with different initialization. The blue curve is the computation time when the warm starts learned by Algorithm 6.1 are used as initial points, which is lower than the random initialization (red curve) almost for every instance.

around the nominal value. For each given load, we solve the ACOPF problem using IPOPT to obtain multiple solutions (this is done by using a number of random initializations). Then we construct 3 different training sets with different proportions of strict local solutions as shown in Fig 6.8. There are 1900 training samples and 90% used for training and 10% for testing. In Fig. 6.8, we compare the generation costs, which are normalized to the global optimal value, of the obtained solutions using Algorithm 6.1 to that predicted by the baseline method.

The predicted cost using the baseline method is larger than the globally optimal cost on every training set, and increases as the quality of the training data degrade (the proportion of local solutions increases).

In contrast, Algorithm 6.1 is able to obtain the globally optimal cost regardless of the quality of the training data. Even when the training data only contains the solutions with the highest cost, the predicted cost using Algorithm 6.1 is globally optimal. This confirms with the intuition in Section 6.4 that the predicted solutions of the partial Lagrangian would be close to the global minimum of the ACOPF problem, and hence could be good warm

starts for the solver to reach the global solution.

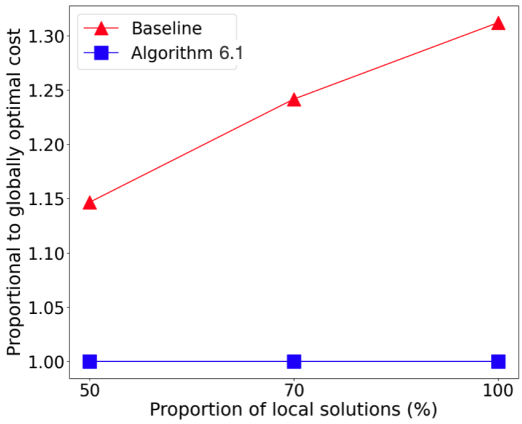


Figure 6.8: Generation costs of the obtained solutions using Algorithm 6.1 and the baseline method on different training sets for the 118-bus network. All the generation costs are represented proportionally to the globally optimal cost. Algorithm 6.1 is able to obtain the global solution even when the training data only consists of local solutions.

6.6 Conclusion

In this chapter, we propose a partial Lagrangian-based learning approach to predict solutions of the ACOPF problem. First, we use a neural network to learn dual variables of the ACOPF problem. Then we use a second neural network to predict solutions of the partial Lagrangian from the predicted dual variables. Using the predicted solutions of the partial Lagrangian as warm starts, the ACOPF solver can reach more globally optimal solutions. We validate the effectiveness of our algorithm on standard 22-bus, 39-bus and 118-bus networks, and show our algorithm is able to obtain the globally optimal solutions and offer significant speedups.

Chapter 7

CONVEX RESTRICTION OF FEASIBLE SETS FOR AC RADIAL NETWORKS

7.1 Introduction

Many problems in power systems determine an optimal network operating point that seeks to minimize a certain objective, all while satisfying a set of power flow equations and engineering constraints, such as optimal power flow (OPF), state estimation, voltage regularization, etc. However, the inherent nonlinearity of power flow equations gives rise to nonconvex feasible sets for these problems, which makes even finding feasible solutions nontrivial [168].

One strategy to address the nonconvexity of OPF problems is to convexify the constraints. This results in convex problems that are simpler to solve. There are mainly two convexification approaches. First is convex relaxation, which finds an outer approximation of the feasible set by modeling the original problem as a semidefinite or a conic program [169, 170, 171, 172]. If the solution obtained from relaxation is also a feasible solution for the original problem, then it is a globally optimal solution [154, 173]. This implies that the relaxation is tight. However, in practice, the convex relaxation can lead to non-physical solutions when it is not tight (this is common when bus power lower bounds are binding) [174]. In such situations, distinguishing between whether the original problem is genuinely infeasible or if the relaxation method has failed becomes challenging.

Unlike convex relaxation, the second approach, convex restriction, provides an inner approximation to the feasible region. Optimization within these convex subsets guarantees the feasibility of solutions for the original OPF problem. In essence, if the convex restriction algorithm produces a solution, that solution is guaranteed to be physically attainable [175, 176]. In this chapter, we focus on convex restrictions of OPF feasible sets in radial networks. It's noteworthy that most distribution networks are operated radially. Due to the growing need for integrating distributed generation and facilitating demand response [177], solving

OPF in distribution networks has become increasingly important.

Most existing convex restriction approaches work in the power injection space [178, 179]. It turns out working in this space often requires an assumption that the admissible power injection space or voltage space (or both) has a polytopic shape for analytical convenience. Determining a non-conservative polytopic inner approximation of the original feasible set can be nontrivial. It may involve solving nonconvex optimization problems [178]. Attempts to simplify or bypass this computationally intensive step often result in overly conservative results (see e.g., Figure 7.1 as an example). Moreover, this assumption itself may be overly restrictive as it approximates every nonlinear constraint using a linear one.

To address these challenges, we propose to construct the convex restriction in a transformed coordinate space of voltage phase angles. Specifically, we apply a change of variables such that the active and reactive power equations become naturally convex after variable change, hence eliminating the need to approximate the bus power upper bound constraints. The lower bound constraints can be approximated using the first-order Taylor approximation, which is the best (the least conservative) upper bound one could have for concave functions. We use a 3-bus line network as an example (Figure 7.1) to show that the convex restriction constructed in this way can be a maximal convex subset, meaning it cannot be contained within any other convex subset.

One may note that different tangent points can produce different first-order Taylor approximations, thus leading to different convex restricted sets. Therefore, we introduce an iterative algorithm that progressively refines the locations of tangent points to find the optimal convex restriction, one that contains the optimal solution of the original problem. We test our method on the IEEE 123-bus distribution network with three types of objective functions applied: power loss minimization, generation cost minimization, and state estimation. The simulation results show that the iterative algorithm that builds on the proposed convex restriction method always finds a good feasible solution within at most 10 iterations, even when the traditional methods failed.

This chapter is organized as follows. In Section 7.2, we present the original formulation of the AC optimal power flow problem. Section 7.3.3 introduces our proposed convex restriction method and the resulting convex restricted OPF. We examine the geometry of our

constructed convex restricted set in Section 7.4. Based on the insights from Section 7.4, we present the iterative algorithm in Section 7.5.3 to obtain an optimal solution for the original OPF problem. Additionally, in Section 7.5.3, we also compare the proposed method against several baseline methods using the IEEE 123-bus distribution network. Finally, Section VI concludes this chapter.

7.2 Model and Problem Formulation

Consider a radial network, where $\mathcal{N}$ is the set of buses and $\mathcal{E}$ the set of lines. For simplicity, we assume that the voltage magnitudes are at 1 p.u. and consider the following problem:

$$\min_{\boldsymbol{\delta}} c(\mathbf{p}^{inj}, \mathbf{q}^{inj}) \tag{7.1a}$$

$$\text{s.t. } p_i^{inj} = \textstyle\sum_{j:(i,j)\in\mathcal{E}} g_{ij} - g_{ij}\cos(\delta_{ij}) + b_{ij}\sin(\delta_{ij}) \tag{7.1b}$$

$$q_i^{inj} = \textstyle\sum_{j:(i,j)\in\mathcal{E}} b_{ij} - b_{ij}\cos(\delta_{ij}) - g_{ij}\sin(\delta_{ij}) \tag{7.1c}$$

$$\underline{P}_i \leqslant p_i^{inj} \leqslant \overline{P}_i,\ \underline{Q}_i \leqslant q_i^{inj} \leqslant \overline{Q}_i,\ \underline{\delta}_{ij} \leqslant \delta_{ij} \leqslant \overline{\delta}_{ij} \tag{7.1d}$$

where $\boldsymbol{\delta}$ is the voltage angle vector, $\delta_{ij} = \delta_i - \delta_j$ is the angle difference between bus i and j, $g_{ij} - jb_{ij}$ is the admittance of the line (i,j), and p_i^{inj} and q_i^{inj} are active and reactive power at each bus i, respectively. We do not explicitly specify whether a bus is a generator or a load. This information can be inferred from the upper and lower bounds on power, for example, if $\overline{P}_i$ is negative, then bus i is a load bus. We will come back to the objective in later sections, but it suffices to think of c as some cost function in active and reactive power.

The constraint $\underline{\delta}_{ij} \leqslant \delta_{ij} \leqslant \overline{\delta}_{ij}$ is important to our analysis. Particularly, we assume that the limits satisfy $[\underline{\delta}_{ij}, \overline{\delta}_{ij}] \subset (-\frac{\pi}{2}, \frac{\pi}{2})$. Under this assumption, $\sin(\delta_{ij})$ is a monotonic function of δ_{ij}, which forms the basis of our approach. We believe this assumption is likely to be true in practice, since it is difficult to think of a distribution system where the angle differences would be larger than 90 degrees.

We denote the feasible set of (7.1) as δ. Since δ is not convex in general, solving (7.1) is nontrivial and a number of numerical methods have been developed [180, 154, 155, 156, 157]. A drawback of all these methods is that if they do not return a solution–for example,

when a convex relaxation is not tight or when a Newton-type algorithm fails to converge– it's not easy to tell whether the problem itself is infeasible or it's the algorithm that has failed. Unlike these methods, convex restriction finds an inner subset of δ. If this convex subset is not empty, it serves as evidence that the original problem is indeed feasible, and optimization over this subset is guaranteed to produce a feasible solution. In the next section, we introduce a simple change of variables technique that helps to construct a convex restriction of δ.

7.3 Convex Restriction Algorithm

We introduce the following change of variables: Let $z_{ij} = \sin(\delta_{ij})$, for all $(i,j) \in \mathcal{E}$. Note that this transformation preserves the feasibility and optimality of (7.1), namely, the problem in the transformed coordinates is equivalent to (7.1). Since the sine function is monotonically increasing in $(-\frac{\pi}{2}, \frac{\pi}{2})$ and is invertible, the angle difference δ_{ij} and hence the angles themselves $\boldsymbol{\delta}$ can be readily obtained from z_{ij}. Therefore, in the rest of this chapter, we focus on solving the problem in the transformed coordinates. Next, we look at how each of the constraints in (7.1) are represented after this change of variables.

7.3.1 Angle Difference Constraints

The angle difference constraints $\underline{\delta}_{ij} \leqslant \delta_{ij} \leqslant \overline{\delta}_{ij}$ become $\underline{\delta}_{ij} \leqslant \sin^{-1}(z_{ij}) \leqslant \overline{\delta}_{ij}$. This appears to be nonconvex in z_{ij}, but a simple observation is that because sin is monotonically increasing, the constraint is equivalent to $\underline{z}_{ij} \leqslant z_{ij} \leqslant \overline{z}_j$, with $\underline{z}_{ij} = \sin^{-1} \underline{\delta}_{ij}$ and $\overline{z}_{ij} = \sin^{-1} \overline{\delta}_{ij}$, which are linear inequalities (and hence convex).

7.3.2 Active and Reactive Power Constraints

Using the simple fact that for $\delta \in (-\frac{\pi}{2}, \frac{\pi}{2})$, $\cos(\delta) = \sqrt{1 - \sin(\delta)^2}$, the power flow from bus i to bus j is

$$p_{ij}^f(z_{ij}) = g_{ij} - g_{ij}\sqrt{1 - z_{ij}^2} + b_{ij} z_{ij}$$

in the z_{ij} variables. Then letting $\mathbf{z} = \{z_{ij}, (i,j) \in \mathcal{E}\}$, the nodal active power injection at bus i is

$$p_i(\mathbf{z}) = \sum_{j:(i,j)\in\mathcal{E}} g_{ij} - g_{ij}\sqrt{1 - z_{ij}^2} + b_{ij} z_{ij}, \tag{7.2}$$

with the relevant constraint being $\underline{P}_i \leqslant p_i(\mathbf{z}) \leqslant \overline{P}_i$.

The affine terms in (7.2) cause no difficulty. The nonlinear term, $-g_{ij}\sqrt{1 - z_{ij}^2}$, is more interesting. By elementary calculations, $\sqrt{1 - z_{ij}^2}$ is a concave function of z_{ij}. Therefore, $-g_{ij}\sqrt{1 - z_{ij}^2}$ is convex in $z_{i,j}$. Consequently, the upper bound on the active power, $p_i(\mathbf{z}) \leqslant \overline{P}_i$, is a convex constraint. In contrast to other methods where every nonlinear constraint need to be approximated [178, 179] and causes the convex restriction to shrink, the upper bounds on active power are naturally convex in the $\mathbf{z}$ variables. This actually recovers a known result in OPF, where the problem tends to be convex under a condition called load over-satisfaction, meaning that the lower bounds are removed [52].

The lower bound on active power has the form of a convex function greater than a constant, and is nonconvex. We replace it by a supporting hyperplane of the convex function, to create an upper estimate of the lower bound. Specifically, given a differentiable function f, it is convex if and only if the following first order condition is satisfied:

$$f(\mathbf{x}) \geqslant f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x} - \mathbf{y}) \tag{7.3}$$

for all $\mathbf{x}$ and $\mathbf{y}$ in its domain. Applying (7.3) to $p_i^g(\mathbf{z})$, we define:

$$\underline{p}_i(\mathbf{z}) := p_i(\tilde{\mathbf{z}}_i^p) + \nabla p_i(\tilde{\mathbf{z}}_i^p)^T(\mathbf{z} - \tilde{\mathbf{z}}_i^p), \tag{7.4}$$

for some base point $\tilde{\mathbf{z}}_i^p$. With (7.4), the two convex inequalities, $p_i(\mathbf{z}) \leqslant \overline{P}_i$ and $\underline{p}_i(\mathbf{z}) \geqslant \underline{P}_i$, together imply the original nonconvex inequality $\underline{P}_i \leqslant p_i(\mathbf{z}) \leqslant \overline{P}_i$. Particularly, in order to find a supporting hyperplane for each lower bound constraint, one must choose base points where these constraints are active. A valuable insight is that the constraint $p_i(\mathbf{z}) \leqslant \underline{P}_i$ is indeed convex. Therefore, a base point for each lower bound constraint can be found by starting with a strictly feasible point $\mathbf{z}^{\mathrm{o}}$ and projecting $\mathbf{z}^{\mathrm{o}}$ onto the lower bound constraints. Details of this procedure are described in the next section.

The reactive power constrains can be treated in exactly the same way, by noticing that

$$q_i(\mathbf{z}) = \sum_{j:(i,j)\in\mathcal{E}} b_{ij} - b_{ij}\sqrt{1 - z_{ij}^2} - g_{ij} z_{ij}, \tag{7.5}$$

is convex. The lower bounds can be handled by defining

$$\underline{q}_i(\mathbf{z}) := q_i(\tilde{\mathbf{z}}_i^q) + \nabla q_i(\tilde{\mathbf{z}}_i^q)^T(\mathbf{z} - \tilde{\mathbf{z}}_i^q), \tag{7.6}$$

at some base point $\tilde{\mathbf{z}}_i^q$. Then the convex restriction of the nonconvex inequality $\underline{Q}_i \leqslant q_i(\mathbf{z}) \leqslant \overline{Q}_i$ can be represented as two convex inequalities, $q_i(\mathbf{z}) \leqslant \overline{Q}_i$ and $\underline{q}_i(\mathbf{z}) \geqslant \underline{Q}_i$.

7.3.3 OPF with a Convex Feasible Set

All together, the convex restricted version of the problem in (7.1) is

$$\min_{\mathbf{z}} c\big(\mathbf{p}(\mathbf{z}), \mathbf{q}(\mathbf{z})\big) \tag{7.7a}$$

$$\text{s.t. } (7.2),\ (7.5), \tag{7.7b}$$

$$\underline{\mathbf{z}} \leqslant \mathbf{z} \leqslant \overline{\mathbf{z}}, \tag{7.7c}$$

$$p_i(\mathbf{z}) \leqslant \overline{P}_i,\ q_i(\mathbf{z}) \leqslant \overline{Q}_i, \tag{7.7d}$$

$$\underline{p}_i(\mathbf{z}) \geqslant \underline{P}_i,\ \underline{q}_i(\mathbf{z}) \geqslant \underline{Q}_i. \tag{7.7e}$$

The following theorem summarizes the main result of this section:

Theorem 7.1. *The feasible set of* (7.7) *is convex, and a feasible solution of* (7.7) *is a feasible solution of* (7.1).

In Figure 7.1, we use a three bus line network as an example to illustrate the discussions in this section. The top figure shows its feasible set in the δ space (green-colored area) and the bottom figure shows the transformed feasible set in the z space, which is the cyan-colored area overlaid by the yellow-colored area. Particularly, the yellow region in the bottom figure is the convex restriction constructed using the techniques introduced in this section. For comparison, the magenta-colored region, obtained from the method in [179] (it appears nonconvex because of the change in coordinates), is much more restricted than ours.

In fact, in this example, it is not possible to find another convex subset of the feasible region that contains the yellow region. Therefore, the yellow region can be considered as

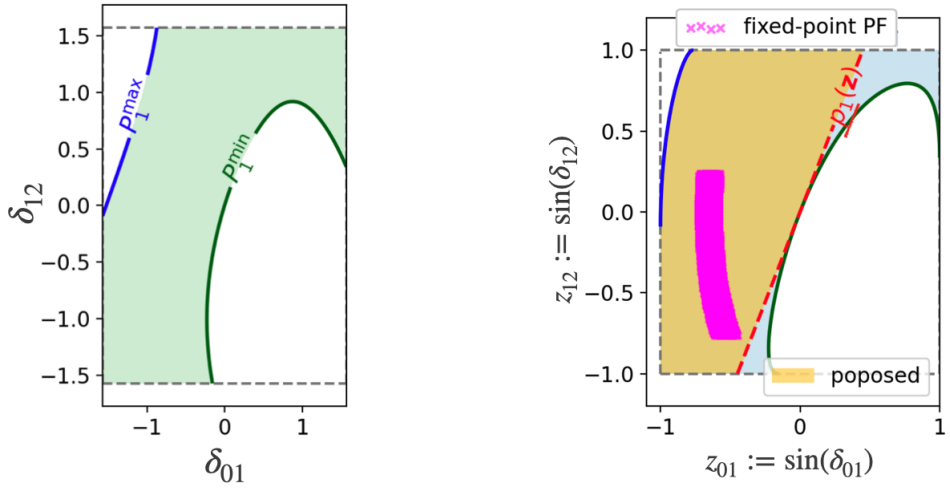


Figure 7.1: A 3-bus line network example. Top is the feasible set in the δ space, and bottom is the feasible set in the z space. Yellow is the convex restriction found by our method and magenta is the set found by the method in [179].

the maximal convex subset. In the next section, we will take a closer look at the geometry of the feasible set for (7.7) and show why it is in some sense an optimal convex restriction.

7.4 Geometry of Convex Restrictions

We start with the following definition:

Definition 7.2 (Maximal convex restriction)**.** *Let f be a function from $\mathbb{R}^n$ to $\mathbb{R}$. Let $\mathcal{A} = \{\mathbf{x} : f(\mathbf{x}) \leqslant 0\}$ be a set in $\mathbb{R}^n$. We say a convex set $\mathcal{B}$ is a maximal convex restriction of $\mathcal{A}$ if 1) $\mathcal{B} \subseteq \mathcal{A}$, and 2) there does not exist another convex set $\mathcal{C}$ such that $\mathcal{B} \subset \mathcal{C} \subset \mathcal{A}$.*

Note that the bus power lower bound constraints all take the form of $\mathcal{A} = \{\mathbf{x} : f(\mathbf{x}) \leqslant 0\}$. Definition 7.2 says that given a set defined in this form, a convex restriction is maximal if it is not included in a larger convex subset of the set $\mathcal{A}$. The convex restriction shown in Fig. 7.1 illustrates this point. The yellow region at the bottom is maximal, since there does not exist another convex subset of the feasible region that contains it. But it is not the only maximal set, since the line that forms $\underline{P}_i$ could be tangent to the curve $p_i(\mathbf{z}) = \underline{P}_i$ at other points. In fact, by changing the tangent lines, we obtain the family of maximal

convex restrictions to the feasible set. We formalize this observation in the next theorem.

Theorem 7.3. *If* (7.1) *is strictly feasible, then each of the constraints defining the feasible set of* (7.7) *forms a convex restriction that is maximal in the sense of Definition 7.2.*

We want to emphasize that the maximal property is defined at a per-constraint level. Because multiple restricted sets can be maximal, the size of the feasible set in (7.7) depends on the choice of the base points. After stating the proof of Theorem 7.3, we outline an iterative procedure to select the base point that jointly optimizes the convex restrictions of all constraints.

The proof of the theorem follows from the fact that the convex restriction of the bus power lower bound constraints is to find the best concave lower bound of a convex function, which is one of its supporting hyperplanes. More precisely, we have the following lemma:

Lemma 7.4. *Let $f(\mathbf{x})$ be a convex function. Let $g(\mathbf{x})$ be a concave function. If $g(\mathbf{x}) \leqslant f(\mathbf{x})$ for all $\mathbf{x}$, then there exist an affine function $h(\mathbf{x})$ such that $g(\mathbf{x}) \leqslant h(\mathbf{x}) \leqslant f(\mathbf{x})$.*

Proof. First consider where the inequality is strict, that is, $g(\mathbf{x}) < f(\mathbf{x})$. A fundamental result in convex geometry is that there exist a separating hyperplane between f and g [77], and it serves as the function h. If the inequality is not strict, then pick a $\mathbf{y}$ such that $g(\mathbf{y}) = f(\mathbf{y})$. Because f is convex and g is concave, there is a supporting hyperplane through the point $\mathbf{y}$ in the form of an affine function $h(\mathbf{x})$ such that $g(\mathbf{x}) \leqslant h(\mathbf{x}) \leqslant f(\mathbf{x})$. ■

Lemma 7.4 states that using affine functions to replace the active and reactive power lower bounds is the best one could do, and there does not exist another type of functions that will convexify the constraints while enlarging the feasibility region. Because the affine functions $\underline{p}_i(\mathbf{z})$ and $\underline{q}_i(\mathbf{z})$ are tangent to a point on the original lower bound, they are then maximal in the sense of Definition 7.2.

Next, we show how these tangent points can be found due to the following lemma, which requires the strict feasibility condition in the statement of Theorem 7.3.

Lemma 7.5. *Let $\mathbf{z}^{\mathrm{o}}$ be a strictly feasible solution for* (7.1) *and define $\underline{\mathcal{P}}_i = \{\mathbf{z} : p_i(\mathbf{z}) \leqslant \underline{P}_i\}$. Let $\tilde{\mathbf{z}}_i^p$ be the Euclidean (2-norm) projection of $\mathbf{z}^{\mathrm{o}}$ onto $\underline{\mathcal{P}}_i$, i.e., $\tilde{\mathbf{z}}_i^p = \arg\min_{\mathbf{z}\in\underline{\mathcal{P}}_i} \|\mathbf{z} - \mathbf{z}^{\mathrm{o}}\|_2^2$. Then the projection $\tilde{\mathbf{z}}_i^p$ lies on the curve $p_i(\mathbf{z}) = \underline{P}_i$ whenever $\underline{\mathcal{P}}_i$ is nonempty.*

By defining $\underline{\mathcal{Q}}_i = \{\mathbf{z} : q_i(\mathbf{z}) \leqslant \underline{Q}_i\}$*, we have the same argument for reactive power equations that the projection onto* $\mathcal{Q}_i$ *lies on the curve* $q_i(\mathbf{z}) = \underline{Q}_i$*.*

Proof. Since $\mathbf{z}^{\mathrm{o}}$ is a strictly feasible point, $\mathbf{z}^{\mathrm{o}} \notin \underline{\mathcal{P}}_i$. Further, the set $\underline{\mathcal{P}}_i$ is convex. Due to these facts, the Euclidean projection of $\mathbf{z}^{\mathrm{o}}$ onto $\underline{\mathcal{P}}_i$ lies on the boundary of $\underline{\mathcal{P}}_i$. Now suppose for the sake of contradiction that the projection is an interior point of $\underline{\mathcal{P}}_i$, i.e., $p_i(\tilde{\mathbf{z}}_i^p) < \underline{P}_i$. Since the function $\hat{p}_i(\alpha) := p_i(\alpha\tilde{\mathbf{z}}_i^p + (1-\alpha)\mathbf{z}^{\mathrm{o}})$ is continuous for $\alpha \in [0,1]$, and $\hat{p}_i(0) > \underline{P}_i$ and $\hat{p}_i(1) < \underline{P}_i$, there exists $\alpha^\star \in (0,1)$ such that $\hat{p}_i(\alpha^\star) = \underline{P}_i$. Let $\alpha^\star\tilde{\mathbf{z}}_i^p + (1-\alpha^\star)\mathbf{z}^{\mathrm{o}} = \mathbf{z}^{\mathrm{b}}$, and we have $\|\mathbf{z}^{\mathrm{b}} - \mathbf{z}^{\mathrm{o}}\|_2^2 = \|\alpha^\star\tilde{\mathbf{z}}_i^p + (1-\alpha^\star)\mathbf{z}^{\mathrm{o}} - \mathbf{z}^{\mathrm{o}}\|_2^2 = \alpha^{\star 2}\|\tilde{\mathbf{z}}_i^p - \mathbf{z}^{\mathrm{o}}\|_2^2 < \|\tilde{\mathbf{z}}_i^p - \mathbf{z}^{\mathrm{o}}\|_2^2$. This contradicts with $\tilde{\mathbf{z}}_i^p$ being the projection. The same proof logic applies to the reactive power equations. ■

Lemma 7.5 suggests that we can obtain a series of tangent points by projecting a strictly feasible point onto each of the lower bound constraints in (7.7d) and (7.7e). Figure 7.2 shows the linearization of the lower bound constraints on active and reactive power for a 3-bus line network using the tangent points found by projection. Note that if $\underline{\mathcal{P}}_i$ is empty, then it does not need to be linearized since the corresponding constraint $p_i(\mathbf{z}) \geqslant \underline{P}_i$ is vacuous and never active. This is easily checked when solving the projection. If the projection problem is infeasible, then a base point is not needed and the constraint can be removed.

It's important to note that by choosing a different initial feasible point $\mathbf{z}^{\mathrm{o}}$, a different set of tangent points can be obtained, hence resulting in a different convex restricted set. Building on this insight, in the next section, we will introduce an iterative algorithm that progressively refines the choice of the feasible point and hence locations of the tangent points. As a result, a sequence of convex restricted sets are generated and each contains solutions progressively approach the optimal ones. We will evaluate the algorithm's performance using the IEEE 123-bus distribution network, considering various types of objective functions.

7.5 Simulation Studies

In this section, we present the algorithm for solving (7.1) by iteratively solving a sequence of (7.7). In each iteration, we use the solution from the previous step to update the convex restricted feasible set for the next problem in the sequence. Since these problems have

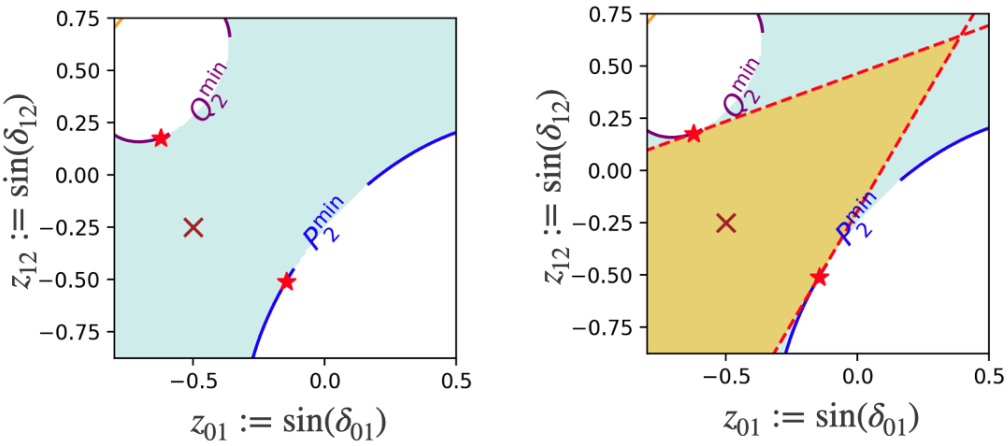


Figure 7.2: The cyan-colored region is the feasible set for a 3-bus line network and the brown cross in it is a strictly feasible point. The red stars are its projections onto the convex sets $\underline{\mathcal{P}}_2 = \{\mathbf{z} : p_2(\mathbf{z}) \leqslant P_2^{\min}\}$ and $\underline{\mathcal{Q}}_2 = \{\mathbf{z} : q_2(\mathbf{z}) \leqslant Q_2^{\min}\}$, respectively. In the bottom figure, the red dashed lines illustrate the linearization around these projected points and the resulting convex restriction is represented by the yellow region.

convex feasible sets, finding a solution is always possible, which ensures the viability of this algorithm. The iterative procedure can be terminated once the obtained solutions stop changing or after a predetermined number of steps.

Note that the convexity of the overall problem in (7.7) (both constraints and objective function) depends on the specific form of the objective function. We will show later that for a commonly used class of objective functions–positive linear combination of the generation costs–(7.7) is convex. For other cost functions, we will show that the iterative algorithm still performs very well.

7.5.1 Iterative Algorithm for Solving (7.1)

Starting from an initial feasible point $\mathbf{z}^{\mathrm{o},0}$, denote the feasible point at the k-th iteration as $\mathbf{z}^{\mathrm{o},k}$. Let $\{\tilde{\mathbf{z}}_i^{p,k}, \tilde{\mathbf{z}}_i^{q,k}\}_{i\in\mathcal{N}}$ be the projected (tangent) points of $\mathbf{z}^{\mathrm{o},k}$ onto each of the lower bound constraints in (7.7d) and (7.7e), and $\mathcal{Z}^k$ the convex restricted set constructed using

Proposed Iterative Algorithm for Solving (7.1)

Inputs: Initial feasible point $\mathbf{z}^{\mathrm{o},0}$, stopping criterion ϵ, maximum number of iterations K.

For iteration k:

Project $\mathbf{z}^{\mathrm{o},k}$ onto $\underline{\mathcal{P}}_i$ and $\underline{\mathcal{Q}}_i$ to get tangents points for linearization:

$$\tilde{\mathbf{z}}_i^{p,k} = \arg\min_{\mathbf{z}\in\underline{\mathcal{P}}_i} \|\mathbf{z} - \mathbf{z}^{\mathrm{o},k}\|_2^2$$
$$\tilde{\mathbf{z}}_i^{q,k} = \arg\min_{\mathbf{z}\in\underline{\mathcal{Q}}_i} \|\mathbf{z} - \mathbf{z}^{\mathrm{o},k}\|_2^2.$$

With the obtained tangent points $\{\tilde{\mathbf{z}}_i^{p,k}, \tilde{\mathbf{z}}_i^{q,k}\}_{i\in\mathcal{N}}$, linearize bus power lower bounds using (7.4) and (7.6).

Formulate the convex restricted problem (7.7).

Solve (7.7) by calling a solver to obtain a solution $\hat{\mathbf{z}}^k$ and the associated objective value $\hat{c}^k$.

Update the feasible point by $\hat{\mathbf{z}}^k \longrightarrow \mathbf{z}^{\mathrm{o},k+1}$.

Let $k+1 \longrightarrow k$.

Repeat the above procedure until $\|\hat{c}^k - \hat{c}^{k-1}\|_2^2 \leqslant \epsilon$ or $k = K$.

Outputs: optimal solution $\hat{\mathbf{z}}^k$ and objective value $\hat{c}^k$.

Table 7.1: The iterative algorithm for solving (7.1) using the proposed convex restriction technique, starting with an initial feasible point $\mathbf{z}^{\mathrm{o}}$.

these tangent points, i.e., $\mathcal{Z}^k = \{\mathbf{z} : (7.7\mathrm{b}) - (7.7\mathrm{d})\}$. To solve (7.7) with $\mathcal{Z}^k$, we can call a convex solver such as CVXPY [88] if the objective function is convex, or an NLP solver like IPOPT [1] if the objective is non-convex. We denote the obtained solution as $\hat{\mathbf{z}}^k$, which is surely a feasible point, and use it to derive a new set of tangent points through projection. With these updated tangent points, we construct another convex feasible set $\mathcal{Z}^{k+1}$ and solve (7.7) on it. This iterative process continues until the sequence of objective values converges or a maximum number of steps is reached. We summarize this iterative algorithm for solving (7.1) in Table 7.1.

7.5.2 Convexity of Objective Functions

In this part, we provide a condition on when the objective functions are convex in the z space, thus making the entire optimization problem in (7.7) convex.

Lemma 7.6. *If the cost function can be written as*

$$c\big(\mathbf{p}(\mathbf{z}), \mathbf{q}(\mathbf{z})\big) = \sum_i f_i(p_i(\mathbf{z})) + g_i(q_i(\mathbf{z})), \tag{7.8}$$

and each f_i and g_i are nondecreasing and convex functions, then the cost $c\big(\mathbf{p}(\mathbf{z}), \mathbf{q}(\mathbf{z})\big)$ is convex in $\mathbf{z}$.

Proof. Note that the active and reactive power injections in the $\mathbf{z}$ space, given by (7.2) and (7.5), are convex functions. Using the fact that composition of a convex function and a convex and nondecreasing function is convex [77], we have that the objective function is convex. ■

Lemma 7.6 applies to many standard OPF objective functions that are of the form $c\big(\mathbf{p}(\mathbf{z}), \mathbf{q}(\mathbf{z})\big) = \sum_{i=1}^{N} c_i p_i(\mathbf{z})$, $c_i \geqslant 0$, such as minimizing total power loss and total generation cost. It's interesting to note that the conditions on the cost in Lemma 7.6 are the same as the ones found in SDP [174] or SOCP relaxations [156].

7.5.3 Numerical Results

In this part, we use the iterative algorithm from Table 7.1 to solve (7.1) for the IEEE 123-bus distribution network [181], testing with different types of objective functions: loss minimization, generation cost minimization and state estimation. We compare our method against several baseline approaches. The first is running the *runopf* module in MATPOWER [2], which is based on Newton-Raphson type algorithms. The second is the second-order cone program (SOCP) relaxation [169, 172]. The third method creates a convex region in the power injection space by restricting the permissible voltage phase angles within a polytope and approximating each nonlinear constraint [179].

Loss Minimization. The first problem we consider is to minimize the total active power loss in meeting the load. The objective function in this case is

$$c\big(\mathbf{p}(\mathbf{z}), \mathbf{q}(\mathbf{z})\big) = \sum_i p_i(\mathbf{z}),$$

which satisfies the condition in Lemma 7.6. As a benchmark test, we set the lower bounds in (7.1) such that they satisfy the conditions (see [156] for details) where the SOCP relaxation is exact. Therefore, the SOCP solution is optimal and we are interested in if we can achieve the same loss through convex restriction.

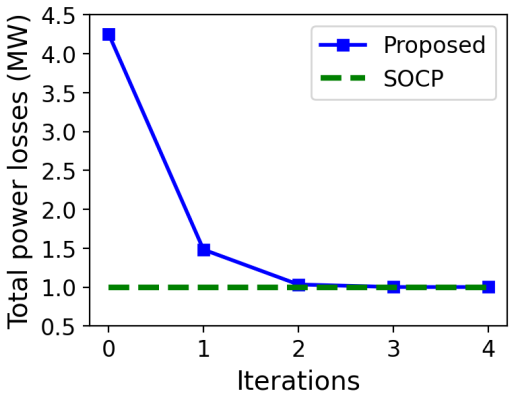


Figure 7.3: Comparison of the proposed algorithm against the baseline method on the 123-bus network for active power loss minimization. The Y-axis is normalized to the minimum objective value within the sequence generated by our algorithm. The lower bounds in (7.1) are set such that the SOCP relaxation is exact. Our algorithm starts at a higher loss (with a random starting point), but quickly achieves the same loss as SOCP relaxation after 2 iterations.

Figure 7.3 shows the performance of our algorithm compared to the solution of SOCP. We start with a random intialization point that has a high loss, but we quickly reach the same loss as SOCP relaxation in two iterations. This shows that our convex restricted feasible set (after an iteration) contains the optimal solution of the original non-convex problem. It's worth noting that both the SOCP and our algorithm solve convex problems of the same size, and they have roughly the same computational speed.

Generation Cost Minimization. Here, we consider a cost in the form of

$$c\big(\mathbf{p}(\mathbf{z}), \mathbf{q}(\mathbf{z})\big) = \sum_i c_i p_i(\mathbf{z}),$$

where c_i are positive constants. We set the lower bounds such that the conditions for SOCP relaxation to be exact are not met and test the performance of our algorithm in these scenarios. Indeed, when the SOCP relaxation is solved, it does not give a physical solution,[1] and thus it does not return a feasible solution to the original problem. In addition, the *runopf* module in MATPOWER also fails to find a solution. As for the method that assumes a polytopic phase angle feasible set, it proves to be overly restrictive and results in an empty set unless we carefully adjust the hyperparameters used for approximating each nonlinear constraint.

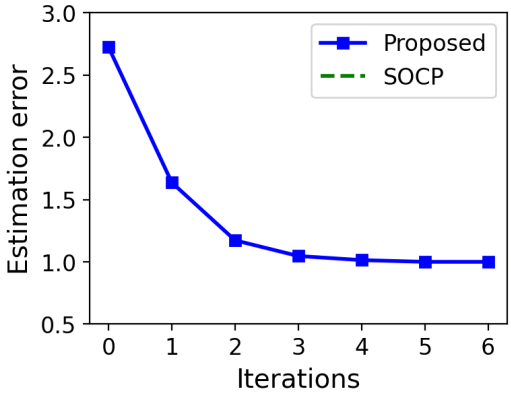


Figure 7.4: Performance of the proposed algorithm on the 123-bus network for generation cost minimization. The Y-axis is normalized to the minimum objective value within the sequence generated by our algorithm. SOCP turns out to be inexact in this case and does not produce a feasible solution. The *runopf* routine in MATPOWER also fails to converge. In contrast, our algorithm reliably decreases the cost and converges to a physically feasible solution.

In Figure 7.4, we show the performance of our algorithm, where our method reliably decreases the cost and converges to a good feasible solution. This shows the benefit of

[1] Essentially, the SOCP relaxation relaxes an equality constraint of the type $R_{ij}^2 + I_{ij}^2 = 1$ into $R_{ij} + I_{ij} \leqslant 1$, where R_{ij} and I_{ij} are variables associated with line ij. When the relaxation is exact, the optimal solution $(R_{ij}^\star)^2$ and $(I_{ij}^\star)^2$ will satisfy the equality constraint. However, if $(R_{ij}^\star)^2 + (I_{ij}^\star)^2 < 1$, the relaxation is not exact and we cannot recover a solution to the original problem (7.1).

working with a convex restriction, since we are always guaranteed to find a feasible solution of the original problem.

State Estimation Given some measurements of nodal active and reactive power injections, the goal of state estimation is to find the optimal $\mathbf{z}$ that best matches these measurements. For example, suppose we take active and reactive power measurements at each bus, denoted by $\{\hat{p}_1, \ldots, \hat{p}_N, \hat{q}_1, \ldots, \hat{q}_N\}$, state estimation solves (7.7) with the following objective:

$$c\big(\mathbf{p}(\mathbf{z}), \mathbf{q}(\mathbf{z})\big) = \sum_{i=1}^{N} (\hat{p}_i - p(\mathbf{z})_i)^2 + \sum_{i=1}^{N} (\hat{q}_i - q(\mathbf{z})_i)^2. \tag{7.9}$$

Due to the non-monotonicity in the objective function (7.9), (7.7) is non-convex, but it can still be solved using a nonlinear solver such as IPOPT. Note that since the feasible set of the convex restricted problems are convex, we are still guaranteed that the solution is feasible (although perhaps not optimal).

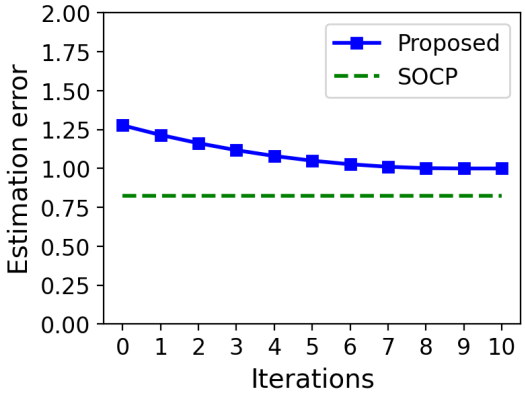


Figure 7.5: Performance of the proposed algorithm on the 123-bus network for state estimation. The Y-axis is normalized to the minimum objective value within the sequence generated by our algorithm. SOCP turns out to be inexact and the cost returned by it is not actually achievable. But it serves as a lower bound, and our algorithm is not very far from this bound, and it produces a sequence of feasible solutions.

In the case of state estimation, the SOCP relaxation is quite far away from being exact. This is expected, since the objective function is not increasing in the active power, and convex relaxation algorithms tend to struggle to produce physically meaningful solutions.

In contrast, our proposed algorithm continues to perform effectively. To illustrate this, we present the convergence of the objective value sequence generated by our algorithm in Figure 7.5. Since the SOCP is inexact, the dashed line in Figure 7.5 is not actually achievable. But it serves as a lower bound, and our algorithm converges to a physically feasible solution not very far from this bound.

7.6 Conclusions

In this chapter, we focused on developing a convex restriction approach for solving AC power flow problems in radial networks. We introduced a simple change of variables technique, showing that the active and reactive power equations are naturally convex in the transformed coordinate space. A detailed procedure was provided to construct a convex subset in this transformed space, and the convex restriction constructed in this way was shown to be a maximal one. Furthermore, we proposed an iterative algorithm to improve on the solution quality by constructing a series of convex restricted sets, each containing solutions progressively closer to the optimal ones. We conducted numerical experiments on the IEEE 123-bus distribution network and solved the ACOPF problem by applying different types of objective functions. The numerical results showed that our method produced good feasible solutions within just a few iterations for all study cases, even when traditional methods failed to return a solution.

Chapter 8

CONCLUSIONS

In this dissertation, we develop machine learning algorithms for faster and more optimal power system operations in uncertain environments by incorporating problem-specific structural insights into algorithm design. More specifically, we develop a flow-based generative approach to model residential load behaviors and generate varied and plentiful future scenarios. Then, to optimize and plan power systems across these diverse scenarios, we design neural network-based solvers to offer solutions orders of magnitude faster than conventional methods. By integrating problem-specific structural insights, we ensure learned solutions satisfy engineering and operational constraints within power systems. These insights also enhance the data efficiency and generalization capabilities of the learning algorithms. In addition, we apply the proposed algorithmic designs to the fundamental resource allocation problem in power system operations, i.e., optimal power flow, to showcase the effectiveness of these methods.

In Chapter 2, we propose a novel scenario forecasting approach for residential load using flow-based conditional generative models. Compared to existing scenario forecasting methods, our approach can generate scenarios that are not only able to infer possible future realizations of residential load from the observed historical data but also realistic enough to cover a wide range of behaviors. In Chapter 3, we propose a novel algorithm for solving DCOPF that guarantees the generalization performance. First, by utilizing the convexity of DCOPF problem, we train an input convex neural network. Second, we construct the training loss based on KKT optimality conditions. By combining these two techniques, the trained model has provable generalization properties, where small training error implies small testing errors. In experiments, our algorithm significantly outperforms other machine learning methods. In Chapter 4, we propose a learning method to solve the two-stage problem in a more efficient and optimal way. A technique called the gauge map is incorporated

into the learning architecture design to guarantee the learned solutions' feasibility to the network constraints. Simulation results on standard IEEE systems show that, compared to iterative solvers and the widely used affine policy, our proposed method not only learns solutions of good quality but also accelerates the computation by orders of magnitude. In Chapter 5, we propose a simple iterative approach to improve the quality of solutions to ACOPF problems. First, we call an existing solver for the ACOPF problem. From the solution and the associated dual variables, we form a partial Lagrangian. Then we optimize this partial Lagrangian and use its solution as a warm start to call the solver again for the ACOPF problem. By repeating this process, we can iteratively improve the solution quality, moving from local solutions to global ones. We show the effectiveness of our algorithm on standard IEEE networks. The simulation results show that our algorithm can escape from local solutions to achieve global optimums within a few iterations. In Chapter 6, we propose a Lagrangian-based learning approach to overcome the challenge that the training data for ACOPF may contain suboptimal solutions. First, we use a neural network to learn the dual variables of the ACOPF problem. Then, from the predicted dual variables, we use a second neural network to predict solutions of the partial Lagrangian and use the predicted solutions as warm starts for the ACOPF problem. We test our approach on standard and modified IEEE networks and show that our approach can reach more globally optimal solutions with significant computational speedup even when the training data consists of mostly suboptimal solutions. In Chapter 7, we propose an analytical method to construct the convex restriction of the feasible set for AC power flows in radial networks. The construction relies on simple geometrical ideas and is explicit, in the sense that it does not involve solving other complicated optimization problems. We also show that the construct restrictions are in some sense maximal, that is, the best possible ones. The numerical experiments on the IEEE 123-bus distribution network show that our method finds good feasible solutions within just a few iterations and works well with various objective functions, even in situations where traditional methods fail to return a solution.

Possible future directions include:

- Extending the proposed learning algorithm for two-stage DCOPF problems to accom-

modate multiple stages. In practical applications, transmission networks are often managed over long time horizons, making them suitable for being modeled as multi-stage optimization problems. However, multistage optimization presents significantly more challenges than two-stage optimization. Incorporating neural networks into this framework requires ensuring that their outputs satisfy the constraints at each stage of the optimization process. To the best of the authors' knowledge, there is currently a lack of research on learning-based algorithms that solve multistage OPF problems while guaranteeing feasibility. Exploring this area could be very promising for facilitating dynamic decision-making in power systems with uncertainty.

- Integrating the gauge map technique and the convex restriction method to develop a learning-based algorithm for solving ACOPF with guaranteed feasibility. In Chapter 4, the gauge map technique is used to transform the neural network's output into an interior point of the feasible set of DCOPF. By designing an algorithm that maps the neural network's output into the convex restriction of the feasible set of ACOPF, we can ensure that the learned solutions are feasible. This is particularly beneficial because finding feasible solutions for ACOPF is generally challenging due to the problem's non-linear and non-convex nature. By combining learning and the convex restriction technique, we can significantly accelerate the solving process, achieving orders of magnitude improvements in speed while always guaranteeing feasible, high-quality solutions.

- Developing machine learning-based optimization methods for applications beyond power system operations, such as transportation and smart building control. A significant issue in modern transportation is electrification and its impact on power system operations. For example, the limited capacity of distribution systems makes simultaneous charging of electric vans in urban delivery fleets impractical. Power system operators must therefore distribute charging activities across different times and locations. This distribution can cause inconvenience for drivers, who may need to alter their routes or charge frequently to complete their deliveries on time. It will be in-

teresting to design efficient, learning-based algorithms for jointly optimizing charging schedules and delivery routes to maintain power system stability, ensure timely deliveries, and enhance driver satisfaction.

Bibliography

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

[2] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[3] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Foundations and Trends in Electric Energy Systems*, vol. 4, no. 1-2, pp. 1–221, 2019.

[4] W. A. Bukhsh, A. Grothey, K. I. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4780–4788, 2013.

[5] L. Zhang and B. Zhang, "An iterative approach to improving solution quality for ac optimal power flow problems," in *Proceedings of the ACM E-Energy*, 2022, p. 289–301.

[6] Y. Guo, M. Pan, and Y. Fang, "Optimal power management of residential customers in the smart grid," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, pp. 1593–1606, 2012.

[7] Y. Ji, E. D. Buechler, and R. Rajagopal, "Data-driven load modeling and forecasting of residential appliances," 2018.

[8] A. Taşcıkaraoğlu, A. Boynuegri, and M. Uzunoglu, "A demand side management strategy based on forecasting of residential renewable sources: A smart home system in turkey," *Energy and Buildings*, vol. 80, pp. 309–320, 2014.

[9] K. Clement-Nyns, E. Haesen, and J. Driesen, "The impact of charging plug-in hybrid electric vehicles on a residential distribution grid," *IEEE Transactions on power systems*, vol. 25, no. 1, pp. 371–380, 2009.

[10] M. Reis, A. Garcia, and R. J. Bessa, "A scalable load forecasting system for low voltage grids," in *2017 IEEE Manchester PowerTech*. IEEE, 2017, pp. 1–6.

[11] Z. Yu, L. Jia, M. C. Murphy-Hoye, A. Pratt, and L. Tong, "Modeling and stochastic control for home energy management," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2244–2255, 2013.

[12] R. Sevlian and R. Rajagopal, "A scaling law for short term load forecasting on varying levels of aggregation," *International Journal of Electrical Power & Energy Systems*, vol. 98, pp. 350–361, 2018.

[13] A. Veit, C. Goebel, R. Tidke, C. Doblander, and H.-A. Jacobsen, "Household electricity demand forecasting–benchmarking state-of-the-art methods," *arXiv preprint arXiv:1404.0200*, 2014.

[14] S. Humeau, T. K. Wijaya, M. Vasirani, and K. Aberer, "Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households," in *2013 Sustainable Internet and ICT for Sustainability (SustainIT)*, Oct 2013, pp. 1–6.

[15] K. Gajowniczek and T. Zabkowski, "Electricity forecasting on the individual household level enhanced based on activity patterns," *PLoS ONE*, vol. 12, 2017.

[16] X. Monica Zhang, K. Grolinger, M. Capretz, and L. Seewald, "Forecasting residential energy consumption: Single household perspective," in *IEEE International Conference on Machine Learning and applications*, 2018, pp. 110–117.

[17] B. Yildiz, J. Bilbao, J. Dore, and A. Sproul, "Short-term forecasting of individual household electricity loads with investigating impact of data resolution and forecast horizon," *Renewable Energy and Environmental Sustainability*, vol. 3, p. 3, 2018.

[18] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914—-938, 2016.

[19] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014.

[20] H. S. Hippert, C. Pedreira, and R. Castro Souza, "Neural networks for short-term load forecasting: A review and evaluation," *Power Systems, IEEE Transactions on*, vol. 16, pp. 44–55, 2001.

[21] B.-J. Chen, M.-W. Chang, and C.-J. Lin, "Load forecasting using support vector machines: A study on eunite competition 2001," *Power Systems, IEEE Transactions on*, vol. 19, pp. 1821–1830, 2004.

[22] R. Weron, *Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach*, 01 1998.

[23] N. Pflugradt and U. Muntwyler, "Synthesizing residential load profiles using behavior simulation," *Energy Procedia*, vol. 122, pp. 655–660, 2017.

[24] N. Narayan, Z. Qin, J. Popovic-Gerber, J.-C. Diehl, P. Bauer, and M. Zeman, "Stochastic load profile construction for the multi-tier framework for household electricity access using off-grid dc appliances," *Energy Efficiency*, 2018.

[25] J. Linssen, P. Stenzel, and J. Fleer, "Techno-economic analysis of photovoltaic battery systems and the influence of different consumer load profiles," *Applied Energy*, vol. 185, pp. 2019–2025, 2017, clean, Efficient and Affordable Energy for a Sustainable Future.

[26] J. Xie, T. Hong, T. Laing, and C. Kang, "On normality assumption in residual simulation for probabilistic load forecasting," *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1046–1053, 2017.

[27] P. E. McSharry, S. Bouwman, and G. Bloemhof, "Probabilistic forecasts of the magnitude and timing of peak electricity demand," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1166–1172, 2005.

[28] S. Fan and R. J. Hyndman, "Short-term load forecasting based on a semi-parametric additive model," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 134–141, 2012.

[29] V. Dordonnat, A. Pichavant, and A. Pierrot, "Gefcom2014 probabilistic electric load forecasting using time series and semi-parametric regression models," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1005–1011, 2016.

[30] J. Xie and T. Hong, "Temperature scenario generation for probabilistic load forecasting," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1680–1687, 2018.

[31] Y. Chen, Y. Wang, D. S. Kirschen, and B. Zhang, "Model-free renewable scenario generation using generative adversarial networks," *IEEE Transactions on Power Systems*, vol. 33, pp. 3265–3275, 2018.

[32] Y. Chen, X. Wang, and B. Zhang, "An unsupervised deep learning approach for scenario forecasts," *2018 Power Systems Computation Conference (PSCC)*, pp. 1–7, 2018.

[33] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[34] Y. Gu, Q. Chen, K. Liu, L. Xie, and C. Kang, "Gan-based model for residential load generation considering typical consumption patterns," in *Conference on Innovative Smart Grid Technologies*, 11 2018.

[35] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International Conference on Machine Learning*, 2017.

[36] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[37] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[38] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *CoRR*, vol. abs/1410.8516, 2014.

[39] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *CoRR*, vol. abs/1605.08803, 2017.

[40] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 236—-10 245.

[41] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[42] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[43] R. Durrett, *Forecasting: Principles and Practice*. Cambridge university press, 2019, vol. 49.

[44] L. Rüschendorf, "The wasserstein distance and approximation theorems," *Probability Theory and Related Fields*, vol. 70, no. 1, pp. 117–129, 1985.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Conference on Neural Information Processing Systems*, 2017.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[47] P. S. Inc. [Online]. Available: `https://dataport.pecanstreet.org`

[48] A. H. Murphy, "What is a good forecast? an essay on the nature of goodness in weather forecasting," *Weather and Forecasting*, vol. 8, no. 2, pp. 281–293, 1993.

[49] P. Pinson, J. Juban, and G. N. Kariniotakis, "On the quality and value of probabilistic forecasts of wind generation," in *2006 International Conference on Probabilistic Methods Applied to Power Systems*, 2006, pp. 1–7.

[50] C. Villani, *Optimal Transport: Old and New*, ser. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008. [Online]. Available: `https://books.google.com/books?id=hV8o5R7_5tkC`

[51] H. W. Dommel and W. F. Tinney, "Optimal power flow solutions," *IEEE Transactions on power apparatus and systems*, no. 10, pp. 1866–1876, 1968.

[52] R. Baldick, *Applied optimization: formulation and algorithms for engineering systems*. Cambridge University Press, 2006.

[53] J. D. Glover, T. J. Overbye, and M. S. Sarma, *Power System Analysis and Design*. CENGAGE Learning, 2017.

[54] B. Stott, J. Jardim, and O. Alsaç, "Dc power flow revisited," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1290–1300, 2009.

[55] F. Milano, "An open source power system analysis toolbox," *IEEE Transactions on Power systems*, vol. 20, no. 3, pp. 1199–1206, 2005.

[56] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2963–2973, 2017.

[57] E. Mohagheghi, M. Alramlawi, A. Gabash, and P. Li, "A survey of real-time optimal power flow," *Energies*, vol. 11, no. 11, p. 3142, 2018.

[58] A. Hauswirth, S. Bolognani, G. Hug, and F. Dörfler, "Projected gradient descent on riemannian manifolds with applications to online power system optimization," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 225–232.

[59] Y. Zhang, E. Dall'Anese, and M. Hong, "Dynamic admm for real-time optimal power flow," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 1085–1089.

[60] S. Huang and V. Dinavahi, "Fast batched solution for real-time optimal power flow with penetration of renewable energy," *IEEE Access*, vol. 6, pp. 13 898–13 910, 2018.

[61] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2019, pp. 1–6.

[62] ——, "Deepopf: A deep neural network approach for security-constrained dc optimal power flow," *arXiv preprint arXiv:1910.14448*, 2019.

[63] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast ac optimal power flow," *arXiv preprint arXiv:1910.01213*, 2019.

[64] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for minimising worst-case violations in dc optimal power flow," *arXiv preprint arXiv:2107.00465*, 2021.

[65] D. Deka and S. Misra, "Learning for dc-opf: Classifying active sets using neural nets," in *2019 IEEE Milan PowerTech*. IEEE, 2019, pp. 1–6.

[66] L. A. Roald and D. K. Molzahn, "Implied constraint satisfaction in power system optimization: The impacts of load variations," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 308–315.

[67] Y. Chen and B. Zhang, "Learning to solve network flow problems via neural decoding," *arXiv:2002.04091*, 2020.

[68] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine learning*, vol. 15, no. 2, pp. 201–221, 1994.

[69] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, 2016, pp. 372–387.

[70] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," 2018.

[71] D. E. Womble, "Machine learning: Issues and opportunities," Oak Ridge National Lab, Tech. Rep. 112595, 2018.

[72] J. L. Cremer, I. Konstantelos, and G. Strbac, "From optimization-based machine learning to interpretable security rules for operation," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 3826–3836, 2019.

[73] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *International Conference on Machine Learning*, 2017.

[74] Y. Chen, Y. Shi, and B. Zhang, "Optimal control via neural networks: A convex approach," in *International Conference on Learning Representations, ICLR*, 2019.

[75] ——, "Data-driven optimal voltage regulation using input convex neural networks," *Electric Power Systems Research*, vol. 189, 2020.

[76] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv:1609.04836*, 2016.

[77] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[78] Y. Ji, "Operation under uncertainty in electric grid: A multiparametric programming approach," Ph.D. dissertation, Cornell University, 2017.

[79] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Learning optimal power flow: Worst-case guarantees for neural networks," *arXiv preprint arXiv:2006.11029*, 2020.

[80] M. K. Singh, S. Gupta, V. Kekatos, G. Cavraro, and A. Bernstein, "Learning to optimize power distribution grids using sensitivity-informed deep neural networks," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2020, pp. 1–6.

[81] B. Zhang, R. Rajagopal, and D. Tse, "Network risk limiting dispatch: Optimal control and price of uncertainty," *IEEE Transactions on Automatic Control*, vol. 59, no. 9, pp. 2442–2456, 2014.

[82] R. Diestel, "Graph theory, volume 173 of," *Graduate texts in mathematics*, p. 7, 2012.

[83] D. S. Kirschen and G. Strbac, *Fundamentals of power system economics*. John Wiley & Sons, 2018.

[84] D. Bertsimas, J. N. Tsitsiklis, and J. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific, 1997, vol. 6.

[85] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Physical review letters*, vol. 59, no. 19, p. 2229, 1987.

[86] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning polynomials with neural networks," in *International conference on machine learning*, 2014, pp. 1908–1916.

[87] Z. Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 6158–6169.

[88] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[89] M. Andersen, J. Dahl, and L. Vandenberghe. Cvxopt: A python package for convex optimization, version 1.1.6. [Online]. Available: `https://cvxopt.org/`

[90] D. Bienstock, M. Chertkov, and S. Harnett, "Chance-constrained optimal power flow: Risk-aware network control under uncertainty," *SIAM Review*, vol. 56, no. 3, pp. 461–495, 2014. [Online]. Available: `https://doi.org/10.1137/130910312`

[91] P. P. Varaiya, F. F. Wu, and J. W. Bialek, "Smart operation of smart grid: Risk-limiting dispatch," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 40–57, 2011.

[92] R. Rajagopal, S. Univ, E. Bitar, P. Varaiya, F. Wu, and U. Berkeley, "Risk-limiting dispatch for integrating renewable power," *International Journal of Electrical Power and Energy Systems*, vol. 44, 10 2011.

[93] B. Zhang, R. Rajagopal, and D. Tse, "Network risk limiting dispatch: Optimal control and price of uncertainty," *IEEE Transactions on Automatic Control*, vol. 59, no. 9, pp. 2442–2456, 2014.

[94] E. Sjödin, D. F. Gayme, and U. Topcu, "Risk-mitigated optimal power flow for wind powered grids," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 4431–4437.

[95] L. Roald, S. Misra, T. Krause, and G. Andersson, "Corrective control to handle forecast uncertainty: A chance constrained optimal power flow," *IEEE Transactions on Power Systems*, vol. 32, no. 2, pp. 1626–1637, 2016.

[96] J. Birge, "State-of-the-art-survey—stochastic programming: Computation and applications," *INFORMS Journal on Computing*, vol. 9, pp. 111–133, 05 1997.

[97] A. Shapiro, *Stochastic programming by Monte Carlo simulation methods.* Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät, 2000.

[98] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on Stochastic Programming.* Society for Industrial and Applied Mathematics, 2009. [Online]. Available: `https://epubs.siam.org/doi/abs/10.1137/1.9780898718751`

[99] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, 2nd ed. Springer Publishing Company, Incorporated, 2011.

[100] J. Linderoth, A. Shapiro, and S. Wright, "The empirical behavior of sampling methods for stochastic programming," *Annals of Operations Research*, vol. 142, pp. 215 –, 02 2006.

[101] T. H. de Mello and G. Bayraksan, "Monte carlo sampling-based methods for stochastic optimization," *Surveys in Operations Research and Management Science*, vol. 19, no. 1, pp. 56–85, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876735414000038

[102] S. Kim, R. Pasupathy, and S. G. Henderson, "A guide to sample average approximation," *Handbook of simulation optimization*, pp. 207–243, 2015.

[103] Y. Chen, Y. Wang, D. Kirschen, and B. Zhang, "Model-free renewable scenario generation using generative adversarial networks," *IEEE Transactions on Power Systems*, vol. 33, no. 3, pp. 3265–3275, 2018.

[104] F. Fioretto, "Integrating machine learning and optimization to boost decision making," in *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2022, pp. 5808–5812. [Online]. Available: https://doi.org/10.24963/ijcai.2022/815

[105] F. Capitanescu, J. M. Ramos, P. Panciatici, D. Kirschen, A. M. Marcolini, L. Platbrood, and L. Wehenkel, "State-of-the-art, challenges, and future trends in security constrained optimal power flow," *Electric power systems research*, vol. 81, no. 8, pp. 1731–1741, 2011.

[106] L. Zhang, Y. Chen, and B. Zhang, "A convex neural network solver for dcopf with generalization guarantees," *IEEE Transactions on Control of Network Systems*, vol. 9, pp. 719–730, 2020.

[107] D. Kuhn, W. Wiesemann, and A. Georghiou, "Primal and dual linear decision rules in stochastic and robust optimization," *Mathematical Programming*, vol. 130, pp. 177–209, 2011.

[108] M. Vrakopoulou, K. Margellos, J. Lygeros, and G. Andersson, "Probabilistic guarantees for the n-1 security of systems with wind power generation," *Reliability and risk evaluation of wind integrated power systems*, pp. 59–73, 2013.

[109] L. A. Roald, D. Pozo, A. Papavasiliou, D. K. Molzahn, J. Kazempour, and A. Conejo, "Power systems optimization under uncertainty: A review of methods and applications," *Electric Power Systems Research*, vol. 214, p. 108725, 2023. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0378779622007842`

[110] Y. Zhang, S. Shen, and J. L. Mathieu, "Distributionally robust chance-constrained optimal power flow with uncertain renewables and uncertain reserves provided by loads," *IEEE Transactions on Power Systems*, vol. 32, no. 2, pp. 1378–1388, 2017.

[111] M. Vrakopoulou and I. A. Hiskens, "Optimal control policies for reserve deployment with probabilistic performance guarantees," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 4470–4475.

[112] R. Kannan, J. R. Luedtke, and L. A. Roald, "Stochastic dc optimal power flow with reserve saturation," 2019. [Online]. Available: `https://arxiv.org/abs/1910.04667`

[113] L. Duchesne, E. Karangelos, and L. Wehenkel, "Recent developments in machine learning for energy systems reliability management," *Proceedings of the IEEE*, 2020.

[114] M. K. Singh, S. Gupta, V. Kekatos, G. Cavraro, and A. Bernstein, "Learning to optimize power distribution grids using sensitivity-informed deep neural networks," in *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids*, 2020.

[115] P. Donti, A. Agarwal, N. V. Bedmutha, L. Pileggi, and J. Z. Kolter, "Adversarially robust learning for security-constrained optimal power flow," in *Conference on Neural Information Processing Systems, virtual conference*, 2021.

[116] D. Tabas and B. Zhang, "Computationally efficient safe reinforcement learning for power systems," *2022 American Control Conference (ACC)*, pp. 3303–3310, 2021.

[117] ——, "Safe and efficient model predictive control using neural networks: An interior point approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 1142–1147.

[118] M. Li, S. Kolouri, and J. Mohammadi, "Learning to solve optimization problems with hard linear constraints," *IEEE Access*, pp. 1–1, 2023. [Online]. Available: `https://doi.org/10.1109%2Faccess.2023.3285199`

[119] A. Khoshrou and E. J. Pauwels, "Short-term scenario-based probabilistic load forecasting: A data-driven approach," *Applied Energy*, 2019.

[120] L. Zhang and B. Zhang, "Scenario forecasting of residential load profiles," 2019. [Online]. Available: `https://arxiv.org/abs/1906.07373`

[121] M. Garcia and R. Baldick, "Approximating economic dispatch by linearizing transmission losses," *IEEE Transactions on Power Systems*, vol. 35, no. 2, pp. 1009–1022, 2019.

[122] R. Palma-Benhke, A. Philpott, A. Jofré, and M. Cortés-Carmona, "Modelling network constrained economic dispatch problems," *Optimization and Engineering*, vol. 14, pp. 417–430, 2013.

[123] J. Higle, "Stochastic programming: Optimization when uncertainty matters," *TutORials in operations research*, 09 2005.

[124] A. Shapiro and A. Ruszczyński, Eds., *Stochastic Programming*, ser. Handbooks in Operations Research and Management Science. Elsevier, 2003, vol. 10.

[125] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[126] E. Oki, "Glpk (gnu linear programming kit)," in *Department for Applied Informatics, Moscow Aviation Institute*, 2012.

[127] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the theory of policy gradient methods: Optimality, approximation, and distribution shift," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 4431–4506, 2021.

[128] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Birkhäuser Basel, 2007.

[129] W. Lee, H. Yu, X. Rival, and H. Yang, "On correctness of automatic differentiation for non-differentiable functions," in *Advances in Neural Information Processing Systems*, 2020.

[130] W. A. Bukhsh, A. Grothey, K. I. M. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4780–4788, 2013.

[131] S. Babaeinejadsarookolaee, A. Birchfield, R. D. Christie, C. Coffrin, C. DeMarco, R. Diao, M. Ferris, S. Fliscounakis, S. Greene, R. Huang, C. Josz, R. Korab, B. Lesieutre, J. Maeght, T. W. K. Mak, D. K. Molzahn, T. J. Overbye, P. Panciatici, B. Park, J. Snodgrass, A. Tbaileh, P. V. Hentenryck, and R. Zimmerman, "The power grid library for benchmarking ac optimal power flow algorithms," 2021.

[132] Welcome to colaboratory. [Online]. Available: `https://colab.research.google.com/notebooks/intro.ipynb`

[133] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: `https://arxiv.org/abs/1412.6980`

[134] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, "Adjustable robust solutions of uncertain linear programs1," *Mathematical Programming*, vol. 99, pp. 351–376, 01 2004.

[135] M. B. Cain, R. P. O'neill, A. Castillo *et al.*, "History of optimal power flow and formulations," *Federal Energy Regulatory Commission*, vol. 1, pp. 1–36, 2012.

[136] I. A. Hiskens and R. J. Davy, "Exploring the power flow solution space boundary," *IEEE transactions on power systems*, vol. 16, no. 3, pp. 389–395, 2001.

[137] Z. Qiu, G. Deconinck, and R. Belmans, "A literature survey of optimal power flow problems in the electricity market context," in *IEEE/PES Power Systems Conference and Exposition*, 2009, pp. 1–6.

[138] F. Capitanescu, "Critical review of recent advances and further developments needed in ac optimal power flow," *Electric Power Systems Research*, vol. 136, pp. 57–68, 2016.

[139] W. Ma and J. S. Thorp, "An efficient algorithm to locate all the load flow solutions," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1077–1083, 1993.

[140] J. A. Momoh, R. Adapa, and M. El-Hawary, "A review of selected optimal power flow literature to 1993. i. nonlinear and quadratic programming approaches," *IEEE transactions on power systems*, vol. 14, no. 1, pp. 96–104, 1999.

[141] B. Lesieutre and D. Wu, "An efficient method to locate all the load flow solutions-revisited," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015, pp. 381–388.

[142] J. Momoh, R. Koessler, M. Bond, B. Stott, D. Sun, A. Papalexopoulos, and P. Ristanovic, "Challenges to optimal power flow," *IEEE Transactions on Power systems*, vol. 12, no. 1, pp. 444–455, 1997.

[143] H. Wei, H. Sasaki, J. Kubokawa, and R. Yokoyama, "An interior point nonlinear programming for optimal power flow problems with a novel data structure," *IEEE Transactions on Power Systems*, vol. 13, no. 3, pp. 870–877, 1998.

[144] D. Wu, D. K. Molzahn, B. C. Lesieutre, and K. Dvijotham, "A deterministic method to identify multiple local extrema for the ac optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 654–668, 2017.

[145] N. Costilla-Enriquez, Y. Weng, and B. Zhang, "Combining newton-raphson and stochastic gradient descent for power flow analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 1, pp. 514–517, 2021.

[146] J. Mulvaney-Kemp, S. Fattahi, and J. Lavaei, "Load variation enables escaping poor solutions of time-varying optimal power flow," in *PESGM*, 2020.

[147] B. Lesieutre, J. Lindberg, A. Zachariah, and N. Boston, "On the distribution of real-valued solutions to the power flow equations," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 165–170.

[148] J. Lindberg, A. Zachariah, N. Boston, and B. C. Lesieutre, "The distribution of the number of real solutions to the power flow equations," *arXiv preprint arXiv:2010.03069*, 2020.

[149] S. Li, D. Tylavsky, D. Shi, and Z. Wang, "Implications of stahl's theorems to holomorphic embedding pt. i: Theoretical convergence," *CSEE Journal of Power and Energy Systems*, vol. 7, no. 4, pp. 761–772, 2021.

[150] A. Dronamraju, S. Li, Q. Li, Y. Li, D. Tylavsky, D. Shi, and Z. Wang, "Implications of stahl's theorems to holomorphic embedding pt. ii: Numerical convergence," *CSEE Journal of Power and Energy Systems*, vol. 7, no. 4, pp. 773–784, 2021.

[151] A. G. Bakirtzis, P. N. Biskas, C. E. Zoumas, and V. Petridis, "Optimal power flow by enhanced genetic algorithm," *IEEE Trans. on power Systems*, vol. 17, no. 2, pp. 229–236, 2002.

[152] M. A. Abido, "Optimal power flow using particle swarm optimization," *International Journal of Electrical Power & Energy Systems*, vol. 24, no. 7, pp. 563–571, 2002.

[153] H. Oh, "A unified and efficient approach to power flow analysis," *Energies*, vol. 12, no. 12, p. 2425, 2019.

[154] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, 2011.

[155] B. Zhang and D. Tse, "Geometry of injection regions of power networks," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 788–797, 2012.

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

[157] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Now Publishers*, 2019.

[158] K. Baker, "Solutions of dc opf are never ac feasible," in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, 2021, pp. 264–268.

[159] Y. Tang and S. Low, "Distributed algorithm for time-varying optimal power flow," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 3264–3270.

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

[161] R. T. Rockafellar, "Lagrange multipliers and optimality," *SIAM review*, vol. 35, no. 2, pp. 183–238, 1993.

[162] H. D. Nguyen and K. S. Turitsyn, "Appearance of multiple stable load flow solutions under power flow reversal conditions," in *2014 IEEE PES General Meeting—Conference & Exposition*. IEEE, 2014, pp. 1–5.

[163] R. Mieth and Y. Dvorkin, "Distribution electricity pricing under uncertainty," *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 2325–2338, 2020.

[164] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," 2020.

[165] M. K. Singh, V. Kekatos, and G. B. Giannakis, "Learning to solve the ac-opf using sensitivity-informed deep neural networks," *ArXiv:2103.14779*, 2021.

[166] K. Baker, "Emulating ac opf solvers for obtaining sub-second feasible, near-optimal solutions," *ArXiv:2012.10031*, 2020.

[167] P. L. Donti, D. Rolnick, and J. Z. Kolter, "Dc3: A learning method for optimization with hard constraints," *arXiv:2104.12225*, 2021.

[168] K. Lehmann, A. Grastien, and P. Van Hentenryck, "Ac-feasibility on tree networks is np-hard," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 798–801, 2016.

[169] R. Jabr, "Radial distribution load flow using conic programming," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1458–1459, 2006.

[170] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, "Semidefinite programming for optimal power flow problems," *International Journal of Electrical Power*

& Energy Systems, vol. 30, no. 6, pp. 383–392, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061507001378

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

[172] ——, "Convex relaxation of optimal power flow—part ii: Exactness," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 2, pp. 177–189, 2014.

[173] J. Lavaei, D. Tse, and B. Zhang, "Geometry of power flows and optimization in distribution networks," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 572–583, mar 2014. [Online]. Available:

[174] B. Zhang and D. Tse, "Geometry of injection regions of power networks," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 788–797, 2013.

[175] F. Wu and S. Kumagai, "Steady-state security regions of power systems," *IEEE Transactions on Circuits and Systems*, vol. 29, no. 11, pp. 703–711, 1982.

[176] J. W. Simpson-Porco, "A theory of solvability for lossless power flow equations—part ii: Conditions for radial networks," *IEEE Transactions on Control of Network Systems*, vol. 5, pp. 1373–1385, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:52304657

[177] A. Ipakchi and F. Albuyeh, "Grid of the future," *IEEE Power and Energy Magazine*, vol. 7, no. 2, pp. 52–62, 2009.

[178] H. D. Nguyen, K. Dvijotham, and K. Turitsyn, "Constructing convex inner approximations of steady-state security regions," *IEEE Transactions on Power Systems*, vol. 34, no. 1, pp. 257–267, 2019.

[179] D. Lee, H. D. Nguyen, K. Dvijotham, and K. Turitsyn, "Convex restriction of power flow feasibility sets," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 3, pp. 1235–1245, 2019.

[180] D. I. Sun, B. Ashley, B. Brewer, A. Hughes, and W. F. Tinney, "Optimal power flow by newton approach," *IEEE Transactions on Power Apparatus and systems*, no. 10, pp. 2864–2880, 1984.

[181] W. Kersting, "Radial distribution test feeders," *2001 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.01CH37194)*, vol. 2, pp. 908–912 vol.2, 1991. [Online]. Available: https://api.semanticscholar.org/CorpusID:65242268

[182] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[183] Smart Electric Power Alliance and Navigant Research, "2017 utility demand response market snapshot," 2017. [Online]. Available: https://sepapower.org/resource/2017-utility-demand-response-market-snapshot

[184] I. Rouf, H. A. Mustafa, M. Xu, W. Xu, R. D. Miller, and M. Gruteser, "Neighborhood watch: security and privacy analysis of automatic meter reading systems," in *ACM Conference on Computer and Communications Security*, 2012.

[185] M. Marcellino, J. Stock, and M. W. Watson, "A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series," *Journal of Econometrics*, vol. 135, pp. 499–526, 2006.

[186] Cplex, IBM ILOG, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[187] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2020. [Online]. Available: `http://www.gurobi.com`

[188] H. Gangammanavar, S. Sen, and V. M. Zavala, "Stochastic optimization of sub-hourly economic dispatch with wind energy," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 949–959, 2015.

[189] S. Xu, W. Wu, Y. Yang, B. Wang, and X. Wang, "Stochastic real-time power dispatch with large-scale wind power integration and its analytical solution," *arXiv preprint arXiv:1905.09480*, 2019.

[190] Y. Gu and L. Xie, "Stochastic look-ahead economic dispatch with variable generation resources," *IEEE Transactions on Power Systems*, vol. 32, no. 1, pp. 17–29, 2016.

[191] S. Sen and Y. Liu, "Mitigating uncertainty via compromise decisions in two-stage stochastic linear programming: Variance reduction," *Operations Research*, vol. 64, no. 6, pp. 1422–1437, 2016.

[192] A. Alqurashi, A. H. Etemadi, and A. Khodaei, "Treatment of uncertainty for next generation power systems: State-of-the-art in stochastic optimization," *Electric Power Systems Research*, vol. 141, pp. 233–245, 2016.

[193] W. Lin, Z. Yang, J. Yu, K. Xie, X. Wang, and W. Li, "Tie-line security region considering time coupling," *IEEE Transactions on Power Systems*, vol. 36, no. 2, pp. 1274–1284, 2021.

[194] T. Li, Z. Li, and W. Li, "Scenarios analysis on the cross-region integrating of renewable power based on a long-period cost-optimization power planning model," *Renewable Energy*, vol. 156, pp. 851–863, 2020. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0960148120306261`

[195] H. Seifi and M. Sepasian, *Electric Power System Planning: Issues, Algorithms and Solutions*, ser. Power Systems. Springer Berlin Heidelberg, 2011. [Online]. Available: https://books.google.com/books?id=PW39l0GgPy8C

[196] T. J. Overbye, X. Cheng, and Y. Sun, "A comparison of the ac and dc power flow models for lmp calculations," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the.* IEEE, 2004, pp. 9–pp.

[197] J. Sun and L. Tesfatsion, "Dc optimal power flow formulation and solution using quadprogj," *Iowa State University Digital Repository*, 2010.

[198] W. Deng, Y. Ji, and L. Tong, "Probabilistic forecasting and simulation of electricity markets via online dictionary learning," *arXiv preprint arXiv:1606.07855*, 2016.

[199] M. Padhee and A. Pal, "Fast dtw and fuzzy clustering for scenario generation in power system planning problems," *arXiv preprint arXiv:2007.00805*, 2020.

[200] J. Li, F. Lan, and H. Wei, "A scenario optimal reduction method for wind power time series," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1657–1658, 2016.

[201] B. G. Gorenstin, N. M. Campodonico, J. P. Costa, and M. V. F. Pereira, "Power system expansion planning under uncertainty," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 129–136, 1993.

[202] M. L. Crow, *Computational methods for electric power systems.* Crc Press, 2015.

[203] Y. Ji, R. J. Thomas, and L. Tong, "Probabilistic forecasting of real-time lmp and network congestion," *IEEE Transactions on Power Systems*, vol. 32, no. 2, pp. 831–841, 2016.

[204] B. Stott, J. Jardim, and O. Alsac, "Dc power flow revisited," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1290–1300, 2009.

[205] D. Tabas and B. Zhang, "Computationally efficient safe reinforcement learning for power systems," 2021. [Online]. Available: https://arxiv.org/abs/2110.10333

[206] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002. [Online]. Available: https://doi.org/10.1137/S1052623499363220

[207] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *NIPS Workshop Autodiff*, 2017.

[208] ——, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019, pp. 8024–8035.

[209] L. Roald, G. Andersson, S. Misra, M. Chertkov, and S. Backhaus, "Optimal power flow with wind power control and limited expected risk of overloads," 06 2016, pp. 1–7.

[210] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[211] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, "Adjustable robust solutions of uncertain linear programs1," *Mathematical Programming*, vol. 99, pp. 351–376, 01 2004.

[212] Y. Ng, S. Misra, L. A. Roald, and S. N. Backhaus, "Statistical learning for dc optimal power flow," *2018 Power Systems Computation Conference (PSCC)*, pp. 1–7, 2018.

[213] F. Fioretto, T. W. Mak, and P. V. Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *AAAI Conference on Artificial Intelligence*, 2019.

[214] M. K. Singh, V. Kekatos, and G. B. Giannakis, "Learning to solve the ac-opf using sensitivity-informed deep neural networks," *IEEE Transactions on Power Systems*, 2022.

[215] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3270–3273, 2020.

[216] M. Li, S. Kolouri, and J. Mohammadi, "Learning to solve optimization problems with hard linear constraints," 08 2022.

[217] R. P. Klump and T. J. Overbye, "Techniques for improving power flow convergence," in *2000 Power Engineering Society Summer Meeting*. IEEE, 2000.

[218] B. Zhang and D. Tse, "Geometry of injection regions of power networks," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 788–797, 2013.

[219] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, 2012.

[220] B. C. Lesieutre, D. K. Molzahn, A. R. Borden, and C. L. DeMarco, "Examining the limits of the application of semidefinite programming to power flow problems," in *2011 49th Annual Allerton Conference*, 2011.

[221] R. Mieth and Y. Dvorkin. Dlmp under uncertainty code supplement. [Online]. Available: `https://github.com/mieth-robert/DLMP_uncertainty_CodeSupplement`

[222] R. D. Zimmerman, C. E. Murillo-sánchez, R. J. Thomas, and L. Fellow, "Matpower steady-state operations, planning and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, pp. 12–19, 2011.

[223] A. S. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," 2019.

[224] F. Fioretto, T. W. K. Mak, and P. V. Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," 2019.

[225] N. Guha, Z. Wang, M. Wytock, and A. Majumdar, "Machine learning for ac optimal power flow," 2019.

[226] A. Castillo and R. P. O'Neill, "Computational performance of solution techniques applied to the acopf," *Federal Energy Regulatory Commission, Optimal Power Flow Paper*, vol. 5, 2013.

[227] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 5930–5934.

[228] H. Lange, B. Chen, M. Berges, and S. Kar, "Learning to solve ac optimal power flow by differentiating through holomorphic embeddings," *arXiv:2012.09622*, 2020.

[229] Y. Zhou, B. Zhang, C. Xu, T. Lan, R. Diao, D. Shi, Z. Wang, and W.-J. Lee, "A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 6, pp. 1128–1139, 2020.

[230] B. Zhang, A. Y. S. Lam, A. Dominguez-Garcia, and D. Tse, "An optimal and distributed method for voltage regulation in power distribution systems," 2015.

[231] J. Lavaei, D. Tse, and B. Zhang, "Geometry of power flows and optimization in distribution networks," *IEEE Transactions on Power Systems*, vol. 29, 04 2012.

[232] S. Bose, D. Gayme, S. Low, and K. Chandy, "Optimal power flow over tree networks," 09 2011, pp. 1342–1348.

[233] S. Sojoudi and J. Lavaei, "Physics of power networks makes hard optimization problems easy to solve," in *2012 IEEE Power and Energy Society General Meeting*, 2012, pp. 1–8.

[234] S. Bose, S. Low, T. Teeraratkul, and B. Hassibi, "Equivalent relaxations of optimal power flow," *IEEE Transactions on Automatic Control*, vol. 60, 01 2014.

[235] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, "Semidefinite programming for optimal power flow problems," *International Journal of Electrical Power & Energy Systems*, vol. 30, pp. 383–392, 2008.

Appendix A

APPENDIX OF CHAPTER 3

A.1 Fundamental Flows

In the DC power flow model, the power flow on the lines are determined by the angle differences. Let δ_i be the angle of bus i. Let $p_{ij}^f = b_{ij}(\delta_i - \delta_j)$ be the flow along the line connecting i and j. If a network has cycles, let buses $1, , \ldots, n_c$ be the buses in a cycle, counted in either clockwise or counterclockwise direction. The weighted sum $p_{12}^f/b_{12} + p_{23}^f/b_{23} + \cdots + p_{n_c1}^f/b_{n_c1} = 0$ and therefore, the flows lie in a subspace.

Repeating the above calculation for every cycle in a network gives that the flows lie in a subspace of dimension $n-1$ for a connected network with n buses. A basis of this subspace is called a set of fundamental flows. There are multiple bases to choose the fundamental flows from. A popular way is to choose a spanning tree and consider the flows on the branches as fundamental, and everything else can be derived from them.

A.2 Proof of Theorem 3.5

Proof. Since $\boldsymbol{\ell}^{\text{new}}$ is in the convex hull of $\mathcal{D}_{trn}$, there are positive coefficients $\alpha_1, \ldots, \alpha_N$ such that

$$\boldsymbol{\ell}^{\text{new}} = \alpha_1 \boldsymbol{\ell}^1 + \cdots + \alpha_N \boldsymbol{\ell}^N,$$

and $\alpha_1 + \ldots, \alpha_n = 1$. By convexity,

$$f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) \leqslant \alpha_1 f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^1) + \cdots + \alpha_N f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^N). \tag{A.1}$$

By the assumption that $f_{\boldsymbol{\theta}}(\boldsymbol{\ell})$ is well-trained, we have

$$\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i) = \boldsymbol{\mu}, \text{ for } i = 1, \cdots, N. \tag{A.2}$$

Using first-order conditions of convex functions, we have $f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) \geqslant f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i) + \boldsymbol{\mu}(\boldsymbol{\ell}^{\text{new}} - \boldsymbol{\ell}^i)$ for all i. Multiplying the i'th equation by α_i and summing gives

$$f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) \geqslant \alpha_1 f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^1) + \cdots + \alpha_N f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^N). \tag{A.3}$$

Combining (A.1) and (A.3) gives

$$f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{\text{new}}) = \alpha_1 f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^1) + \cdots + \alpha_N f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^N). \tag{A.4}$$

This implies the function is linear in the convex hull of $\mathcal{D}_{trn}$ and all the points have the same gradient. ■

A.3 Proof of Theorem 3.6

Suppose $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ is a convex function. Given $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$ in the domain of g and let $\boldsymbol{\mu}^i = \nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i)$. Let $\boldsymbol{\ell}^{\text{new}}$ be a point in the convex hull of $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$ and denote $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new}) = \boldsymbol{\mu}$. By the convexity of $g_{\boldsymbol{\theta}}$, we have

$$(\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i) - \nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new}))^T (\boldsymbol{\ell}^{\text{new}} - \boldsymbol{\ell}^i) \leqslant 0, \forall i. \tag{A.5}$$

The inequalities in (A.5) constrain the values that $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ can take. We show that these inequalities actually describe a bounded polytope in $\mathbb{R}^n$ through a proof by contradiction.

Suppose the region defined by the inequalities in (A.5) is not bounded. Then $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ can be scaled arbitrarily and all of the inequalities in (A.5) would still hold. Then we can take the norm of $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})$ to be large enough such that it would dominate the $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^i)$ terms. Then (A.5) becomes

$$\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T (\boldsymbol{\ell}^{\text{new}} - \boldsymbol{\ell}^i) \geqslant 0, \forall i. \tag{A.6}$$

Since $\boldsymbol{\ell}^{\text{new}}$ is in the convex hull of $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$, we can write it as $\boldsymbol{\ell}^{\text{new}} = \alpha_1 \boldsymbol{\ell}^1 + \cdots + \alpha_N \boldsymbol{\ell}^N$ and $\alpha_i \geqslant 0$ and sums up to 1. Substituting this into (A.6) and rearranging the terms, we have

$$\boldsymbol{\ell}^N \nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^i \geqslant \sum_{i=1}^{N} \alpha_i \nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^i.$$

By the assumption that $N \geqslant n+1$ and $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^N$ are not in a lower dimensional subspace of $\mathbb{R}^n$, $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^i$ will be nonzero for at least two i's. But it is not possible to have a convex combination of scalars (the $\nabla_{\boldsymbol{\ell}} f_{\boldsymbol{\theta}}(\boldsymbol{\ell}^{new})^T \boldsymbol{\ell}^i$'s) larger than every scalar in the set when at least two are nonzero (this follows from Farkas' lemma). This contradicts the assumption that the polytope created by (A.5) is unbounded.

A.4 Quadratic Costs

The DCOPF problem with quadratic cost is:

$$J(\boldsymbol{\ell}) = \min_{\mathbf{x},\mathbf{p}^e} \quad \sum_{i=1}^{n} \frac{q_i}{2} p_i^{g2} + c_i p_i^g \tag{A.7a}$$

$$\text{s.t. (1b), (1c) and (1d)}, \tag{A.7b}$$

where q_i and c_i are the cost coefficients. As with the linear cost case, we assume that the multipliers ($\boldsymbol{\mu}$) with respect to the power balance equations have been learned. If q_i's are not zero, the dual of (A.7) is

$$\max_{\boldsymbol{\mu},\bar{\boldsymbol{\lambda}},\underline{\boldsymbol{\lambda}},\bar{\boldsymbol{\nu}},\underline{\boldsymbol{\nu}}} \quad \boldsymbol{\mu}^T\ \boldsymbol{\ell} - \underline{\boldsymbol{\lambda}}^T \bar{\mathbf{p}}^f - \bar{\boldsymbol{\lambda}}^T \bar{\mathbf{p}}^f - \bar{\boldsymbol{\nu}}^T \bar{\mathbf{p}}^g \tag{A.8a}$$

$$\text{s.t.} \quad \mathbf{Q}\mathbf{x} + \mathbf{c} - \boldsymbol{\mu} - \underline{\boldsymbol{\nu}} + \bar{\boldsymbol{\nu}} = \mathbf{0} \tag{A.8b}$$

$$-\mathbf{M}^{B^T}\boldsymbol{\mu} - \mathbf{M}^{F^T}\underline{\boldsymbol{\lambda}} + \mathbf{M}^{F^T}\bar{\boldsymbol{\lambda}} = \mathbf{0} \tag{A.8c}$$

$$\bar{\boldsymbol{\nu}} \geqslant \mathbf{0}, \underline{\boldsymbol{\nu}} \geqslant \mathbf{0}, \bar{\boldsymbol{\lambda}} \geqslant \mathbf{0}, \underline{\boldsymbol{\lambda}} \geqslant \mathbf{0}, \tag{A.8d}$$

where $\mathbf{Q}$ is a diagonal matrix with the value of q_i on the i'th diagonal. There are two differences between the linear and quadratic costs. The first is that the constraint associated with the generations, (A.8b), also include the primal variables $\mathbf{x}$. The second is that a quadratic program may not have the same number of binding constraints as the variables.

We use the following simple lemma to determine whether a generator constraint is binding by following simple economic principles. Specifically, given the optimal LMP $\boldsymbol{\mu}^*$, p_i^g is associated with the following active/inactive constraints:

$$x_i = \begin{cases} \bar{p}_i^g, & \text{if } \mu_i^* - c_i - 2q_i\bar{p}_i^g > 0 \\ 0 & \text{if } \mu_i^* - c_i < 0 \\ (0,\, \bar{p}_i^g)\,, & \text{otherwise} \end{cases} \tag{A.9}$$

Once $\boldsymbol{\mu}$ is known, the dual problem associated with $\underline{\boldsymbol{\lambda}}$ and $\overline{\boldsymbol{\lambda}}$ is identical to the linear cost case and the binding line constraints can be recovered through the same process.

Once we identify all of the binding constraints, we can encode it into a matrix of the form $\mathbf{M}\mathbf{y} = \mathbf{a}$, where $\mathbf{y}$ is the concatenation of $\mathbf{x}$ and $\mathbf{p}^e$. Here the number of constraints

(rows of $\mathbf{M}$) can be less than the number of variables and we still need to solve the following optimization problem:

$$\min \frac{1}{2}\mathbf{y}^T\hat{\mathbf{Q}}\mathbf{y} + \hat{\mathbf{c}}^T\mathbf{y} \tag{A.10a}$$

$$\text{s.t. } \mathbf{M}\mathbf{y} = \mathbf{a}, \tag{A.10b}$$

where $\hat{\mathbf{Q}}$ is a diagonal matrix with $(q_1, \ldots, q_n, 0, \ldots 0)$ on its diagonal and $\hat{\mathbf{c}} = (c_1, \ldots, c_n, 0, \ldots, 0)$. Fortunately (A.10) can be solved as a linear system. Following standard quadratic programming results, let $\boldsymbol{\tau}^*$ be the optimal Lagrangian multiplier of (A.10b), then the optimal solution of (A.10) is given by the following linear system

$$\begin{bmatrix} \hat{\mathbf{Q}} & \mathbf{M}^T \\ \mathbf{M} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}^* \\ \boldsymbol{\tau}^* \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{c}} \\ \mathbf{a} \end{bmatrix}. \tag{A.11}$$

Once the active constraints are identified, a linear system of equations can again be solved to find the optimal solutions.

Appendix B

APPENDIX OF CHAPTER 4

B.1 Expressions of B, F *and* E

Suppose we use $\mathcal{E}$ to denote the set of all lines in the power system and (i, j) the line connecting bus-i and bus-j. Without loss of generality, we can assume the line (i, j) is the k-th out of all lines. Let b_{ij} be the susceptance for the line (i, j), then the flow on line (i, j) is $p_{ij}^f = b_{i,j}(\delta_i - \delta_j)$. The nodal power injection by bus-i is $p_i^{inj} = \sum_{k:(i,k)\in\mathcal{E}} p_{ik}^f = \sum_{k:(i,k)\in\mathcal{E}} b_{i,j}(\delta_i - \delta_j)$. As a result, the matrix $\mathbf{B}$ that transforms the phase angle $\boldsymbol{\delta}^{ns} \in \mathbb{R}^{n-1}$ into the nodal power injections at all buses can be expressed as

$$\forall i, j : B_{ij} = \begin{cases} -b_{ij}, \text{ if } (i,j) \in \mathcal{E} \text{ and } i \neq j \\ \sum_{k:(k,j)\in\mathcal{E}} b_{kj}, \text{ if } (i,j) \in \mathcal{E} \text{ and } i = j \\ 0, \text{ otherwise.} \end{cases}$$

The matrix $\mathbf{F}$ that maps $\boldsymbol{\delta}^{ns}$ to flows on all lines is given by

$$F_{ki} = b_{ij}, F_{kj} = -b_{ij},$$
$$\forall k \in \{1, \cdots, m\}, i, j \in \{1, \cdots, n\} \text{ and } i \neq j.$$

The incidence matrix $\mathbf{E}$ has a row for each line and a column for each node. To construct $\mathbf{E}$, we can treat the power network as a directed graph, where each line has a starting node (source) and an ending node (target), For instance, consider a line denoted by (i, j) as the k-th among all lines, with $i > j$. In this setup, we assume that the node with a smaller index, i.e., node i, serves as the source node, while node j is designated as the target node. Accordingly, the entry $\mathbf{E}_{[[k,i]]}$ in $\mathbf{E}$ is assigned a value of 1, and the entry $\mathbf{E}_{[[k,j]]}$ is assigned -1.

B.2 SGD Updating Rules

The stochastic gradients of the loss function with respect to $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ at a randomly chosen data point $\bar{\boldsymbol{\ell}}^i$ are calculated using the chain rule in the backward pass, which can be

expressed as follows

$$\frac{\partial L^i(\boldsymbol{\theta}^I,\boldsymbol{\theta}^R)}{\partial \boldsymbol{\theta}^I} = \widetilde{c}'\frac{\partial f^I(\bar{\boldsymbol{\ell}}^i;\boldsymbol{\theta}^I)}{\partial \boldsymbol{\theta}^I} + \frac{1}{K}\sum_{k=1}^{K}\frac{\partial f^R(\tilde{s}_\ell^{ik};\boldsymbol{\theta}^R)}{\partial \tilde{s}_\ell^{ik}}\frac{\partial \tilde{s}^{ik}}{\partial f^I(\bar{\boldsymbol{\ell}}^i;\boldsymbol{\theta}^I)}\frac{\partial f^I(\bar{\boldsymbol{\ell}}^i;\boldsymbol{\theta}^I)}{\partial \boldsymbol{\theta}^I} \tag{B.1a}$$

$$\frac{\partial L^i(\boldsymbol{\theta}^I,\boldsymbol{\theta}^R)}{\partial \boldsymbol{\theta}^R} = \frac{1}{K}\sum_{k=1}^{K}\frac{\partial f^R(\tilde{s}_\ell^{ik};\boldsymbol{\theta}^R)}{\partial \boldsymbol{\theta}^R}. \tag{B.1b}$$

where $\widetilde{c}'$ is the derivative of $\widetilde{c}(\cdot)$. At each iteration t, SGD repeats the following updates on $\boldsymbol{\theta}^I$ and $\boldsymbol{\theta}^R$ until a certain stopping criterion is reached:

$$\boldsymbol{\theta}^{I(t+1)} \longleftarrow \boldsymbol{\theta}^{I(t)} - \rho\frac{\partial L^i(\boldsymbol{\theta}^I,\boldsymbol{\theta}^R)}{\partial \boldsymbol{\theta}^I}\Big|_{\boldsymbol{\theta}^{I(t)}} \tag{B.2a}$$

$$\boldsymbol{\theta}^{R(t+1)} \longleftarrow \boldsymbol{\theta}^{R(t)} - \rho\frac{\partial L^i(\boldsymbol{\theta}^I,\boldsymbol{\theta}^R)}{\partial \boldsymbol{\theta}^R}\Big|_{\boldsymbol{\theta}^{R(t)}}, \tag{B.2b}$$

where ρ denotes the step size. Note that all the backward pass gradients given by (B.1) can be computed using the automatic differentiation engine in machine learning libraries, such as *autograd* in Pytorch [207, 208], and the SGD updating rules in (B.2) can also be implemented therein.

B.3 Proof of Theorem 4.2

To show that the polyhedron given by (4.11) is bounded, we use the definition of a bounded polyhedra: *a polyhedra is bounded if* $\exists K > 0$ *such that* $\|\boldsymbol{\delta}^{ns}\| \leqslant K$*, for all* $\boldsymbol{\delta}^{ns} \in \mathcal{P}_{feas}$.

From Appendix B.1, we know that the flow on line (i,j) can be expressed as $p_{ij}^f = b_{i,j}(\delta_i - \delta_j)$; therefore, we can rewrite the polyhedra in (4.11) as

$$-\bar{p}_{i,j}^f \leqslant b_{i,j}(\delta_i - \delta_j) \leqslant \bar{p}_{i,j}^f, \forall (i,j) \in \mathcal{E},$$

which are equivalent to

$$\frac{-\bar{p}_{i,j}^f}{b_{i,j}} \leqslant \delta_i - \delta_j \leqslant \frac{\bar{p}_{i,j}^f}{b_{i,j}}, \forall (i,j) \in \mathcal{E}, \tag{B.3}$$

that is, both δ_i and δ_j must be bounded, otherwise, (B.3) would be violated. Since every bus in the system must be connected to at least one other bus, (B.3) implies that $\exists K_i \in \mathbb{R}_+$ such that $|\delta_i| \leqslant K_i, \forall i \in \{1,\cdots,n\}$, therefore, we can choose $K = \max_i\{K_i\}$ and then we have $\|\boldsymbol{\delta}^{ns}\| \leqslant K$. By definition, the polyhedra given by (4.11) is bounded.

B.4 Proof of Proposition 4.4.2

By Definition 4.3, we can express the gauge function associated with the polyhedral set $\mathcal{P} = \{\mathbf{z} \in \mathbb{R}^n | \mathbf{A}\mathbf{z} \leqslant \mathbf{b}, \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^n\}$ as the optimization problem $g_{\mathcal{P}}(\mathbf{z}) = \min\{\lambda | \mathbf{A}\mathbf{z} \leqslant \lambda\mathbf{b}\}$, which is equivalent to finding a value of λ such that $\mathbf{a}_i^T\mathbf{z} \leqslant \lambda b_i, \forall i \in \{1, \cdots, m\}$, that is, $\lambda \geqslant \frac{\mathbf{a}_i^T\mathbf{z}}{b_i}, \forall i \in \{1, \cdots, m\}$. Therefore, the optimal value of λ, namely, the value of the gauge function $g_{\mathcal{P}}(\mathbf{z})$, can be given by $\max_{i=1,\cdots,m}\left\{\frac{\mathbf{a}_i^T\mathbf{z}}{b_i}\right\}$.

B.5 Proof of Theorem 4.5

Since $g_{\mathcal{P}_{feas}}(\mathbf{u}) = \max_{i=1,\cdots,m}\left\{\frac{\mathbf{a}_i^T\mathbf{u}}{b_i}\right\}$, it follows that $\frac{\mathbf{a}_k^T\mathbf{u}}{b_k} \leqslant g_{\mathcal{P}_{feas}}(\mathbf{u})$, which is equivalent to $\frac{1}{g_{\mathcal{P}_{feas}}(\mathbf{u})}\mathbf{a}_k^T\mathbf{u} \leqslant b_k$, for any index k. Given that $\mathbf{u}$ lies within the unit hypercube $\mathbb{B}_\infty \triangleq \{\mathbf{z} \in \mathbb{R}^n | -1 \leqslant z_i \leqslant 1, \forall i\}$, it follows that $\|\mathbf{u}\|_\infty \leqslant 1$. Bringing these together, we have $\frac{\|\mathbf{u}\|_\infty}{g_{\mathcal{P}_{feas}}(\mathbf{u})}\mathbf{a}_k^T\mathbf{u} \leqslant b_k$, for any k. This series of inequalities can be compactly expressed as $\frac{\|\mathbf{u}\|_\infty}{g_{\mathcal{P}_{feas}}(\mathbf{u})}\mathbf{A}\mathbf{u} \leqslant \mathbf{b}$. By using (4.12), we can then conclude that $\mathbf{A}\boldsymbol{\delta}_u^{ns} \leqslant \mathbf{b}$, which indicates that $\boldsymbol{\delta}_u^{ns}$ is a feasible point lying within the set $\mathcal{P}_{feas}$.

B.6 Proof of Theorem 4.6

Let $\mathcal{P} = \{\mathbf{z} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{z} \leqslant \mathbf{b}\}$ and $\mathcal{Q} = \{\mathbf{z} \in \mathbb{R}^n \mid \mathbf{C}\mathbf{z} \leqslant \mathbf{d}\}$ with $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m_{++}, \mathbf{C} \in \mathbb{R}^{k\times n}$, and $\mathbf{d} \in \mathbb{R}^k_{++}$. Let $\mathcal{A}_{ij}$ be the polytope described as $\{\mathbf{z} \in \mathcal{P} \mid i \in \arg\max_{i=1,\cdots,m}\frac{\mathbf{a}_i^T\mathbf{z}}{b_i}, j \in \arg\max_{j=1,\cdots,k}\frac{\mathbf{c}_j^T\mathbf{z}}{d_j}\}$. The set $\{\mathcal{A}_{ij} \mid i \in 1, \cdots, m,\ j \in 1, \cdots, k\}$ forms a polyhedral partition of $\mathcal{P}$, and the gauge map is an analytic function on the interior of each $\mathcal{A}_{ij}$ except when $\mathbf{c}_j^T\mathbf{z} = 0$ or $\mathbf{z} = 0$. Specifically, the gauge map on the interior of $\mathcal{A}_{ij} \subseteq \mathcal{P}$ can be written as $G(\mathbf{z} \mid \mathcal{P}, \mathcal{Q}) = \frac{\mathbf{a}_i^T\mathbf{z}/b_i}{\mathbf{c}_j^T\mathbf{z}/d_j}\mathbf{z}$.

For any $j \in 1, \cdots, k$, $\mathbf{c}_j^T\mathbf{z} = 0$ if and only if $\mathbf{z} = 0$: since Q forms a full-dimensional and bounded polytope, $\mathbf{C}$ must be full-rank and tall ($k > n$). Thus, $\mathbf{C}\mathbf{z} = 0$ if and only if $\mathbf{z} = 0$.

We can now justify the choice $G(0 \mid \mathcal{P}, \mathcal{Q}) := 0$ as follows. Let $\mathbf{z} = \alpha\mathbf{h}$ for some $\alpha > 0$ and $\mathbf{h} \in \mathbb{R}^n\backslash\{0\}$. There exist some (i, j) and sufficiently small $\varepsilon > 0$ such that $\mathbf{z} \in \mathcal{A}_{ij}\ \forall\ \alpha \in (0, \varepsilon)$. The limit of $\frac{\mathbf{a}_i^T\mathbf{z}/b_i}{\mathbf{c}_j^T\mathbf{z}/d_j}\mathbf{z}$ as $\alpha \to 0$ is equal to $0 \in \mathbb{R}^n$.

By the above analysis, the gauge map is *piecewise analytic under analytic partition* (PAP) on $\mathcal{P}$ which implies desirable properties for automatic differentiation [129]. Specifi-

cally, PAP functions can be composed with one another (they obey a chain rule), they are differentiable almost everywhere (except possibly on a set of measure zero), and standard automatic differentiation tools will compute the derivatives at all points where the function is differentiable.

B.7 Formulation of Affine Policy

Given a first stage dispatch $\mathbf{p}^I$, we adopt the following affine policy to approximate the recourse dispatch $\mathbf{p}^R(\boldsymbol{\omega}^k)$ by representing it as an affine function of the load realization $\boldsymbol{\ell}(\boldsymbol{\omega}^k)$:

$$\mathbf{p}^R(\boldsymbol{\omega}^k) = \boldsymbol{\xi}(\mathbf{1}^T\boldsymbol{\ell}(\boldsymbol{\omega}^k) - \mathbf{1}^T\mathbf{p}^I). \tag{B.4}$$

Here, $\boldsymbol{\xi} = [\xi_1, \cdots, \xi_n]^T \in \mathbb{R}^n$ denotes the participation factor vector, which represents the contribution of each generator to balancing the discrepancy between the actual load and the first-stage dispatch. These participation factors are subject to the following constraints:

$$\xi_i \geqslant 0, \forall i \in \mathcal{G},\ \xi_i = 0, \forall i \in \mathcal{N}/\mathcal{G},\ \sum_{i\in\mathcal{G}} \xi_i = 1, \tag{B.5}$$

where the set $\mathcal{G}$ represents all buses hosting generators, and the set $\mathcal{N}/\mathcal{G}$ includes all buses except for those connected to generators. Note that the values of $\boldsymbol{\xi}$ are determined during the first stage and are not influenced by the particular load realization in the second stage.

To implement the affine policy, we start by substituting the variable $\mathbf{p}^R(\boldsymbol{\omega}^k)$ with the affine function (B.4) in the original problem formulations (4.3) and (4.6). This enables us to solve the following approximation problems to determine the values of $\boldsymbol{\xi}$:

Approximation problem of risk-limiting dispatch:

$$\min_{\substack{\mathbf{p}^I,\boldsymbol{\xi}\\ \{\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k)\}_{k=1}^K}} \quad \mathbf{c}^T\mathbf{p}^I + \frac{1}{K}\sum_{k=1}^{K} \mathbf{q}^T[\boldsymbol{\xi}(\mathbf{1}^T\boldsymbol{\ell}(\boldsymbol{\omega}^k) - \mathbf{1}^T\mathbf{p}^I)]^+$$

$$\text{s.t. } (4.3\text{b}) - (4.3\text{e}), (\text{B}.4), (\text{B}.5).$$

Approximation problem of reserve scheduling:

$$\min_{\substack{\mathbf{p}^I,\widehat{\mathbf{r}},\breve{\mathbf{r}},\\ \boldsymbol{\xi},\{\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k)\}_{k=1}^K}} \quad \mathbf{c}^T\mathbf{p}^I + \boldsymbol{\gamma}^T(\widehat{\mathbf{r}}+\breve{\mathbf{r}}) + \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{q}^{\mathrm{res}T}\Big([\boldsymbol{\xi}(\mathbf{1}^T\boldsymbol{\ell}(\boldsymbol{\omega}^k)-\mathbf{1}^T\mathbf{p}^I)-\widehat{\mathbf{r}}]^+-[\boldsymbol{\xi}(\mathbf{1}^T\boldsymbol{\ell}(\boldsymbol{\omega}^k)-\mathbf{1}^T\mathbf{p}^I)+\breve{\mathbf{r}}]^-\Big)\right)$$
$$\text{s.t. (4.6b)} - \text{(4.6h)}, \text{(B.4)}, \text{(B.5)}.$$

Once determined, $\boldsymbol{\xi}$ is treated as fixed and we solve the following simplified linear programs to obtain the first-stage decisions for a set of anticipated load realizations:

Implement affine policy for risk-limiting dispatch:

$$\min_{\substack{\mathbf{p}^I,\\ \{\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k)\}_{k=1}^K}} \quad \mathbf{c}^T\mathbf{p}^I + \frac{1}{K}\sum_{k=1}^{K}\mathbf{q}^{resT}[\boldsymbol{\xi}(\mathbf{1}^T\boldsymbol{\ell}(\boldsymbol{\omega}^k)-\mathbf{1}^T\mathbf{p}^I)]^+ \tag{B.8}$$
$$\text{s.t. (4.3b)} - \text{(4.3e)}, \text{(B.4)}.$$

Implement affine policy for reserve scheduling:

$$\min_{\substack{\mathbf{p}^I,\widehat{\mathbf{r}},\breve{\mathbf{r}},\\ \{\boldsymbol{\delta}^{ns}(\boldsymbol{\omega}^k)\}_{k=1}^K}} \quad \mathbf{c}^T\mathbf{p}^I + \boldsymbol{\gamma}^T(\widehat{\mathbf{r}}+\breve{\mathbf{r}}) + \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{q}^{\mathrm{res}T}\Big([\boldsymbol{\xi}(\mathbf{1}^T\boldsymbol{\ell}(\boldsymbol{\omega}^k)-\mathbf{1}^T\mathbf{p}^I)-\widehat{\mathbf{r}}]^+-[\boldsymbol{\xi}(\mathbf{1}^T\boldsymbol{\ell}(\boldsymbol{\omega}^k)-\mathbf{1}^T\mathbf{p}^I)+\breve{\mathbf{r}}]^-\Big)\right) \tag{B.9}$$
$$\text{s.t. (4.6b)} - \text{(4.6h)}, \text{(B.4)}.$$

In contrast to the original problems presented in (4.3) and (4.6), the linear programs in (B.8) and (B.9) reduce numbers of both decision variables and constraints. As a result, these simplified linear programs can be solved more quickly.

Appendix C

APPENDIX OF CHAPTER 5

C.1 Determine global minimum for ACOPF

In Section 5.5.1 and 5.5.2, we find two solutions to the supply/balance equality constraint, which satisfy the inequalities in (5.12) or (5.23). In this part, we give the reason why the smaller solution in (5.12) (or (5.23)) is the global minimum and the larger solution is the local minimum.

Let us subtract the power received at the load bus from the generation at the generator, then we have the transmission loss as follows:

$$\begin{aligned}\text{loss} &= g - g\cos(\delta) + b\sin(\delta) - (-g + g\cos(\delta) + b\sin(\delta)) \\ &= 2g(1-\cos(\delta)).\end{aligned}$$

Due to the periodicity of arctangent function, the larger value $\bar{\delta}$ must be larger than $\pi/2$. Then the loss at $\delta^\star$ is smaller than the loss at $\bar{\delta}$. So $\delta^\star$ is an more optimal solution than $\bar{\delta}$. Since there are only two solutions for this example, $\delta^\star$ must be the global minimum and $\bar{\delta}$ is the strict local minimum.

C.2 Determine global minimum for the Lagrangian

In this part, we determine the global minimum of the Lagrangian problem for the 2-bus network, where we fix the voltage magnitudes and optimize over the angles.

Let us denote the two solutions of the Lagrangian problem in (5.6) as $\hat{\delta}$ and $\bar{\delta}$, and the multipliers associated with them are $\hat{\mu}$ and $\bar{\mu}$, respectively. Then from (5.17), we have

$$-\frac{\pi}{2} < \hat{\delta} = \tan^{-1}\left(\frac{\hat{\mu}-c'}{\hat{\mu}+c'}\frac{b}{g}\right) < \tan^{-1}\left(\frac{b}{g}\right) \tag{C.1a}$$

$$\frac{\pi}{2} < \bar{\delta} = \tan^{-1}\left(\frac{\bar{\mu}-c'}{\bar{\mu}+c'}\frac{b}{g}\right) + \pi < \tan^{-1}\left(\frac{b}{g}\right) + \pi. \tag{C.1b}$$

Also we can represent the multiplier using δ by rearranging the terms in (5.16). We take $(\hat{\delta}, \hat{\mu})$ as an example, and $\bar{\mu}$ can be represented using $\bar{\delta}$ in a similar way. The expression of $\hat{\mu}$ in terms of $\hat{\delta}$ is

$$\hat{\mu} = -\frac{g\sin(\hat{\delta}) + b\cos(\hat{\delta})}{g\sin(\hat{\delta}) - b\cos(\hat{\delta})}. \tag{C.2}$$

Now let us write out the second-order derivative of the Lagrangian function, and plug (C.2) into it. Then we have:

$$\begin{aligned}\mathcal{L}''_{\hat{\mu}}(\hat{\delta}) &= (1+\hat{\mu})g\cos(\hat{\delta}) - (1-\hat{\mu})b\sin(\hat{\delta}) \\ &= -\frac{2gb}{g\cos(\hat{\delta})(\tan(\hat{\delta}) - \frac{b}{g})}.\end{aligned}$$

Using the inequalities in (C.1), we have

$$\begin{aligned}\mathcal{L}''_{\hat{\mu}}(\hat{\delta}) &> 0, \\ \mathcal{L}''_{\bar{\mu}}(\bar{\delta}) &< 0.\end{aligned}$$

Based on the first-order and second-order optimality conditions, $\hat{\delta}$ is the minimum of the Lagrangian problem, and $\bar{\delta}$ is the maximum.

C.3 Voltage Inequality Constraints

In this part, we prove that not all inequality constraints in (5.22c) are inactive by contradiction. We first suppose all inequality constraints in (5.22c) are inactive, and convert (5.22) to the penalized unconstrained formulation:

$$\mathcal{L}_\rho(\mathbf{x}) = f(\mathbf{x}) + \rho/2[h(\mathbf{x})]^2. \tag{C.3}$$

Assume ρ is sufficiently large, then (C.3) can be viewed as being equivalent to the original problem (5.22). Let us take gradients of $\mathcal{L}_\rho(\mathbf{x})$ with respect to V_1 and V_2 at a feasible solution $\tilde{\mathbf{x}}$. Since $\tilde{\mathbf{x}}$ satisfies $h(\tilde{\mathbf{x}}) = 0$, the terms multiplied by ρh in the gradients can be ignored. So the gradients are given by

$$\frac{\partial \mathcal{L}_\rho}{\partial V_1} = 2g\tilde{V}_1 - \tilde{V}_2(g\cos(\tilde{\delta}) - b\sin(\tilde{\delta})) \tag{C.4a}$$

$$\frac{\partial \mathcal{L}_\rho}{\partial V_2} = -\tilde{V}_1(g\cos(\tilde{\delta}) - b\sin(\tilde{\delta})). \tag{C.4b}$$

1) If $\frac{\partial \mathcal{L}_\rho}{\partial V_1} = 0$, then we have

$$g\cos(\tilde{\delta}) - b\sin(\tilde{\delta}) = 2g\frac{\tilde{V}_1}{\tilde{V}_2} \ (\tilde{V}_2 \neq 0). \tag{C.5}$$

Plug (C.5) into (C.4b) and we get

$$\frac{\partial \mathcal{L}_\rho}{\partial V_2} = -2g\frac{\tilde{V}_1^2}{\tilde{V}_2} < 0.$$

This means if V_1 is inactive, then V_2 must be on the boundary of the constraint set.
2) Suppose $\frac{\partial \mathcal{L}_\rho}{\partial V_2} = 0$. Since $\tilde{V}_1 \neq 0$, we have

$$g\cos(\tilde{\delta}) - b\sin(\tilde{\delta}) = 0. \tag{C.6}$$

If we plug (C.6) into (C.4a), then we have

$$\frac{\partial \mathcal{L}_\rho}{\partial V_1} = 2g\tilde{V}_1 > 0.$$

That is, if V_2 is inactive, then V_1 must be on the boundary of the constraint set. Therefore one of V_1 and V_2 must be binding, and (5.22) can be reduced to the bivariate optimization problem.

C.4 Hessian matrix of the Lagrangian

In this part, we derive the Hessian matrix of the Lagrangian function for problem (5.22), where we optimize both voltage magnitudes and angles for a 2-bus network. In Appendix C.3, we have shown that one of V_1 and V_2 must be binding, so here we consider the cases where V_1 is binding or V_2 is binding separately.

We first suppose V_1 is inactive (equivalently, V_2 is binding). Then the Hessian matrix of the Lagrangian is

$$\nabla^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}} = \begin{pmatrix} \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial \delta^2} & \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial \delta \partial V_1} \\ \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial \delta \partial V_1} & \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial V_1^2} \end{pmatrix}.$$

The two leading principal minors of $\nabla^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}$ at a feasible solution $\tilde{\mathbf{x}}$ are

$$\begin{aligned}
\Delta_1(\tilde{\mathbf{x}}) &= \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial \delta^2} \\
&= \tilde{V}_1 \tilde{V}_2[(1+\tilde{\mu})g\cos(\tilde{\delta}) - (1-\tilde{\mu})b\sin(\tilde{\delta})] \\
\Delta_2(\tilde{\mathbf{x}}) &= \nabla^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}} \\
&= 2g\Delta_1(\tilde{\mathbf{x}}) - \tilde{V}_2^2[(1+\tilde{\mu})g\sin(\tilde{\delta}) + (1-\tilde{\mu})b\cos(\tilde{\delta})]^2
\end{aligned}$$

where $\tilde{\mu}$ is the dual solution associated with $\tilde{\mathbf{x}}$. If $(\tilde{\mathbf{x}}, \tilde{\mu})$ are the optimal primal-dual solutions, then we can write out the first-order optimality condition of the Lagrangian w.r.t. δ:

$$(1+\tilde{\mu})g\sin(\tilde{\delta}) + (1-\tilde{\mu})b\cos(\tilde{\delta}) = 0. \tag{C.8}$$

From (C.8), $\Delta_2(\tilde{\mathbf{x}})$ can be simplified as

$$\Delta_2(\tilde{\mathbf{x}}) = 2g\Delta_1(\tilde{\mathbf{x}}).$$

Also, we can represent $\tilde{\mu}$ in terms of $\tilde{\delta}$:

$$\tilde{\mu} = -\frac{g\sin(\tilde{\delta}) + b\cos(\tilde{\delta})}{g\sin(\tilde{\delta}) - b\cos(\tilde{\delta})}. \tag{C.9}$$

If we plug (C.9) into $\Delta_1(\tilde{\mathbf{x}})$, then we have

$$\Delta_1(\tilde{\mathbf{x}}) = \tilde{V}_1\tilde{V}_2 \frac{-2gb}{g\cos(\tilde{\delta})(\tan(\tilde{\delta}) - \frac{b}{g})}.$$

Following from the inequalities in (5.23), $\Delta_1(\tilde{\mathbf{x}})$ and hence $\Delta_2(\tilde{\mathbf{x}})$ are positive at the global minimum and negative at the local minimum. Therefore, Theorem 5.3 holds for the case where V_1 is inactive and V_2 is binding.

Now we suppose V_2 is inactive (equivalently, V_1 is binding). Then the Hessian matrix of the Lagrangian is

$$\tilde{\nabla}^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}} = \begin{pmatrix} \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial \delta^2} & \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial \delta \partial V_2} \\ \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial \delta \partial V_2} & \frac{\partial^2 \mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial V_2^2} \end{pmatrix}. \tag{C.10}$$

The two leading principal minors of $\tilde{\nabla}^2\mathcal{L}_{\lambda,\boldsymbol{\mu}}$ at a feasible solution $\tilde{\mathbf{x}}$ are

$$\begin{aligned}\tilde{\Delta}_1(\tilde{\mathbf{x}}) &= \frac{\partial^2\mathcal{L}_{\lambda,\boldsymbol{\mu}}}{\partial\delta^2} \\ &= \tilde{V}_1\tilde{V}_2\frac{-2gb}{g\cos(\tilde{\delta})(\tan(\tilde{\delta})-\frac{b}{g})} \qquad \text{(C.11a)} \\ \tilde{\Delta}_2(\tilde{\mathbf{x}}) &= \tilde{\nabla}^2\mathcal{L}_{\lambda,\boldsymbol{\mu}} \\ &= 2g\tilde{\mu}\tilde{\Delta}_1(\tilde{\mathbf{x}}) - \tilde{V}_1^2[(1+\tilde{\mu})g\sin(\tilde{\delta}) + (1-\tilde{\mu})b\cos(\tilde{\delta})]^2 \qquad \text{(C.11b)}\end{aligned}$$

where $\tilde{\mu}$ is the dual solution associated with $\tilde{\mathbf{x}}$. If $(\tilde{\mathbf{x}}, \tilde{\mu})$ are the optimal primal-dual solutions, then the first-order optimality condition w.r.t. δ takes the same form as in (C.8). Then $\tilde{\Delta}_2(\tilde{\mathbf{x}})$ in (C.11b) can be simplified as

$$\tilde{\Delta}_2(\tilde{\mathbf{x}}) = 2g\tilde{\mu}\Delta_1(\tilde{\mathbf{x}}). \qquad \text{(C.12a)}$$

Since the multiplier $\tilde{\mu}$ represents the marginal price and is positive at the global minimum, $\tilde{\Delta}_2(\tilde{\mathbf{x}})$ has the same sign as $\tilde{\Delta}_1(\tilde{\mathbf{x}})$. From the inequalities in (5.23), $\tilde{\Delta}_1(\tilde{\mathbf{x}})$ is positive, hence the Hessian matrix $\tilde{\nabla}^2\mathcal{L}_{\lambda,\boldsymbol{\mu}}$ is positive definite at the global minimum. For the local minimum, since $\tilde{\Delta}_1(\tilde{\mathbf{x}})$ is negative from (5.23), the Hessian matrix $\tilde{\nabla}^2\mathcal{L}_{\lambda,\boldsymbol{\mu}}$ cannot be positive definite. This means it is either negative definite or indefinite at the local minimum. Therefore, Theorem 5.3 also holds for the case where V_2 is inactive and V_1 is binding.